Пятое практическое занятие

Работа с коллекциями. Адресная светодиодная лента

Светодиодные ленты

Светодиодные ленты - это универсальный и эффективный источник освещения, который может использоваться для решения широкого спектра задач. Благодаря своим преимуществам, светодиодные ленты становятся все более популярными в быту и на производстве.

Что такое светодиодная лента?

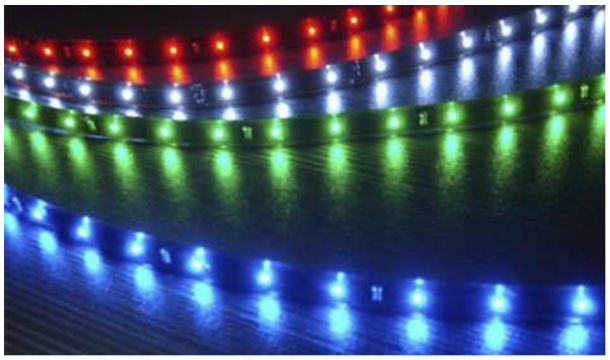
Светодиодная лента - это гибкая печатная плата, на которой расположены светодиоды и резисторы. Ленты бывают различных цветов, длин и плотностей светодиодов, что позволяет подобрать оптимальный вариант для любого проекта.

Преимущества светодиодных лент:

- **Энергоэффективность:** Светодиоды потребляют значительно меньше энергии по сравнению с традиционными источниками света, такими как лампы накаливания и люминесцентные лампы.
- **Долговечность:** Светодиоды служат до 50 000 часов и более, что в несколько раз дольше, чем лампы накаливания.
- **Безопасность:** Светодиодные ленты не содержат вредных веществ, таких как ртуть, и не нагреваются до высоких температур.
- **Гибкость:** Ленты легко изгибаются и могут быть установлены в самых разных местах, включая криволинейные поверхности.
- **Разнообразие:** Светодиодные ленты доступны в различных цветах, яркостях, плотностях светодиодов и типах управления.

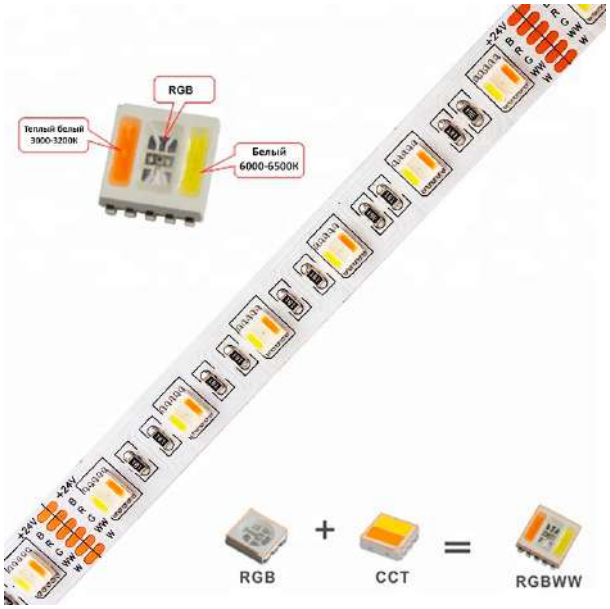
Типы светодиодных лент:

- **По типу светодиодов:**
  - **RGB:** Могут отображать миллионы цветов и оттенков.



- **RGBW:** Помимо RGB, включают белый светодиод для получения чистого белого света.

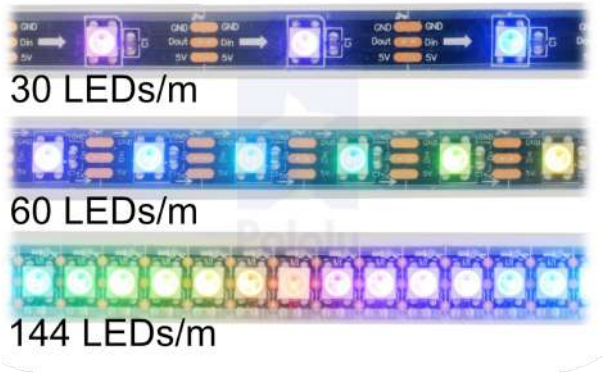
- **RGB+CCT:** Позволяют регулировать цветовую температуру белого света (от теплого до холодного).



- **Одноцветные:** Доступны в различных цветах, включая белый (с разными оттенками).

- **По типу управления:**
  - **Неадресные:** Все светодиоды на ленте светятся одинаково.

    - **Адресные (например, WS2812b):** Каждым светодиодом можно управлять по отдельности, что позволяет создавать динамичные световые эффекты.
- **По плотности светодиодов:**
    - **60 светодиодов на метр:** Подходит для общего освещения.
    - **120 светодиодов на метр:** Обеспечивает более яркое и равномерное освещение.
    - **144 светодиода на метр:** Используется для декоративного освещения и подсветки.



Применение светодиодных лент:

Светодиодные ленты широко используются в различных сферах:

- **Декоративное освещение:** Подсветка мебели, потолков, полов, зеркал, картин.
- **Уличное освещение:** Подсветка фасадов зданий, рекламных щитов, садовых дорожек.
- **Освещение интерьера:** Подсветка рабочих зон, зон отдыха, шкафов.
- **Индикация:** Светодиодные ленты могут использоваться для создания световых панелей, информационных табло.

Адресные светодиодные ленты

Адресные светодиодные ленты - это современная альтернатива традиционным светодиодным лентам, которая позволяет управлять каждым светодиодом по отдельности, что открывает безграничные возможности для создания динамичных и ярких световых эффектов.

