AKADEMIA GÓRNICZO-HUTNICZA

Dokumentacja do projektu

Biblioteka do obsługi ledów cyfrowych ws28**

z przedmiotu

Języki Programowania Obiektowego

Elektronika i Telekomunikacja 3 rok

Wiktor Kidoń

Piątek, 11:30

prowadzący: Jakub Zimnol

09.01.2025

1. Wstęp

Biblioteka WS2812_Simple została zaprojektowana do sterowania paskami LED opartymi na diodach WS2812 za pomocą mikrokontrolera ESP32. Obsługuje ona różne animacje świetlne, takie jak gradienty, tęcze, pulsowanie, a także proste ustawienia kolorów.

2. Główna Klasa i metody

1. Konstruktor

   WS2812_Simple(int ledCount, int ledPin, rmt_channel_t rmtChannel);

   Tworzy obiekt do sterowania paskiem LED. Inicjalizuje pamięć dla danych o diodach oraz ustawia parametry, takie jak liczba diod (ledCount), pin komunikacyjny (ledPin) i kanał RMT (rmtChannel).

2. Inicjalizacja

   void begin();

   Przygotowuje kanał RMT do pracy z diodami LED, konfigurując go i instalując odpowiedni sterownik.

3. Ustawienie jasności

   void setBrightness(uint8_t brightness);

   Ustawia globalną jasność diod w zakresie od 0 (ciemno) do 255 (pełna jasność).

4. Ustawienie koloru diody

   void setPixelColor(int index, uint8_t red, uint8_t green, uint8_t blue);

   void setPixelColor(int index, uint32_t color);

   Ustawia kolor konkretnej diody LED:

   - Pierwsza wersja używa składowych RGB.

   - Druga wersja pozwala ustawić kolor w formacie 32-bitowym (np. 0xRRGGBB).

5. Odczyt koloru diody

   uint32_t getPixelColor(int index);

   Zwraca kolor wybranej diody w formacie 32-bitowym (np. 0xRRGGBB).

6. Wypełnianie kolorem

   void fill(uint8_t red, uint8_t green, uint8_t blue);

   Ustawia cały pasek LED na jeden kolor.

7. Gradient kolorów

void fillGradient(uint8_t startRed, uint8_t startGreen, uint8_t startBlue,
uint8_t endRed, uint8_t endGreen, uint8_t endBlue);

Tworzy płynny gradient od koloru początkowego do końcowego na całym pasku LED.

8. Przesuwający się gradient

void gradientMove(uint8_t startRed, uint8_t startGreen, uint8_t startBlue,
uint8_t endRed, uint8_t endGreen, uint8_t endBlue);

Przesuwa gradient wzdłuż paska LED, tworząc efekt ruchu.

9. Efekt tęczy

void rainbow();

void rainbowMove();

- rainbow: Generuje statyczny efekt tęczy.

- rainbowMove: Przesuwa tęczę wzdłuż paska.

10. Efekt "ścigania" koloru

void chase(uint8_t red, uint8_t green, uint8_t blue, int delayMs);

Zapala diody jedna po drugiej w wybranym kolorze.

11. Mruganie diodami

void blink(uint8_t red, uint8_t green, uint8_t blue, int delayMs);

Wszystkie diody włączają się i wyłączają w określonym kolorze.

12. Efekt pulsowania

void breathing(uint8_t red, uint8_t green, uint8_t blue, int durationMs);

Jasność wybranego koloru płynnie wzrasta i maleje, tworząc efekt "oddychania".

13. Odwrócenie kolejności diod

void setReverseOrder(bool reverse);

Odwraca kolejność diod na pasku LED.

14. Zmiana kolejności kolorów

void setColorOrder(ColorOrder order);

Zmienia kolejność składowych kolorów między RGB a GRB.

15. Wyświetlanie danych

void show();

Wysyła zbuforowane dane do paska LED, aby wyświetlić aktualny stan diod.

16. Konwersja kolorów na sygnał RMT

void rgbToRmt(uint8_t red, uint8_t green, uint8_t blue, rmt_item32_t* data);

Konwertuje wartości RGB na dane zgodne z protokołem WS2812, umożliwiając przesyłanie ich do diod.

3. Przykład użycia

Wokwi Simulator | main.cpp | WS2812.h | WS2812.cpp

src > main.cpp > ...

```cpp
#include "WS2812.h"
#include <Arduino.h>

#define LED_PIN 18
#define LED_COUNT 16

WS2812_Simple leds(LED_COUNT, LED_PIN, RMT_CHANNEL_0);

void setup() {
    leds.begin();
    leds.setBrightness(200);
    leds.fill(255, 0, 0); // Wszystkie diody na czerwono
    leds.show();
}

void loop() {
    leds.rainbowMove(); // Efekt tęczy w ruchu
    leds.show();
    delay(50);
}
```

4. Schemat połączenia