МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Інститут інформатики та радіоелектроніки,
Факультет радіоелектроніки та телекомунікацій
(повне найменування інституту, факультету)

Кафедра інформаційних технологій електронних засобів
(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)
бакалавр
(ступінь вищої освіти)

на тему РОЗРОБКА ПРОГРАМИ ДЛЯ ВІДДАЛЕНОГО КЕРУВАННЯ СВІТЛОВИМИ ЕФЕКТАМИ НА ОСНОВІ КОНТРОЛЛЕРУ ESP32
DESIGNING OF PROGRAM FOR LIGHT EFFECTS REMOTE CONTROL BASED ON ESP32 CONTROLLER

Виконав: студент 4 курсу, групи РТ-517
Спеціальності 172 Телекомунікації та радіотехніка
(код і найменування спеціальності)
Освітня програма (спеціалізація)
Інтелектуальні технології мікросистемної радіоелектронної техніки
Резанова А.М.
(прізвище та ініціали)
Керівник Фарафонов О.Ю.
(прізвище та ініціали)
Рецензент Зеленьова І.Я.
(прізвище та ініціали)

2021

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
(повне найменування закладу вищої освіти)

Інститут, факультет ФРЕТ
Кафедра інформаційних технологій електронних засобів
Ступінь вищої освіти бакалавр
Спеціальність 172 Телекомунікації та радіотехніка
(код і найменування)
Освітня програма (спеціалізація) Інтелектуальні технології мікросистемної радіоелектронної техніки
(назва освітньої програми (спеціалізації)

ЗАТВЕРДЖУЮ

В.о. завідувача кафедри ІТЕЗ, к.т.н, доц.
_____Є.В. Огренич
“ ___ ” травня 20 21року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА

Резанової Аміни Магомеднасирівни
(прізвище, ім’я, по батькові)

1. Тема проєкту (роботи) Розробка програми для віддаленого керування світловими ефектами на основі контроллеру esp32. Designing of program for light effects remote control based on esp32 controller
керівник проєкту (роботи) Фарафонов О. Ю., к.т.н., доцент ,
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)
затверджені наказом закладу вищої освіти від “ 26” квітня 2021 року № 163
2. Строк подання студентом проєкту (роботи) 1 червня 2021 року
3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити Реферат. Огляд. 1.Робота з адресною світлодіодною стрічкою і бібліотекою FastLED. 2. Веб-сервера і робота з ними. 3. Створення веб-сторінок.Висновки. Перелік джерел посилання.
5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)
33 рисунка;слайди презентації
6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-3 Основна частина	Фарафонов О. Ю., доцент	05.04	
Нормоконтроль	Поспеєва І.Є., ст.викладач	28.05	

7. Дата видачі завдання “ 26 ” квітня 2021 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області	2-3 тижні	Огляд
3	Розробка архітектури програми	4 тиждень	Розділ 1-2
4	Розробка програми	5-6 тижні	Розділ 3
5	Тестування та експериментальне дослідження програми	7 тиждень	
6	Оформлення пояснювальної записки та документів до неї. Нормоконтроль та рецензування.	8 тиждень	Реферат Висновки
7	Захист роботи	9 тиждень	

Студент _____ Резанова А. М.
(підпис) (прізвище та ініціали)

Керівник проєкту (роботи) _____ Фарафонов О.Ю.
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка містить: 83 с., 3 ч., 33 рис., 57 джерел.

AJAX, CSS, ESP32, FASTLED, HSV, HTML, HTTP, JAVASCRIPT, JQUERY, RGB, WI-FI, СЕРВЕР.

У дипломній роботі було проведено аналіз існуючих розробок і конструкцій для управління адресними світлодіодними стрічками. Розглядалися різні методи управління:

- за допомогою інфрачервоного пульта і контролера з ІФ-приймачем;
- контроль додатком через Bluetooth;
- управління голосом;
- включення стрічки за допомогою датчика руху;
- зміни режиму за допомогою кнопок;
- перепрошивка плати.

Були проведені порівняльні характеристики актуальних методов. В ході аналізу були виявлені недоліки у кожного метода.Також були розглянуті приклади програм деяких варіантів конструкцій.Після цього була проаналізована актуальність мого методу віддаленого управління освітленням і його переваги перед іншими методами.

При розробці були досліджені різні бібліотеки для керування стрічкою.У порівнянні наведені дві найпопулярніші бібліотеки FastLED і Adafruit NeoPixel.За всіма параметрами бібліотека FastLED більше підходила для мого проекту, так як мала ряд переваг перед іншими бібліотеками.Далі були досліджені сумісні з цією бібліотекою пристрої, всі її функції, приклади їх використання і були розроблені режими світлових ефектів стрічки.

Були описані можливості бібліотеки Wi-Fi.h.

Також були досліджені різні види серверів (синхронного і асинхронного). В результаті аналізу цих серверів був обраний асинхронний сервер для роботи моєї програми. В даному випадку асинхронний сервер має ряд переваг:

- сервер самостійно визначає коли закрити з'єднання і звільнити ресурси;
- будь-який час між відповідями витрачається на запуск призначеного для користувача циклу і обробку інших мережевих пакетів;
- відповіді з використанням HTTP протоколу здійснює сама бібліотека [1];
- при отриманні запиту з боку клієнта сервер в асинхронному режимі буде відправляти відповідь користувачеві, що дозволяє одночасно обробляти запит і реалізовувати функціонал стрічки.

На додаток до цього, в роботі були розглянуті можливості ESP32 при роботі з мережею Wi-Fi.В ході досліджень на цю тему тему було прийнято рішення використовувати плату в режимі станції, так як режим точки доступу працює тільки в синхронному режимі, що призвело б до некоректної роботі програми.

ЗМІСТ

ОГЛЯД

На даний момент на ринку набирають популярність пристрої для освітлення і підсвічування. Існують різні електронні пристрої з адресною світлодіодною стрічкою. Найпоширеніші проекти зроблені на базі Arduino або китайських мікроконтролерів. Живлення може йти від мережі або від акумуляторів. В більшості випадків стрічка управляється через інфрачервоний пульт і контролер з ІФ-приймачем.

Для найпростішого управління стрічкою можна придбати китайський контролер і здійснювати контроль додатком через Bluetooth.