Основные типы адресных светодиодных лент:

1. **WS2812b и его аналоги:**

    - **WS2812b:** Одна из самых популярных адресных светодиодных лент. Каждый светодиод имеет встроенный контроллер, что позволяет управлять цветом, яркостью и режимом свечения каждого светодиода по отдельности.
    - **WS2811:** Аналог WS2812b, но контроллеры расположены на концах ленты, а не в каждом светодиоде.

- **SK6812:** Аналог WS2812b с улучшенными характеристиками, такими как более высокая скорость передачи данных и поддержка RGBW.

2. **APA102 и его аналоги:**
   - **APA102:** Адресная светодиодная лента с двухпроводным протоколом передачи данных (SPI). Обеспечивает более высокую скорость обновления и более стабильную работу при большой длине ленты.
   - **SK9822:** Аналог APA102 с улучшенными характеристиками, такими как более высокая яркость и энергоэффективность.
3. **NeoPixel:**
   - **NeoPixel:** Торговая марка компании Adafruit для адресных светодиодных лент на базе WS2812b. Обеспечивает широкий выбор продукции и поддержку со стороны сообщества.

WS2812b

Адресная светодиодная лента WS2812b - это универсальный и мощный инструмент для создания ярких и динамичных световых эффектов. Благодаря простоте подключения и программирования, она подходит как для начинающих, так и для опытных пользователей.

Основные характеристики:

- **Тип светодиода:** RGB (красный, зеленый, синий)
- **Напряжение питания:** 5 В
- **Потребляемая мощность:** 0,3 Вт на светодиод
- **Расстояние между светодиодами:** 10 мм (в зависимости от модели)
- **Протокол передачи данных:** 1-проводной, синхронный

- **Частота обновления:** до 400 Гц

Применение:

Адресная светодиодная лента WS2812b широко используется для создания:

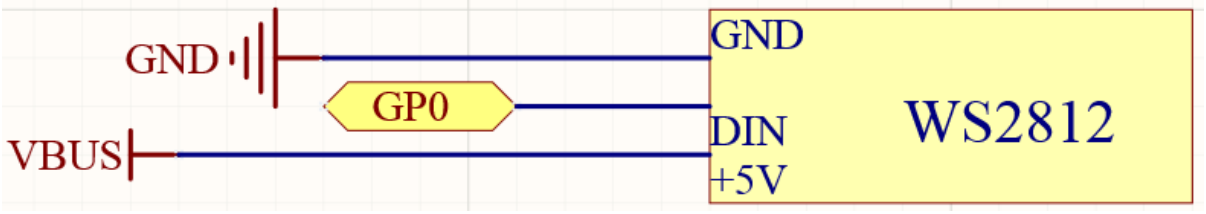
- **Динамических световых эффектов:** бегущие огни, плавная смена цветов, анимация.
- **Освещения:** декоративное освещение, подсветка мебели, потолков, полов.
- **Индикации:** световые панели, информационные табло.

Советы по выбору:

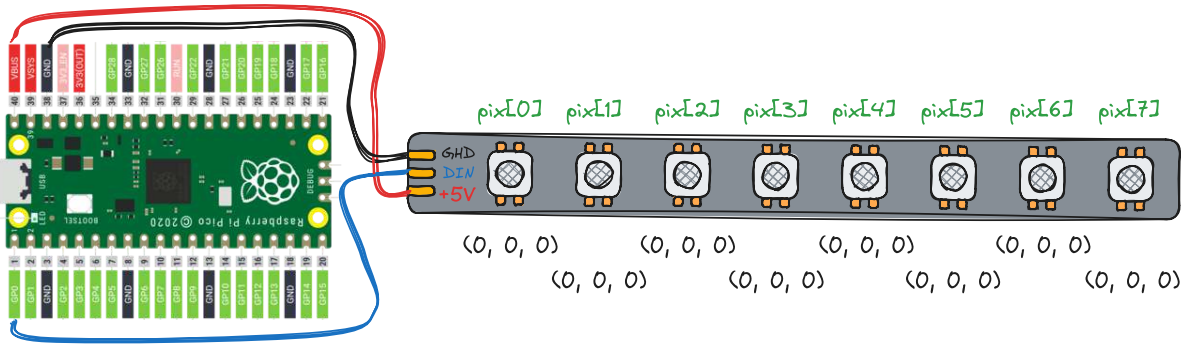
- **Длина ленты:** Определитесь с необходимым количеством светодиодов и длиной ленты.
- **Плотность светодиодов:** Чем выше плотность, тем более равномерное и яркое свечение.
- **Цвет светодиодов:** Выберите цвет светодиодов, который лучше всего подходит для вашего проекта.
- **Блок питания:** Убедитесь, что блок питания соответствует мощности ленты.
- **Контроллер:** Для управления лентой вам понадобится контроллер. Для программирования ленты WS2812b можно использовать различные платформы, такие как Arduino, Raspberry Pi, ESP8266 и другие. Существует множество библиотек, которые упрощают процесс программирования.

Подключение WS2812b

Адресная светодиодная лента WS2812b имеет 3 выхода для подключения:

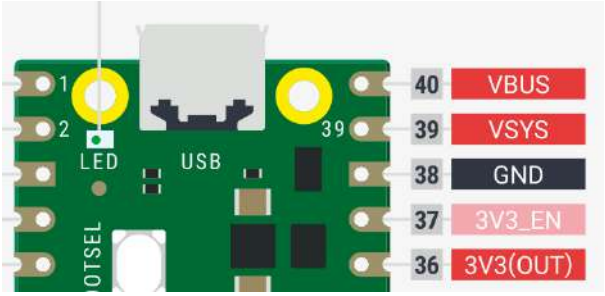


Короткую ленту (до 8 элементов) можно подключать в плате Raspberry Pi Pico напрямую:



Сигнальный провод ленты (DIN) можно подключить к любому цифровому GPIO пину Raspberry Pi Pico (например, GPIO 0).

Основным напряжением с которым работает плата Raspberry Pi Pico является 3.3V, в то время как адресной ленте необходимо напряжение в 5V. Получить его можно подключив ленту к выходу VBUS, напряжение на котором равняется входному напряжению, подающемуся на плату. (5V при условии, что лента питается по USB)



Пример подключения:

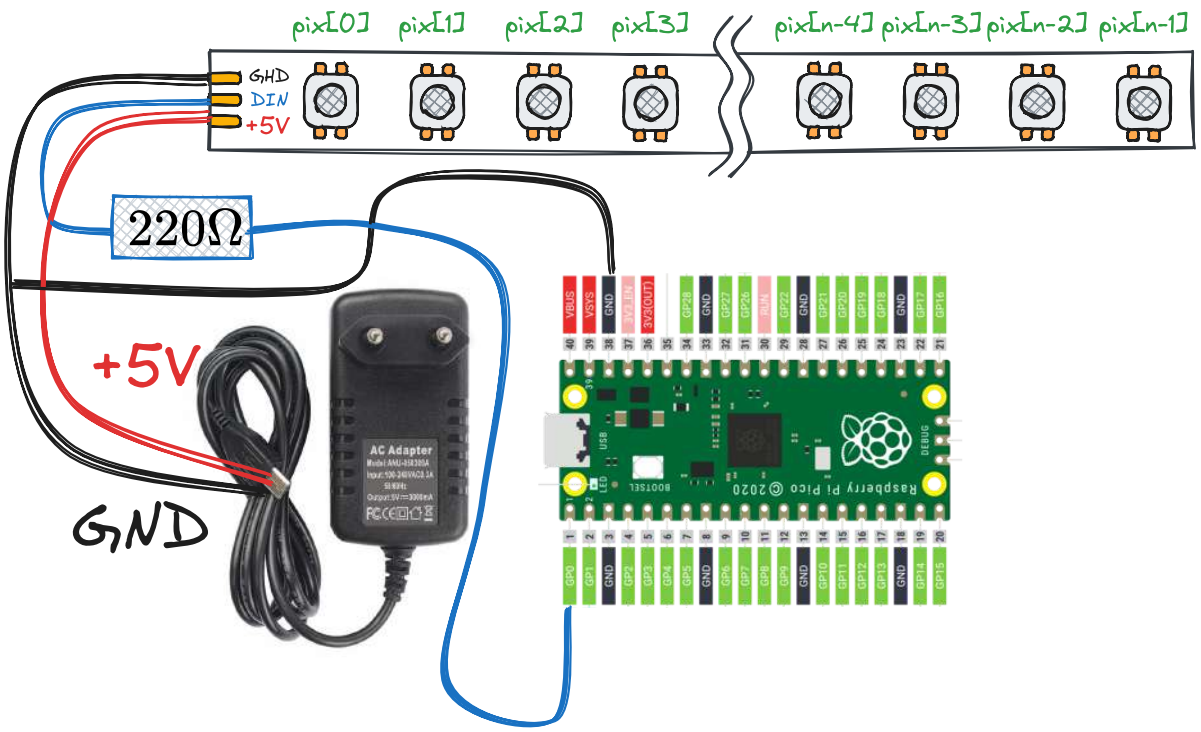
WS2812b	Raspberry Pi Pico
VCC	5V (VBUS)
GND	GND
DIN	GPIO 0

⚠ Подключать безопасно ленту напрямую к микрокотроллеру можно, если в ней не больше 8 элементов!

Это вызвано тем, что потребляемый одним элементом ток при максимальной яркости - 60 мА (белый цвет (255, 255, 255)), в то время как ток который можно получить с USB порта - 500 мА.

Подключение длинной ленты

При большем количестве элементов необходимо запитать ленту от внешнего источника питания на 5V.

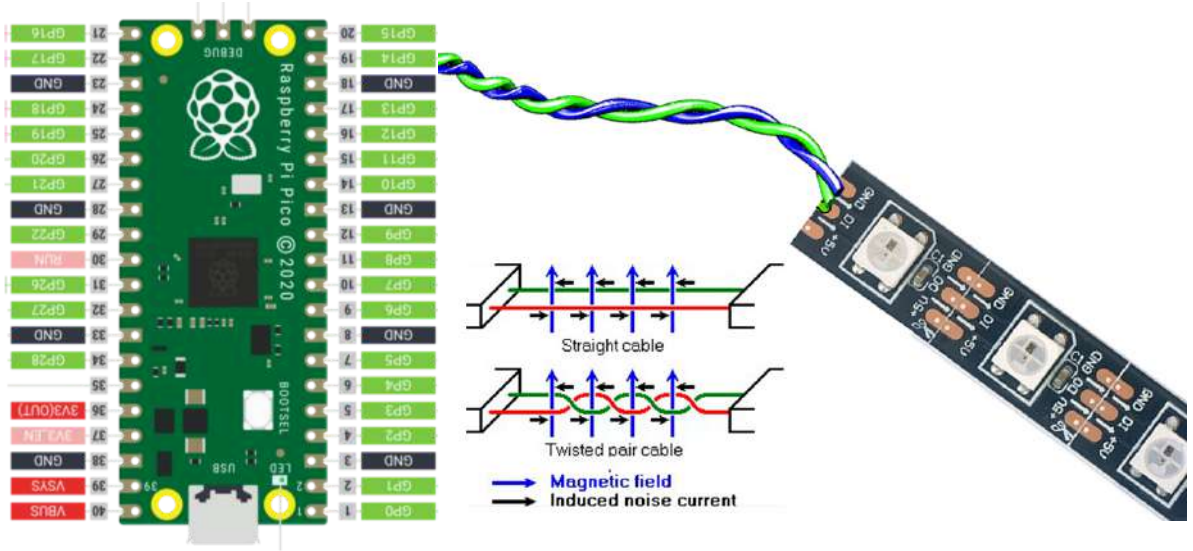


При этом необходимо обеспечить общее заземление микроконтроллера, источника питания и ленты.