Ось приклад управління стрічкою за допомогою Bluetooth(рис. 0.1):

```
const int redPin = 11;
const int greenPin = 10;
const int bluePin = 9;

void setup() {
  Serial.begin(9600);
  pinMode(redPin, OUTPUT);
  pinMode(greenPin, OUTPUT);
  pinMode(bluePin, OUTPUT);

}

void loop() {
  while (Serial.available() > 0) {
    int red = Serial.parseInt();
    int green = Serial.parseInt();
    int blue = Serial.parseInt();

    if (Serial.read() == '\n') {
      red   =   constrain(red, 0, 255);
      green =   constrain(green, 0, 255);
      blue  =   constrain(blue, 0, 255);

      analogWrite(redPin,   red);
      analogWrite(greenPin, green);
      analogWrite(bluePin,  blue);
    }
  }
 }
```

Рисунок 0.1 – Приклад управління стрічкою за допомогою Bluetooth

У setup () починається послідовне з'єднання з частотою 9600 бод і встановлюються всі порти стрічки на OUTPUT.

У циклі loop (), якщо Serial monitor отримує щось, то він (Serial monitor) аналізує отримані дані як Integer (цілочисельний тип даних).

Якщо він отримує символ нового рядка ('\ n'), він спочатку обмежує значення діапазоном 0-255 через діапазон PWM (ШІМ, англ. Pulse-width modulation (PWM)), а потім робить зміни у цифрових портах за допомогою методу analogWrite()[4]. Далі за допомогою додатка Smart Bluetooth створюється призначений для користувача інтерфейс для спілкування з модулем Bluetooth або платою та для управління трьома колірними каналами на стрічці.

Також у таких великих корпорацій як Xiaomi є продукція з адресною стрічкою. Управління у неї здійснюється голосом.

Існує безліч проектів на основі Arduino з можливістю включення стрічки за допомогою датчика руху або зміни режиму за допомогою кнопок або перепрошивки плати.

Приклад для управління адресною світлодіодною стрічкою[5]:

```
#include <Adafruit_NeoPixel.h> // підключаємо бібліотеку
#define PIN 10 // вказуємо пін для підключення стрічки
#define NUMPIXELS 3 // вказуємо кількість світлодіодів в стрічці
Adafruit_NeoPixel strip (NUMPIXELS, PIN, NEO_GRB + NEO_KHZ800);
// створюємо об'єкт strip з потрібними характеристиками
void setup () {
  strip.begin (); // ініціалізуємо стрічку
  strip.setBrightness (50); // вказуємо яскравість світлодіодів (максимум 255)
}
void loop () {
 for (int i = -1; i <NUMPIXELS; i ++) {  // по черзі вмикаємо червоний колір
```

```
    strip.setPixelColor (i, strip.Color (255, 0, 0));
    strip.show ();
    delay (100);
  }
  for (int i = -1; i <NUMPIXELS; i ++) {   // по черзі вмикаємо зелений колір
    strip.setPixelColor (i, strip.Color (0, 255, 0));
    strip.show ();
    delay (100);
  }
  for (int i = -1; i <NUMPIXELS; i ++) {    // по черзі вмикаємо синій колір
    strip.setPixelColor (i, strip.Color (0, 0, 255));
    strip.show ();
    delay (100);
  }}
```

Такі варіанти вимагають додаткового обслуговування:

- кнопка може зношуватися і вимагати заміни;

- для перепрошивки потрібен фізичний доступ, це створює додаткові труднощі, якщо стрічка перебуватиме в важкодоступних місцях.

Також управління голосом не підходить для використання в великих скупченнях людей, на вулицях або в шумних приміщеннях. Кнопки не дають такої кількості різноманітних режимів, які є у моєму пристрої. При управлінні стрічкою за допомогою інфрачервоного пульта і контролера з ІФ-приймачем є також ряд незручностей. Наприклад, потрібно чітко потрапити променем в ІФ-приймач і щоб в приміщенні ніщо не переривало промінь, відповідно, у великому скупченні людей це буде доставляти труднощі і буде працювати тільки при певних умовах, де з будь-якого місця є прямий шлях для променя до ІФ-приймача.

Проаналізувавши всі методи управління, мною було прийнято рішення зробити управління за допомогою бездротових мереж. У мережі Wi-Fi виявився ряд переваг перед іншими бездротовими мережами, саме тому я обрала її.Дальність мережі Wi-Fi набагато більше, наприклад, ніж у Bluetooth.Також у мережі Wi-Fi вище швидкість. Остання версія Wi-Fi забезпечує максимальну швидкість передачі даних 250 Мбіт / с, в той час як Bluetooth 4.0 забезпечує швидкість передачі даних 25 Мбіт / с.[2].

Різниця також в радіусі дії цих стандартів. Так, наприклад радіус дії Wi-Fi дорівнює 300 метрам, в той час час як у Bluetooth всього 30 метрів [3].

На даний момент аналогів моєму пристрою не знайдено. Вище перераховані можливі способи управління стрічкою, але управління через сайт за допомогою мережі Wi-Fi з використанням асинхронного сервера не було знайдено мною.

У моєму проекті буде сайт, з якого будуть відправлятися дані про режим і яскравість стрічки на сервер Esp32, також там будуть кілька самостійних web-сторінок. Ці web-сторінки поділяють види режимів .У кожної є різні режими, які діляться на такі види:

- вбудовані режими;
- змінні режими;
- стробоскоп;
- вкладка для зміни яскравості.

В результаті вийде Wi-Fi контролер для адресної світлодіодної стрічки на базі плати Esp32 з використанням асинхронного сервера.

Код поділяється на окремі частини для управління стрічкою, для створення сервера і для створення веб-сторінок.

У першому розділі описується тема роботи з бібліотекою «FastLed» для управління адресною світлодіодною стрічкою. Розглянуто функції цієї бібліотеки і структура мого коду для управління стрічкою.

У другому розділі розповідається про створення сервера і про види серверів. Також описуються можливості Esp32 при роботі з сервером. А також про HTTP-запити і структуру отримання відповіді і відправлення запиту на сервер.