⚠ Если есть вероятность того, что источник питания ленты будет выключен, а питание микроконтроллера сохраниться, то необходимо добавить в линию управления лентой дополнительный резистор!

Протокол связи между лентой и микроконтроллером достаточно скоростной - 800 кГц, и на него сильно влияют внешние наводки. Это может проявляться так: лента не работает до тех пор, пока не коснёшься рукой сигнального провода. Избежать этого можно экранировав земляной скруткой сигнальный провод.

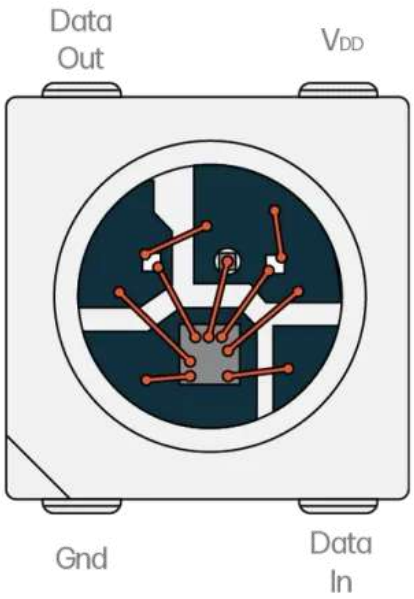
ⓘ Если между лентой и контроллером большое расстояние (длина сигнального провода больше 50 см), то сигнальный провод и землю нужно скрутить в косичку для защиты от наводок:



Управление светодиодной лентой

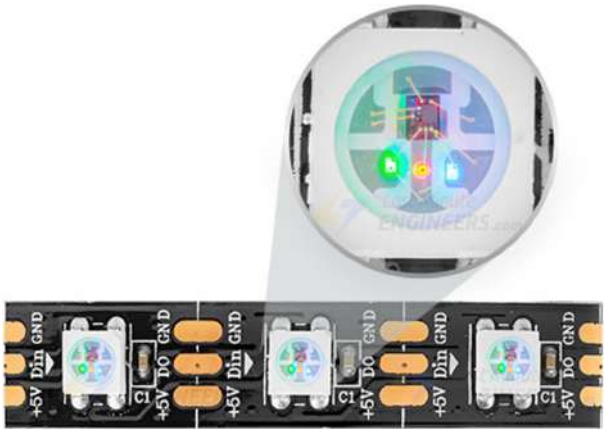
Общие сведения

Архитектура WS2812b



Каждый светодиод WS2812b содержит:

- **RGB-светодиод:** Три отдельных светодиода (красный, зеленый, синий), которые могут управляться независимо для создания миллионов цветов.
- **Цифровой контроллер:** Управляет яркостью и цветом каждого светодиода. Контроллер также содержит буфер для хранения данных о цвете следующего светодиода.



Протокол передачи данных

WS2812b использует 1-проводной синхронный протокол передачи данных. Это означает, что для управления лентой достаточно одного сигнального провода.

Формат данных:

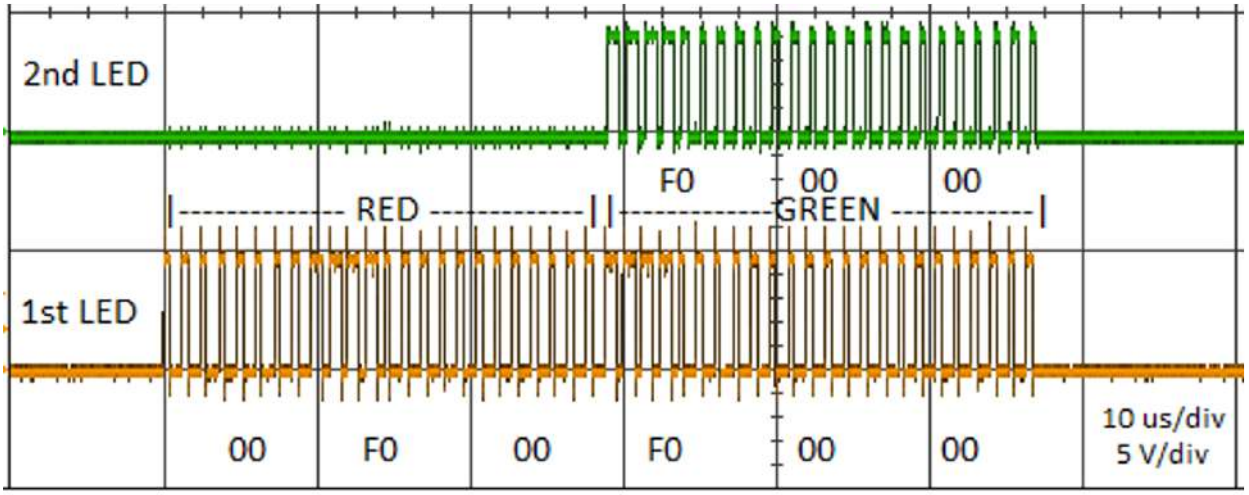
- **Один бит данных:** Представлен двумя уровнями напряжения:
  - **Логическая "1":** Высокий уровень напряжения (около 0,8 мкс) с последующим низким уровнем (около 0,45 мкс).
  - **Логический "0":** Высокий уровень напряжения (около 0,4 мкс) с последующим низким уровнем (около 0,85 мкс).
- **Цвет одного светодиода:** Кодируется 24 битами (8 бит для красного, 8 бит для зеленого, 8 бит для синего).

Последовательность передачи данных:

1. **Стартовый бит:** Высокий уровень напряжения в течение 50 мкс.
2. **Данные:** 24 бита для каждого светодиода.
3. **Сброс:** Низкий уровень напряжения в течение 50 мкс, чтобы сигнализировать о завершении передачи данных.

Принцип работы

1. **Передача данных:** Контроллер (например, Raspberry Pi Pico) передает данные о цвете каждого светодиода по очереди.
2. **Прием данных:** Первый светодиод в цепочке принимает данные и отображает свой цвет. Одновременно он передает оставшиеся данные следующему светодиоду.



3. **Буферизация:** Каждый светодиод хранит данные о своем цвете и передает оставшиеся данные следующему светодиоду.
4. **Отображение:** После получения всех данных светодиоды отображают свои цвета.

Библиотека `neopixel`

Библиотека `neopixel` - это мощный инструмент для управления адресными светодиодными лентами, такими как WS2812b, на микроконтроллерах, поддерживающих MicroPython, например, на Raspberry Pi Pico. Она позволяет

управлять светодиодами с помощью высокоуровневых команд, скрывая сложности низкоуровневого протокола передачи данных.

Основные возможности библиотеки `neopixel`:

- Управление цветом и яркостью каждого светодиода по отдельности.
- Поддержка различных типов адресных светодиодных лент, включая WS2812b, WS2811, SK6812 и другие.
- Простота использования благодаря высокоуровневым функциям.
- Библиотека `neopixel` обычно уже включена в стандартные пакеты MicroPython для большинства плат, таких как Raspberry Pi Pico.

Основные функции библиотеки `neopixel`

Импорт библиотеки

```
from machine import Pin
from neopixel import NeoPixel
```

Инициализация ленты

Для инициализации ленты необходимо указать пин, к которому она подключена, и количество светодиодов на ленте.

```
# Определяем количество светодиодов на ленте
NUM_OF_LEDS = 8

# Определяем пин, к которому подключена лента
PIX_PIN = 0
pin = Pin(PIX_PIN, Pin.OUT)

# Создаем объект NeoPixel
pix = NeoPixel(pin, NUM_OF_LEDS)
```

Установка цвета светодиода

Для установки цвета конкретного светодиода используется индекс светодиода (от 0 до `NUM_OF_LEDS - 1`) и кортеж из трех значений (R, G, B).

```
# Устанавливаем цвет первого светодиода на красный
pix[0] = (255, 0, 0)

# Устанавливаем цвет второго светодиода на зеленый
pix[1] = (0, 255, 0)
```

```
# Устанавливаем цвет третьего светодиода на синий
pix[2] = (0, 0, 255)
```

Запись данных на ленту

После установки цвета каждого светодиода необходимо вызвать метод `write()`, чтобы данные были переданы на ленту.

```
pix.write()
```

Установка цвета всех светодиодов

Для установки одного цвета на все светодиоды используется метод `fill()`

```
pix.fill((255, 0, 0))
```

Получение цвета со светодиода

Узнать текущий цвет светодиода можно обратившись по соответствующему индексу к объекту ленты:

```
pix[0] = (255, 0, 0)

color = pix[0]
print(color)  # (255, 0, 0)
```

Получить все текущие цвета с ленты можно в цикле:

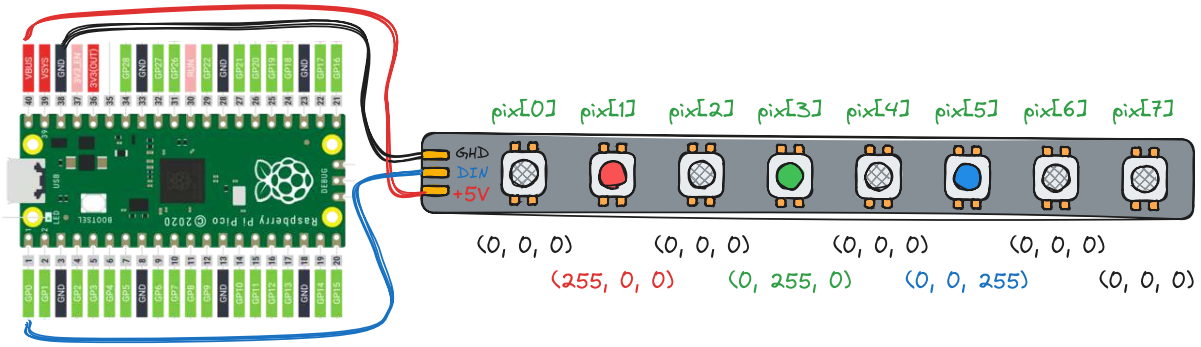
```
for color in pix:
    print(color)
```

или через распаковку:

```
print(*enumerate(pix), sep="\n")
```

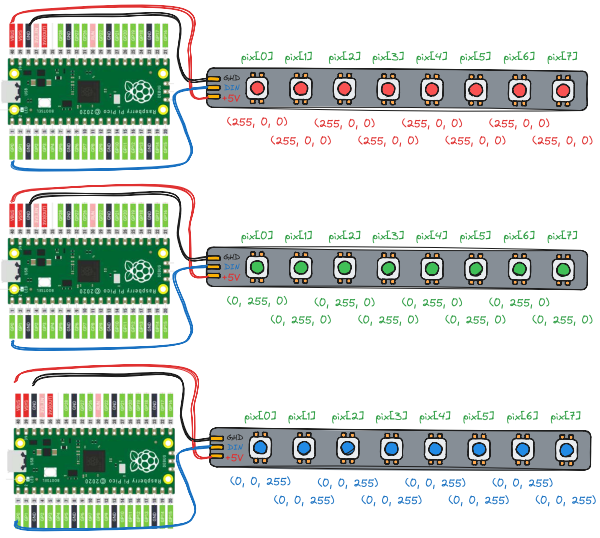
Примеры работы с WS2812b

Установка цвета на элементы ленты по индексам




```
import neopixel
from machine import Pin

PIX_PIN = 0
NUM_OF_LEDS = 8
pix = neopixel.NeoPixel(Pin(PIX_PIN), NUM_OF_LEDS)

print(*enumerate(pix), sep="\t")

pix[1] = (255, 0, 0)
pix[3] = (0, 255, 0)
pix[5] = (0, 0, 255)

print(*enumerate(pix), sep="\t")

pix.write()
```