Логіка серверної частини моєї програми така, що створюється точка доступу в локальній мережі. Підключившись до маршрутизатора, плата отримує IP-адресу, яка не може помінятися до момента перезавантаження сервера. Сервер працює незалежно від нас, в асинхронному режимі, що дозволяє працювати одночасно логіці стрічки і сервера. Відповіді з використанням HTTP протоколу здійснює сама бібліотека, ми можемо тільки поміняти заголовок Content-Type і саме тіло відповіді. У setup() ми прописуємо роути, можливі для обробки нашим сервером і створюємо для них обробник події. При отриманні запиту з боку клієнта сервер в асинхронному режимі буде відправляти відповідь користувачеві, що дозволяє одночасно обробляти запит і реалізовувати функціонал стрічки.

Третій розділ про створення веб-сайту. У ньому будуть використовуватися мови HTML, CSS, JavaScript. Сайт можна буде відкрити з телефону або з ПК. Також буде розказано про структуру мого коду для створення веб-сторінок. Будуть розкриті теми про саму мову HTML, про мову JavaScript і фреймворку для нього jQuery.

Перевага використання мого методу керування, наприклад, у важкодоступних місцях, так як Wi-Fi має досить великий радіус дії. Якщо стрічка знаходиться високо під стелею або використовується для освітлення із

зовнішнього боку будівлі (наприклад, під дахом). Крім того, великою перевагою є зручне управління сайтом з телефону або комп'ютера.

Також буде зручно управляти стрічкою по мережі Wi-Fi на високих вуличних арках для прикраси і освітлення вулиць. У клубах, де немає можливості керувати голосом або потрібно керувати освітленням здалеку і немає можливості близько підійти. В багатокімнатних або багатоповерхневих приміщеннях буде зручно керувати освітленням в залі або кімнаті з іншої точки приміщення.

1 РОБОТА З АДРЕСНОЮ СВІТЛОДІОДНОЮ СТРІЧКОЮ І БІБЛІОТЕКОЮ FASTLED

1.1 Переваги бібліотеки FastLED перед іншими бібліотеками

Для роботи з адресними світлодіодними стрічками, в тому числі і зі стрічкою WS2812B, існують дві основні бібліотеки: FastLED і Adafruit NeoPixel.

Бібліотека Adafruit NeoPixel розробляється компанією Adafruit Industries. Призначена для роботи зі світлодіодними стрічками і неопіксельнимі кільцями, які продаються в їх інтернет магазині. Бібліотека написана на мовах програмування Сі і Ассемблері з невеликим використанням Wiring. Містить менший функціонал в порівнянні з FastLED, трохи повільніша, але має більш компактний вигляд, тільки основне для роботи.

Бібліотека FastLED розробляється Даніелем Гарсіа і Марком Крігсманом. Бібліотека написана тільки на мові Сі, без використання Wiring[6]. Бібліотека FastLED містить безліч функцій, оптимізованих для роботи з 8-бітними цілими числами без знака для зберігання значень кольору, включаючи додавання кольору, 8-бітове масштабування, частково певні колірні палітри.Бібліотека FastLED має безліч зручних інструментів, протоколів і інтерфейсів, а також швидку 8-бітну математику, чого немає в бібліотеці NeoPixel і в меньш популярній бібліотеці LightWS2812, в якій зовсім немає ніяких допоміжних інструментів.З огляду на всі переваги бібліотеки FastLED я вибрала саме її для свого проекта.Далі будуть детально описані її можливості і переваги.[7]

1.2 Підтримка і сумісні пристрої

Чіпсети на основі інтерфейсу SPI, які зазвичай складаються з 4 проводів - даних, годин, живлення і заземлення. Ці світлодіодні чіпсети мають перевагу через свою дешевизну. Протокол даних SPI добре працює на відстані, а з деякими наборами мікросхем можна отримати дійсно високу швидкість передачі даних.

Бібліотека підтримує наступні чіпсети:

– lpd8806 - чіпсет на базі набору мікросхем SPI, забезпечує високу швидкість запису даних;

– ws2801 - чіпсет на базі SPI з більш низькою швидкістю, швидкість передачі даних обмежена до 1 Мбіт / с, мають низьку вартість;

– sm16716 - ще один чіпсет на базі SPI;

– total control lighting - його друга назва P9813, це набір мікросхем SPI, який починає знаходити все більше застосування;

– apa102 - упаковка аналогічна WS2812, але потенційно може мати набагато більш високі швидкості передачі даних (з FastLED2.1).

Трипровідні світлодіодні пікселі стають досить популярними. Маючи тільки лінії передачі даних, заземлення та живлення, вони трохи більш компактні, ніж набори мікросхем на основі SPI (навіть більш того, WS2812B об'єднує мікросхему контролера світлодіодів і світлодіод в одному корпусі).

– neopixel - також відомий як WS2811 (або WS2812, або WS2812B), зараз багато людей вважають за краще саме ці світлодіоди RGB;

– tm1809 / tm1804 - поширений дешевий набір мікросхем, має протокол, аналогічний WS2811 і іншим подібним;

– tm1803;

– ucs1903 / ucs1903b / ucs1904 / ucs2903 - ще один трьохпровідний набір мікросхем, повільний протокол передачі даних;

– gw6205 (з FastLED2.1);

– lpd1886 - трьохпровідний набір мікросхем, який має 12 біт на піксель замість зазвичай 7/8 біт на піксель, які до сих пір використовуються в більшості інших наборів мікросхем (з FastLED2.1).

На даний момент з'явилося безліч нових світлодіодних чіпсетів, які буде підтримувати бібліотека:

– as1130;

– tlc5940;

– tm1829 - трьохпровідний чіпсет, який також дозволяє динамічно змінювати енергоспоживання базової лінії;

– tm1812;

– d3001 / cy3001.

Також має підтримку багатьох контролерів:

– плати Arduino & аналоги і репліки - UNO, Duo, Leonardo, Mega, Nano, Mini, Micro і т.д. (На даний момент повинні підтримуватися практично всі офіційні платформи Arduino, включаючи Due Yún і Zero);

– arduino yún(З FastLED 3.0.3);

– adafruit trinket & gemma (чіпсети на базі ATtiny для переносних проектів);

– teensy 2, teensy ++ 2, teensy 3.0, teensy 3.1 / 3.2, teensy lc, teensy 3.5, teensy 3.6 і teensy 4.0 проектні плати на базі avr і arm від PJRC, схожі з Arduino, але з безліччю додаткових плюсів;

– arduino due і digix(з fastled2.1);

– rfduino;

– sparkcore;

- arduino zero(з fastled 3.1);
- esp8266;
- плати Wino;
- esp32 [8][9].