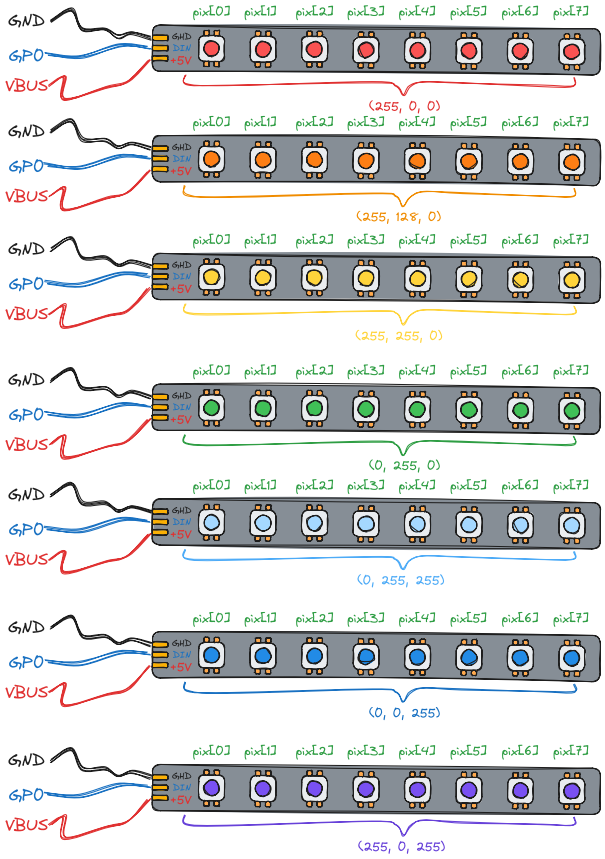
Заполнение всей ленты одним цветом



```
import neopixel
from machine import Pin
from time import sleep

PIX_PIN = 0
NUM_OF_LEDS = 8
```

```
SLEEP_TIME = 2

pix = neopixel.NeoPixel(Pin(PIX_PIN), NUM_OF_LEDS)

while True:
    pix.fill([255, 0, 0])
    pix.write()
    print(*enumerate(pix), sep="\t", end="\n\n")
    sleep(SLEEP_TIME)

    pix.fill([0, 255, 0])
    pix.write()
    print(*enumerate(pix), sep="\t", end="\n\n")
    sleep(SLEEP_TIME)

    pix.fill([0, 0, 255])
    pix.write()
    print(*enumerate(pix), sep="\t", end="\n\n")
    sleep(SLEEP_TIME)
```

Циркулирующая по всей ленте радуга




```
import neopixel
from machine import Pin
from time import sleep
```

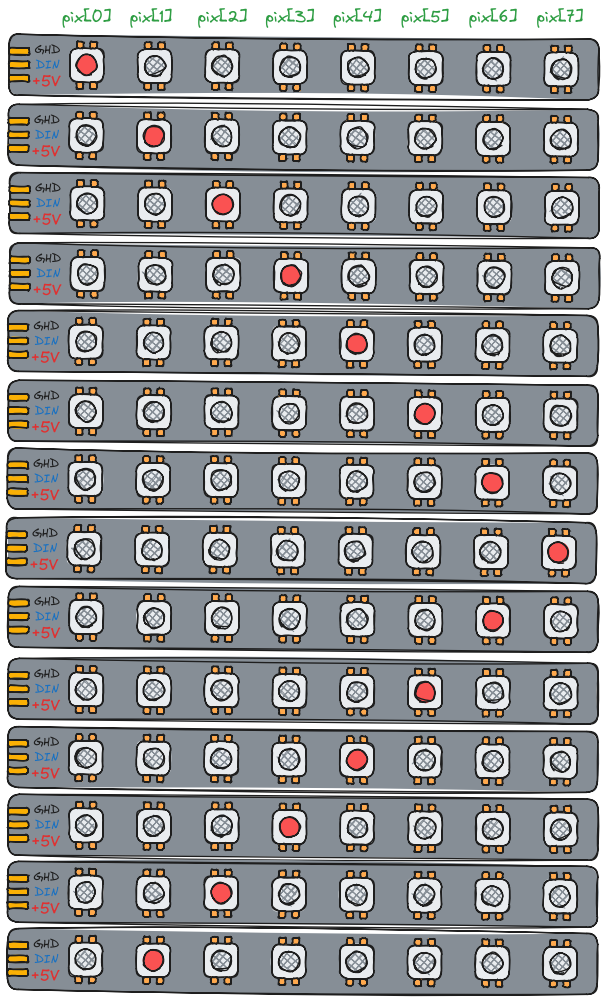
```
PIX_PIN = 0
NUM_OF_LEDS = 8

SLEEP_TIME = 0.025

pix = neopixel.NeoPixel(Pin(PIX_PIN), NUM_OF_LEDS)

def rgb_wheel(degree):
    def color_pattern(degree):
        degree %= 360
        if 0 <= degree < 60:
            value = degree / 60
        elif 180 <= degree < 240:
            value = 1 - (degree - 180) / 60
        else:
            value = 1 if 60 <= degree < 180 else 0
        return int(value * 255)
    red = color_pattern(degree - 240)
    green = color_pattern(degree)
    blue = color_pattern(degree - 120)
    return red, green, blue

while True:
    for deg in range(360):
        color = rgb_wheel(deg)
        pix.fill(color)
        pix.write()
        sleep(SLEEP_TIME)
```