1.3 Унікальні функції бібліотеки FastLED

На додаток до швидкого, ефективного та сумісного коду драйвера світлодіодів FastLED також надає функції, які дозволяють швидко запускати анімацію, а також має такі переваги:

- повна підтримка кольору HSV, а також класичний RGB;
- основні налаштування яскравості контролюють яскравість, енергоспоживання і термін служби батареї;
- швидкі математичні обчислення і функції пам'яті до 10 разів швидше, ніж стандартні бібліотеки Arduino;
- багаторічна історія активного розвитку і еволюції [10].

1.3.1 Математика для налаштування кольорів

При програмуванні світлодіодів часто потрібно обчислити значення rgb або hsv, щоб забезпечити переходи по яскравості і кольору. Але платформа AVR / Arduino має досить повільну математику. Тому для бібліотеки FastLED є перевагою те, що вона надає ряд математичних функцій, налаштованих для 8-бітних операцій, у тому числі функції масштабування, швидкі функції sin / cos, швидкі генератори випадкових чисел та управління пам'яттю[9].

У бібліотеці FastLED безліч математичних функцій для роботи з відтінками. У програмуванні світлодіодів використання математики робить

програму більш простою для читання, більш швидкою у записі і прискорює виконання програми.

Наприклад, додавання двох кольорів RGB:

Якщо треба «додати» значення кольору CRGB «newTint» до існуючих значень RGB світлодіода. (Наприклад, newTint дорівнює (100,50,0).) Раніше доводилося робити ось таке додавання, щоб переконатися, що додавання нових значень до існуючих не викликає «переповнення» більше одного байта:

```
int newRed = led [i] .r + newTint.r;
if (newRed> 255) newRed = 255;
leds [i] .r = newRed;

int newGreen = led [i] .g + newTint.g;
if (newGreen> 255) newGreen = 255;
leds [i] .g = newGreen;

int newBlue = led [i] .b + newTint.b;
if (newBlue> 255) newBlue = 255;
leds [i] .b = newBlue;
```

Бібліотека надає функцію, яка виконує складання двох восьмибітових чисел. При восьмібітного складання з насиченням, якщо сума перевищує 255, вона автоматично фіксується до 255. 8-розрядна функція додавання з насиченням називається qadd8 (x, y).

```
// 100 + 100 = 200
sum = qadd8 (100, 100); // -> 200
// 200 + 200 = 255
sum = qadd8 (200, 200); // -> насичення при 255
```

Це дуже корисно для додавання значень каналів R, G і B, оскільки автоматично запобігає переповнення і зациклення. При використанні qadd8 код, необхідний для додавання «CRGB newTint» до існуючого значенням світлодіода, тепер стає простіше, без «if»:

```
leds [i] .r = qadd8 (leds [i] .r, newTint.r);
leds [i] .g = qadd8 (leds [i] .g, newTint.g);
leds [i] .b = qadd8 (leds [i] .b, newTint.b);
```

Крім функції додавання трьох кольорів світлодіодного каналу також є пряма підтримка додавання кольорів у клас CRGB. Завдяки цьому можна обійтися навіть без попередніх прикладів і просто написати:

```
leds [i] + = newTint;
```

і бібліотека подбає про додавання каналів R, G і B, використовуючи 8-бітний метод насичення quadd8.

Також варто відзначити, що через специфічну мову асемблера, який використовували розробники для реалізації методу, який виконує таку операцію як: «додати цей колір CRGB до цього кольору CRGB, використовуючи математику для кольорів» код вийшов швидше і компактніше, ніж код на мові «C».

Бібліотека має можливість внутрішньо зменшувати числа для своїх глобальних налаштувань яскравості, але також користувач має можливість самостійно зменшити діапазон значень, щоб створити свої власні ефекти загасання. 8-бітові значення йдуть від 0 до 255 і найважливіша особливість функції масштабування в цій бібліотеці - це те, що вона дозволяють користувачеві встановити нову верхню межу. Наприклад ось так:

```
scaled = scale8 (val, 100);
```

Цей рядок масштабує значення 0-255 до значення 0-100.

Також якщо користувачеві знадобиться випадкове число, або 8-бітові, або 16-бітові значення, а функції для випадкових значень за замовчуванням, що надаються бібліотекою Arduino, занадто повільні, то в цій бібліотеці є 6 функцій з випадковими значеннями, 3 8-бітних і 3 16-бітних функції, які можна використовувати:

- random8 () == випадкове число від 0..255;
- random8 (n) == випадкове число від 0 .. (N-1);
- random8 (n, m) == випадкове число від N .. (M-1);
- random16 () == випадкове число від 0..65535;
- random16 (n) == випадкове число від 0 .. (N-1);
- random16 (n, m) == випадкове число від N .. (M-1) [11].

1.3.2 Робота з колірним простором

У бібліотеці є два основні класи для роботи з кольором: клас CRGB і клас CHSV.Дана бібліотека самостійно робить перетворення HSV в RGB.В деяких випадках використовувати колірний простір HSV набагато простіше, ніж при використанні RGB. При визначенні кольорів за допомогою RGB змішуються значення червоного, зеленого і синього кольорів. Замість цього, використовуючи HSV, користувач визначає відтінок кольору (тобто, де він знаходиться на колірному колі), наскільки він насичений і наскільки він яскравий. [9]

Наприклад, ось простий код, який циклічно змінює кольори веселки:

```
#include "FastLED.h"
CRGB leds [60];
void setup() {FastLED.addLeds <NEOPIXEL, 6> (leds, 60); }
void loop() {
```

```
static uint8_t hue = 0;
FastLED.showColor (CHSV (hue ++, 255, 255));
delay (10);
}
```

«CRGB» - це об'єкт, який представляє колір в колірному просторі RGB. Він містить:

- однобайтовое значення (0-255), що представляє кількість червоного;
- однобайтовое значення (0-255), що представляє кількість зеленого;
- однобайтовое значення (0-255), що представляє кількість синього в даному кольорі.

Зазвичай при використанні цієї бібліотеки кожна світлодіодна стрічка представлена у вигляді масиву кольорів CRGB, по одному кольору для кожного пікселя світлодіода.Такі парамаметри треба вказати,щоб створити масив об'єктів класу CRGB, де зберігаються кольори кожного світлодіода:

```
#define NUM_LEDS 160 //тут треба вказати кількість светлодіодів на стрічці
CRGB leds [NUM_LEDS];
```

CRGB має три однобайтових елемента даних, кожен з яких представляє один з трьох каналів червоного, зеленого і синього кольору. Існує кілька способів доступу до даних RGB.Каждий з наступних прикладів робить одне і те ж:

```
leds [i] .red = 50;
leds [i] .green = 100;
leds [i] .blue = 150;

leds [i] .r = 50;
```

```
leds [i] .g = 100;
leds [i] .b = 150;

leds [i] [0] = 50; // red
leds [i] [1] = 100; // green
leds [i] [2] = 150; // blue

leds [i] = CRGB (50, 100, 150);
```

Наведені вище приклади генерують ідентичний машинний код, займаючи однаковий обсяг програмної пам'яті і виконуючись за точно такий же час. Таким чином, вибір способу написання коду повністю залежить від особистого смаку і стилю.

Ось інші високорівневі способи встановити колір CRGB за один крок:

```
leds [i] = 0xFF007F; // в hex-форматі (0xRRGGBB)
leds [i] = CRGB :: HotPink; // через веб-колір HTML
leds [i] .setRGB (50, 100, 150); // через setRGB
```

Для програміста, орієнтованого на продуктивність, варто відзначити, що всі наведені вище приклади компілюються в точно таку ж кількість машинних інструкцій.

Кольори також можна копіювати з одного CRGB в інший:

```
leds [i] = leds [j]; // копіювання кольору CRGB з одного світлодіода в інший
```

Використовуючи тільки значення RGB складно висловити різні відтінки і відтінки одного кольору і може бути особливо складно описати «розмиття кольору» в RGB, яке циклічно обертається навколо веселки відтінків, зберігаючи при цьому ту саму яскравість.

Щоб спростити роботу з кольором, бібліотека надає доступ до альтернативної колірної моделі, заснованої на трьох різних осях: відтінок, насиченість і значення (або «яскравість»):

- відтінок - це кут навколо колірного кола;
- насиченість - це те, наскільки багатий колір (в порівнянні з блідим);
- цінність - наскільки «яскравий» (в порівнянні з тьмяним) колір;

У бібліотеці об'єкт CHSV використовується для представлення кольору в колірному просторі HSV.

Часто в інших колірних просторах HSV відтінок представлений як кут від 0 до 360 градусів. Але для компактності, ефективності і швидкості ця бібліотека представляє відтінок як однобайтовое число від 0 до 255 (рис.1.1). Об'єкт CHSV має три однобайтових елемента даних:

– «відтінок» (hue) - це однобайтовое значення в діапазоні від 0 до 255, де 255-це фіолетовий, 128-це бірюзовий, а 0-це червоний ".

– «насиченість» (saturation) - це однобайтовое значення в діапазоні від 0 до 255, де 255 означає «повністю насичений, чистий колір», 128 означає «наполовину насичений, світлий, блідий колір», а 0 означає «повністю ненасичений: простий. білий ".

– «яскравість» (value) - однобайтовое значення в діапазоні від 0 до 255, що представляє яскравість, де 255 означає «повністю яскравий, повністю освітлений», 128 означає «дещо тьмяний, напівосвітлений», а нуль означає «повністю темний: чорний».

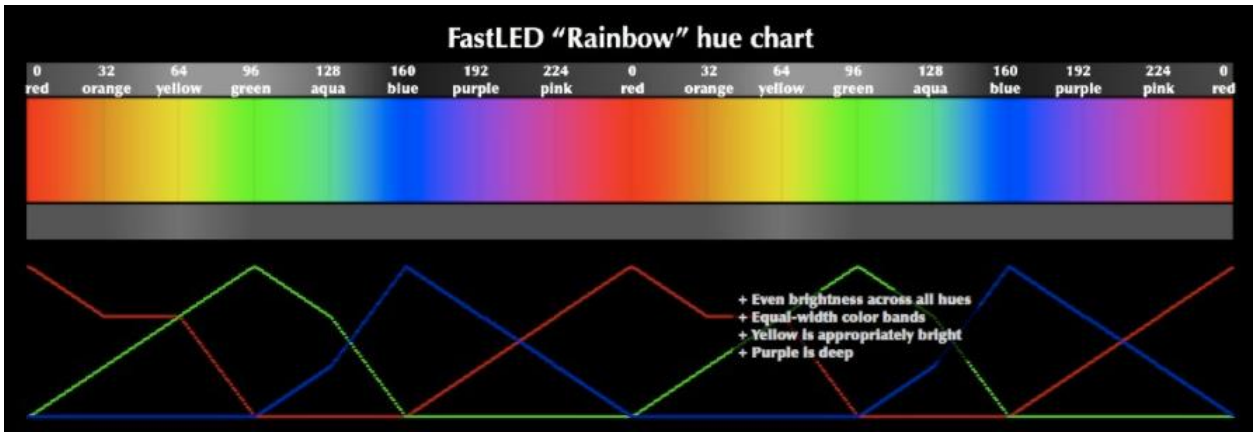

Рисунок 1.1 – Представлення відтінку

Бібліотека надає швидкі і ефективні методи перетворення кольору CHSV в колір CRGB. Багато з них є автоматичними і не вимагають явного коду.

Наприклад, щоб встановити для світлодіода колір, вказаний в HSV, можна просто записати колір в CHSV і він автоматично перетворюється в CRGB:

leds [i] = CHSV (160, 255, 255); // встановлюємо колір з HSV і перетворення в RGB відбувається автоматично.

leds [i] .setHSV (160, 255, 255); // альтернативний варіант

leds [i] .setHue (160); // з використанням setHue

Зворотного перетворення з CRGB в CHSV, що поставляється з бібліотекою, на даний момент немає.