Последовательное заполнение цветом ленты


```
import neopixel
from machine import Pin
from time import sleep

PIX_PIN = 0
NUM_OF_LEDS = 8
pix = neopixel.NeoPixel(Pin(PIX_PIN), NUM_OF_LEDS)

SLEEP_TIME = 1

while True:
    for i in range(0, NUM_OF_LEDS):
        pix[i] = [255, 0, 0]
        pix.write()
        print(*enumerate(pix), sep="\t")
        sleep(SLEEP_TIME)
    for i in range(NUM_OF_LEDS - 1, -1, -1):
        pix[i] = [0, 0, 0]
        pix.write()
```

```
        print(*enumerate(pix), sep="\t")
        sleep(SLEEP_TIME)
```

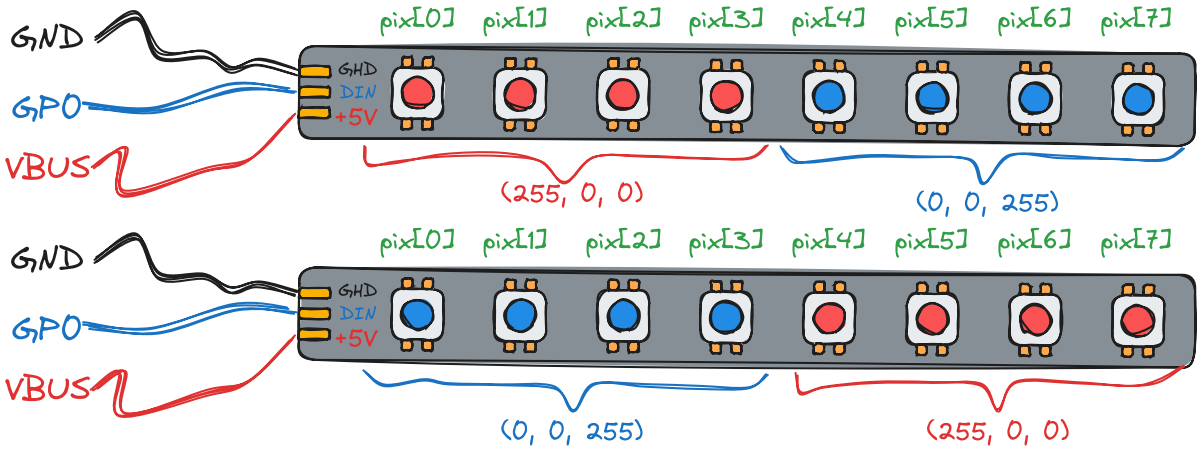
"Бегущий" огонек




```
import neopixel
from machine import Pin
from time import sleep

PIX_PIN = 0
NUM_OF_LEDS = 8
pix = neopixel.NeoPixel(Pin(PIX_PIN), NUM_OF_LEDS)

SLEEP_TIME = 0.75

while True:
    for i in range(0, NUM_OF_LEDS):
        pix.fill([0, 0, 0])
        pix[i] = [255, 255, 255]
        pix.write()
        print(*enumerate(pix), sep="\t")
        sleep(SLEEP_TIME)
    for i in range(NUM_OF_LEDS - 2, 0, -1):
        pix.fill([0, 0, 0])
```

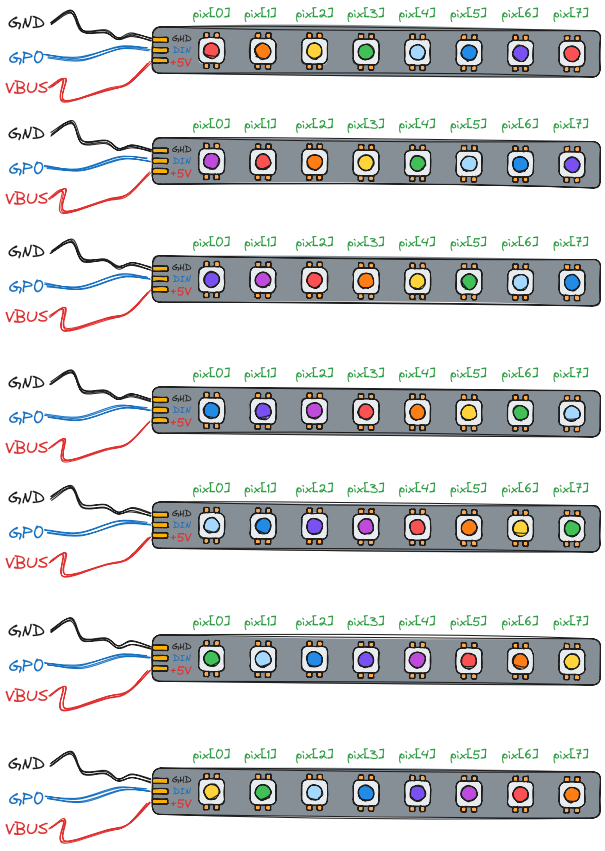
```
        pix[i] = [255, 255, 255]
        pix.write()
        print(*enumerate(pix), sep="\t")
        sleep(SLEEP_TIME)
```

Сирена




```
import neopixel
from machine import Pin
from time import sleep

PIX_PIN = 0
NUM_OF_LEDS = 8
pix = neopixel.NeoPixel(Pin(PIX_PIN), NUM_OF_LEDS)

SLEEP_TIME = 1

RED = (255, 0, 0)
BLUE = (0, 0, 255)

state = 0
while True:
    for i in range(round(NUM_OF_LEDS / 2)):
        pix[i] = RED if state == 0 else BLUE
    for i in range(round(NUM_OF_LEDS / 2), NUM_OF_LEDS):
        pix[i] = BLUE if state == 0 else RED
    state = (state + 1) % 2
    print(*enumerate(pix), sep="  ")
    pix.write()
    sleep(SLEEP_TIME)
```