Бібліотека підтримує багатий набір «математичних операцій з кольорами», які можна виконувати з одним або декількома кольорами. Наприклад, щоб додати трохи червоного до існуючого кольору світлодіода, можна зробити ось так:

leds [i] + = CRGB (20, 0, 0);

Але тут не вистачає перевірки червоного каналу, що виходить за межі 255. Традиційно потрібно написати ось такий код:

uint16_t newRed;

newRed = leds [i] .r + 20;

if (newRed> 255) newRed = 255; // запобігає переповнення

leds [i] .r = newRed;

Але завдяки спеціальній логіці цієї бібліотеки, можно не турбуватися про переповнення при додаванні двох кольорів CRGB всередині оператора + і оператора + =. Більш того, велика частина цієї логіки реалізована безпосередньо на мові асемблера і значно менше і швидше, ніж відповідний код C / C ++. Перевага цієї логіки в тому, що більше не потрібно виконувати всі перевірки самостійно і програма працює швидше.Ці «математичні операції з кольорами» є частиною того, що робить бібліотеку швидкою: вони дозволяють швидше розробляти код, а також швидше його виконувати.

Всі математичні операції, визначені для кольорів CRGB, автоматично захищені від зациклення, переповнення і втрати значущості:

– leds [i] .addToRGB (20); // додавання числа 20 до всіх трьох (RGB) каналів;

– leds [i] ++; // додавання числа 1 до всіх трьох (RGB) каналів;

– leds [i] - = CRGB (20, 0, 0); // віднімання одного кольору CRGB з іншого;

– leds [i] .subtractFromRGB (20); // альтернативний спосіб такого ж віднімання;

– leds [i] -; // віднімання числа 1 з усіх трьох (RGB) каналів.

Бібліотека надає функцію, яка «інвертує» кожен канал RGB. Виконання цієї операції двічі призведе до того ж кольору, з якого ми почали:

leds [i] = -leds [i]; // інвертує кожен канал

Бібліотека надає функцію, яка дозволяє «обмежувати» кожен з каналів RGB, щоб він знаходився в межах заданих мінімумів і максимумів. Користувач має можливість встановити для всіх колірних каналів мінімальне або

максимальне значення. Потім їх можна об'єднати, щоб обмежити як мінімум, так і максимум.

Якщо значення будь-якого каналу нижче заданого мінімуму для цього каналу, то він буде підвищений до заданого мінімуму. Мінімум можна вказати окремо для кожного каналу (як CRGB) або як одне значення:

leds [i] | = CRGB (32, 48, 64);

leds [i] | = 96;

Якщо значення будь-якого каналу вище заданого максимуму для цього каналу, то воно буде зменшено до заданого максимуму. Максимум також можна вказати окремо для кожного каналу (як CRGB) або як одне значення.

leds [i] & = CRGB (192, 128, 192);

leds [i] & = 160; [12]

1.4 Робота зі стрічкою в моїй програмі

У моєму коді присутні такі налаштування для роботи зі світлодіодною стрічкою:

#include <FastLED.h> // Підключаємо бібліотеку FastLED

#define DATA_PIN 5 // Вказуємо пін на ESP32, до якого підключена стрічка

CRGB leds [120]; // Створюємо масив об'єктів класу CRGB, де зберігаються кольори кожного світлодіода

FastLED.addLeds <WS2812B, DATA_PIN, RGB> (leds, strip.ledCount);

// Задаємо основні налаштування для стрічки в setup() (назва адресної світлодіодної стрічки; пін на ESP32, до якого підключена стрічка;послідовність кольорів; назва об'єкта класу CRGB; змінна з кількістю світлодіодів на стрічці)

FastLED.setBrightness (150); // Крім основних налаштувань в процедурі void setup() ще вказується яскравість для світлодіодів за допомогою команди FastLED.setBrightness () - максимальне значення 255. Яскравість можна змінити в будь-якому місці програми, в тому числі і в процедурі void loop ().

У моєму проекті буде сайт, з якого будуть відправлятися дані про режим і яскравість на сервер Esp32, також там будуть кілька самостійних web-сторінок. Ці web-сторінки містять в собі види режимів. У кожної є різні режими, які діляться на такі види:

- вбудовані режими;
- змінні режими;
- стробоскоп;
- вкладка для зміни яскравості.

Для кожного виду режимів світлової анімації в моєму коді створений свій клас і також своя web-сторінка.Далі буде детально описано кожен клас.

Вбудовані режими (class BuiltIn) (рис. 1.2):

```
class BuiltIn{
  public:
  int mode = 1;
  void Show(int count){
    switch(mode){
      case 1:
        rainbow_loop(count);
      break;
      case 2:
        rainbow_fade(count);
      break;
      case 3:
        random_burst(count);
      break;
      case 4:
        rgb_propeller(count);
      break;
      case 5:
        random_color_pop(count);
      break;
    }
  }
```

Рисунок 1.2 – Фрагмент для вбудованих режимів

При включенні стрічки за замовчуванням буде включатися перший режим з класу BuiltIn. Далі режим, який зберігається в змінній mode, буде змінюватися в залежності від даних, які прийдуть з сайту. Для встановлення режиму на стрічці серед певного класу буде використовуватися функція Show с параметром count, в якому буде зберігатися кількість світлодіодів на стрічці. Кожен елемент класу має рівень доступу - один з public (відкритий), private (закритий), protected (захищений). Функція Show має відкритий рівень доступу (public). Функції для кожного режиму матимуть private (закритий) рівень доступа. Також у функцій для кожного режиму є параметр countL, в якому буде зберігатися кількість світлодіодів на стрічці.

private:

int ihue;

Змінна ihue створена для зберігання відтінку кольору.

Функція rainbow_loop це світлові ефекти з райдужним переливанням(рис. 1.3):

```
void rainbow_loop(int countL) {
  for(int i=0;i<=countL;i++){
   ihue = ihue + 10;
  if (ihue > 255) {
    ihue = 0;
  }
  leds[i] = CHSV(ihue, 255, 255);
  LEDS.show();
  delay(20);
  }
}
```

Рисунок 1.3 – Фрагмент коду режиму rainbow_loop

У даній функції за допомогою зміни параметра hue в об'єкті CHSV виходить ефект переливання веселки.

Стандартна команда з бібліотеки FastLED "FastLED.show (); "Відправляє інформацію на стрічку.