"Бегущая" по ленте радуга




```
import neopixel
from machine import Pin
from time import sleep

PIX_PIN = 0
NUM_OF_LEDS = 8
pix = neopixel.NeoPixel(Pin(PIX_PIN), NUM_OF_LEDS)
BRIGHTNESS = 1 if NUM_OF_LEDS == 8 else 0.1

SLEEP_TIME = 0.0001
COLOR_SHIFT = 30
DEG_SHIFT = 1

def rgb_wheel(degree):
    def color_pattern(degree):
        degree %= 360
        if 0 <= degree < 60:
            value = degree / 60
        elif 180 <= degree < 240:
            value = 1 - (degree - 180) / 60
        else:
            value = 1 if 60 <= degree < 180 else 0
        return int(value * 255)
    red = color_pattern(degree - 240)
    green = color_pattern(degree)
    blue = color_pattern(degree - 120)
    return red, green, blue
```

```
def set_brightness(color, brightness=BRIGHTNESS):
    return [int(c * brightness) for c in color]

deg = 0
while True:
    for led in range(NUM_OF_LEDS):
        color = rgb_wheel((deg + led * COLOR_SHIFT) % 360)
        pix[led] = set_brightness(color)
    deg = (deg + DEG_SHIFT) % 360
    pix.write()
    sleep(SLEEP_TIME)
```

Рефакторинг кода в функции

Преамбула файла

```
import neopixel
from machine import Pin
from time import sleep, ticks_ms

PIX_PIN = 0
NUM_OF_LEDS = 8
pix = neopixel.NeoPixel(Pin(PIX_PIN), NUM_OF_LEDS)

BRIGHTNESS = 1 if NUM_OF_LEDS == 8 else 0.1

def rgb_wheel(degree):
    def color_pattern(degree):
        degree %= 360
        if 0 <= degree < 60:
            value = degree / 60
        elif 180 <= degree < 240:
            value = 1 - (degree - 180) / 60
        else:
            value = 1 if 60 <= degree < 180 else 0
        return int(value * 255)
    red = color_pattern(degree - 240)
    green = color_pattern(degree)
    blue = color_pattern(degree - 120)
    return red, green, blue
```

Функция для выключения ленты

```
def switch_off_the_leds(pix):
    pix.fill([0, 0, 0])
    pix.write()
```

Функция для настройки яркости цвета

```
def set_brightness(color, brightness):
    return [int(c * brightness) for c in color]
```

Функция для проверки времени работы других функций

```
is_time_over = lambda start_time, time_of_work: (ticks_ms() - start_time)
/ 1000 >= time_of_work
```

Или (в обычной записи):

```
def is_time_over(start_time, time_of_work):
    return (ticks_ms() - start_time) / 1000 >= time_of_work
```

Функция для циркулирующей по всей ленте радуги

```
def vertical_rainbow(pix, brightness=BRIGHTNESS, time_of_work=10,
sleep_time=0.05):
    time_0 = ticks_ms()
    while True:
        for i in range(360):
            color = set_brightness(rgb_wheel(i), brightness)
            pix.fill(color)
            pix.write()
            sleep(sleep_time)
            if is_time_over(time_0, time_of_work):
                break
        if is_time_over(time_0, time_of_work):
            break
    switch_off_the_leds(pix)
    pix.write()
```

Функция для "бегущего" огонька

```
def running_light(pix, base_color=(0, 0, 255),
                  running_led_color=(255, 0, 0),
                  brightness=BRIGHTNESS,
                  time_of_work=10, sleep_time=0.5):
    time_0 = ticks_ms()
    num_of_leds = len(pix)
    base_color, running_led_color = [set_brightness(c, brightness) for c
in (base_color, running_led_color)]
    while True:
        for i in range(num_of_leds):
```

```
            pix.fill(base_color)
            pix[i] = running_led_color
            pix.write()
            if is_time_over(time_0, time_of_work):
                break
            sleep(sleep_time)
        for i in range(num_of_leds - 2, 0, -1):
            pix.fill(base_color)
            pix[i] = running_led_color
            pix.write()
            if is_time_over(time_0, time_of_work):
                break
            sleep(sleep_time)
        if is_time_over(time_0, time_of_work):
            break
    switch_off_the_leds(pix)
    pix.write()
```

Функция для сирены

```
def flasher(pix, colors=((255, 0, 0), (0, 0, 255)),
            brightness=BRIGHTNESS, time_of_work=10, sleep_time=0.5):
    time_0 = ticks_ms()
    num_of_leds = len(pix)
    color_1, color_2 = [set_brightness(c, brightness) for c in colors]
    state = 0
    while True:
        for i in range(round(num_of_leds / 2)):
            pix[i] = color_1 if state == 0 else color_2
        for i in range(round(num_of_leds / 2), num_of_leds):
            pix[i] = color_2 if state == 0 else color_1
        state = (state + 1) % 2
        pix.write()
        if is_time_over(time_0, time_of_work):
            break
        sleep(sleep_time)
    switch_off_the_leds(pix)
    pix.write()
```

Функция для "бегущей" по ленте радуги

```
def horizontal_rainbow(pix, brightness=BRIGHTNESS,
                       rainbow_count=1, rainbow_speed=1,
                       time_of_work=10, sleep_time=0.01):
    time_0 = ticks_ms()
    num_of_leds = len(pix)
    deg = 0
```

```
    color_shift = round(360 / num_of_leds * rainbow_count)
    while True:
        for led in range(num_of_leds):
            color = rgb_wheel((deg + led * color_shift) % 360)
            pix[led] = set_brightness(color, brightness)
        deg = (deg + rainbow_speed) % 360
        pix.write()
        if is_time_over(time_0, time_of_work):
            break
        sleep(sleep_time)
    switch_off_the_leds(pix)
    pix.write()
```

Последовательный вызов функций

```
while True:
    for func in horizontal_rainbow, flasher, running_light,
vertical_rainbow:
        func(pix)
        sleep(1)
```