Функція rainbow_fade- це плавна зміна кольорів всієї стрічки(рис. 1.4) :

```
void rainbow_fade(int countL) {
ihue++;
if (ihue > 255) {
  ihue = 0;
}
for (int i = 0 ; i < countL; i++ ) {
  leds[i] = CHSV(ihue, 255, 255);
}
FastLED.show();
delay(20);
}
```

Рисунок 1.4 – Фрагмент коду режиму rainbow_fade

Функція random_burst (рис. 1.5)-це поява нових випадкових кольорів на випадкових світлодіодах, при цьому нові кольори залишаються світитися і вся стрічка світиться різними кольорами:

```
void random_burst(int countL) {
  int i = random(0, countL);
  ihue = random(0, 255);
  leds[i] = CHSV(ihue, 255, 255);
 FastLED.show();
  delay(20);
}
```

Рисунок 1.5 – Фрагмент коду режиму random_burst

Для визначення випадкового світлодіода і випадкового кольору використовується функція random (min, max), яка повертає псевдовипадкове число. Якщо при кожному запуску програми необхідно отримувати різні послідовності значень, що генеруються функцією random(), то необхідно ініціалізувати генератор псевдовипадкових чисел з випадковим параметром.

Наприклад, можна використовувати значення, що віддається функцією analogRead() з непідключеного порту [13].

У моїй функції для визначення випадкового світлодіода нижня межа (min) випадкових значень-це нуль, верхня межа-кількість світлодіодів в стрічці, яка записана в змінній countL. Для визначення випадкового кольору в функції random() діапазон значень від 0-255.

У функції rgb_propeller (RGB пропелер) (рис. 1.6) використовуються тільки 3 кольори (червоний, зелений і синій), які плавно переміщуються один за одним по всій стрічці.

```
void rgb_propeller(int countL) {
  for(int j=0;j<=countL;j++){
  int N3  = int(countL / 3);
  int N6  = int(countL / 6);
  int N12 = int(countL / 12);
  for (int i = 0; i < N3; i++ ) {
    int j0 = (j + i + countL - N12) % countL;
    int j1 = (j0 + N3) % countL;
    int j2 = (j1 + N3) % countL;
    leds[j0] = CRGB::Red;
    leds[j1] = CRGB::Blue;
    leds[j2] = CRGB::Green;

  }
 FastLED.show();
  delay(25);
  }
}
```

Рисунок 1.6 – Фрагмент коду режиму rgb_propeller

Функція random_color_pop(рис. 1.7) -це випадкові спалахи, при цьому на стрічці світиться тільки один випадковий світлодіод, а інша стрічка вимкнена, але так як світлодіоди змінюються дуже швидко, створюється враження що на стрічці світиться відразу кілька світлодіодів:

```
void random_color_pop(int countL) {
  int index = random(0, countL);
  ihue = random(0, 255);
  for(int i=0;i<=countL;i++){
    leds[i]=CRGB::Black;
  }
  leds[index] = CHSV(ihue, 255, 255);
 FastLED.show();
  delay(35);
}
```

Рисунок 1.7 – Фрагмент коду режиму random_color_pop

Змінні режими (клас Creator)(рис. 1.8)

```
class Creator{
  public:
  int colorRed=0;
  int colorBlue=255;
  int colorGreen=0;
  int mode = 3;
  void Show(int count){
    switch(mode){
      case 1:
        GoUp(count);
      break;

      case 2:
        GoUpDown(count);
      break;

      case 3:
        Feel(count);
      break;

      case 4:
        FeelBlink(count);
      break;
      case 5:
        FromCenter(count);
      break;
      case 6:
        ToCenter(count);
      break;
    }
  }
```

Рисунок 1.8 – Фрагмент коду зі змінних режимів

У цьому класі є три змінні для запису в них даних про кожного з трьох кольорів. Ці дані приходять із запиту, отриманого з клієнтської частини програми, в форматі цілих чисел від 0 до 255. При переході у вкладку «Змінні режими» за замовчуванням буде включатися третій режим (int mode = 3). Також як і в інших класах, тут є функція Show для вибору режиму.

Для кожного режиму також є своя функція. Функції створені на основі циклів for з різними параметрами, які будуть передаватися в об'єкт CRGB у вигляді трьох параметрів r, g, b.Ось приклад одного з режимів(рис. 1.9) :

```
void GoUpDown(int countL){
  for(int i=0;i<=countL;i++){
  leds[i]=CRGB(colorGreen,colorRed,colorBlue);
  FastLED.show();
  delay(10);
  for(int j=0;j<=countL;j++){
   leds[j]=CRGB(0,0,0);
  }
  FastLED.show();
  delay(10);
}
```

Рисунок 1.9 – Фрагмент коду режиму

Стробоскоп (клас Strobo) (рис. 1.10) :

```
class Strobo{
  public:
  int colorRed=255;
  int colorBlue=255;
  int colorGreen=255;
  int frequency=480;
  void Show(int count){
      for(int i=0;i<=count;i++){
        leds[i]=CRGB(0,0,0);
        }
        delay(20);
        FastLED.show();
      for(int i=0;i<=count;i++){
        leds[i]=CRGB(colorGreen,colorRed,colorBlue);
        }
        delay(frequency);
        FastLED.show();
  }
};
```

Рисунок 1.10 – Фрагмент коду з класу «Стробоскоп»

У класі Strobo є додаткова змінна frequency для зберігання частоти миготіння.

Клас для стрічки (Strip):

Також є основний клас, який відповідає за розподіл режимів. У ньому створюються об'єкти всіх класів режимів, змінна ledCount з кількістю світлодіодів і змінна modeStr для вибору виду режиму (за замовчуванням буде запускатися клас з вбудованими режимами)(рис. 1.11) :

```
BuiltIn builtIn;
Creator creator;
Strobo objStrobo;
int ledCount=120;
int modeStr=1;
```

Рисунок 1.11 – Фрагмент коду з класу Strip

І також тут знаходиться функція ShowStr(рис. 1.12) для вибору виду режима.В кожному класі буде виконуватися своя функція Show і буде передаватися параметр у вигляді змінної ledCount.

```
public:
void ShowStr()
{
  switch(modeStr)
  {
    case 1:
    builtIn.Show(ledCount);
    break;
    case 2:
    creator.Show(ledCount);
    break;
case 3:
    objStrobo.Show(ledCount);
    break;

  }
}
```

Рисунок 1.12 – Фрагмент коду з класу Strip

Далі до void setup() створюється об'єкт класу Strip:

Strip strip;

І в циклі запускається функція ShowStr з класу Strip для вибору виду режиму(рис. 1.13) :

```
void loop(){
  strip.ShowStr();

}
```

Рисунок 1.13 – Фрагмент коду

2 ВЕБ-СЕРВЕРА І РОБОТА З НИМИ

«Запит / відповідь» - це схема обміну повідомленнями, при якій ініціатор запиту відправляє повідомлення-запит системі-відповідачу, яка отримує і обробляє цей запит, а потім відправляє у відповідь повідомлення. Це проста і ефективна схема обміну повідомленнями, і особливо - для архітектур типу «клієнт-сервер».

При записи URL в адресний рядок браузера клієнт відправляє запит серверу за допомогою протоколу HTTP (англ. «Hypertext transfer protocol», тобто «протокол передачі гіпертексту»). Отримавши запит, сервер відправляє відповідь (теж по HTTP), в результаті чого можна побачити в браузері запитану веб-сторінку. Клієнти і сервери коммунікують один з одним по комп'ютерній мережі.

HTTP (HyperText Transfer Protocol - «протокол передачі гіпертексту») - протокол передачі даних, спочатку - у вигляді гіпертекстових документів в форматі HTML, в даний час використовується для передачі довільних данних.[14]

HTTP визначає безліч методів запиту, які вказують, яка бажана дія виконається для даного ресурса[15].

HTTP метод GET відсилає запит на отримання даних. Запити клієнтів, що використовують метод GET повинні тільки отримувати дані і не повинні ніяк впливати на ці дані. Є обмеження по символам, так як дані передаються в URL, то є обмеження у 2048 символах (максимальний рядок символів в URL). За типом даних допускається використання тільки символів ASCII. Меньш безпечний, тому що передані в URL дані видно користувачеві. Дані в URL доступні всім, тому GET-запити не рекомендується використовувати з приватною інформацією. Також важливо, що GET-запит не має тіла[15][16].

GET-запит доступний тільки для читання, але як тільки клієнт отримає дані, він може самостійно виконувати будь-які операції з ними, наприклад, змінити яскравість або режим на стрічці.[17]

Клієнт може передавати параметри виконання запиту після символу «?»: GET / path / resource? Param1 = value1 & param2 = value2 HTTP / 1.1 [18]

Веб-сервер можна уявити як ПО, яке прослуховує вхідні HTTP-запити, а також відправляє відповіді після отримання запитів.

IP-адреса - це "числовий підпис", який присвоєно кожному пристрою, підключеному до комп'ютерної мережі. Отже, звернувшись до потрібної IP-адреси, можна відправити мережевому пристрою будь-яку інформацію. У будь-якої ESP32, підключеної до комп'ютерної мережі, теж буде своя IP-адреса.

2.1 Класи в бібілотека WiFi.h

Клас Client створює клієнтів, які можуть отримувати доступ до служб, які надаються серверами, для відправки, отримання та обробки даних.

У цьому класі є функція, яка підключається до вказаних IP-адрес і порту. Повертає значення про те, чи успішним було підключення чи ні. Також підтримує пошук DNS - на той випадок, якщо підключення буде здійснюватися до доменного імені (наприклад, до «google.com»):

client.connect (ip, port); // для IP-адреси

client.connect (URL, port); // для доменного імені

ip - IP-адреса, до якої буде підключатися клієнт (масив з 4 байтів)

URL - доменне ім'я, до якого буде підключатися клієнт. Тип даних - string. Наприклад «arduino.cc»

port - порт, до якого буде підключатися клієнт. Тип даних - int

Якщо підключення виконано успішно - повертає true, якщо ні - false.

Клас Server створює сервери, які надають функціональні можливості іншим програмам або пристроям, які називаються клієнтами.

Клієнти підключаються до сервера для відправки та отримання даних і доступу до наданих функцій.

У цьому класі є функція, яка змушує сервер почати прослуховування вхідних підключень:

server.begin();

І також є функція для запису даних на всіх клієнтів, підключених до серверу:

server.write (data); // data - значення, які потрібно записати. Тип даних - byte або char.

Повертає кількість записаних байтів (зчитувати це необов'язково). Тип даних - byte.

Клас WiFi ініціалізує бібліотеку і основні мережеві налаштування.

Цей клас буде детально розглянуто в подальших розділах.

Клас IPAddress надає інформацію про поточні налаштування мережі.

Наприклад, за допомогою цього класу ми можемо отримати інформацію про IP-адресу WiFi-модуля:

```
#include <WiFi.h>
 char ssid [] = "yourNetwork"; // SSID вашої мережі
  int status = WL_IDLE_STATUS; // статус WiFi-з'єднання
  IPAddress ip; // IP-адреса WiFi-модуля
  void setup () {
Serial.begin (9600); // ініціалізуємо послідовну комунікацію
  WiFi.begin (ssid);
   if (status! = WL_CONNECTED) {
```

```
Serial.println ( "Could not get a wifi connection"); // "WiFi-з'єднання не встановлено"
while (true);
} // якщо підключилися, показуємо інформацію про з'єднання:
else {
ip = WiFi.localIP (); // показуємо локальну IP-адресу:
Serial.println (ip);
}}
void loop () {}
```

Клас UDP дозволяє відправляти і отримувати UDP-повідомлення.

В цьому класі існує функція begin(), яка ініціалізує клас WiFiUDP і мережеві налаштування. Запускає WiFiUDP-сокет з прослуховуванням зазначеного локального порту:

```
WiFiUDP.begin (port);
```

Де, port - локальний порт, який потрібно прослуховувати. Тип даних - int. Якщо все минуло успішно - повертає «1», якщо немає сокетів, доступних для використання - «0».

І також є функція stop(), яка відключає від сервера і звільняє всі ресурси, які використовувалися під час UDP-сесії:

```
WiFiUDP.stop ();
```

[19] [20][21]

2.2 Алгоритм роботи сервера

Кожен запит і відповідь від сервера до клієнта і навпаки має певний формат і стандарт, одним з форматів є HTTP.Протокол HTTP має в своїх запитах і відповідях такі параметри:

Запит від клієнта:

- адреса посилання;
- метод;
- заголовки запиту (версія і дані про клієнта);
- тіло (основна інформація).

Відповідь від сервера:

- статус відповіді;
- заголовки відповіді;
- заголовки (версія і дані про сервер);
- тіло, відповідь сервера.

Весь HTML код можна використовувати як відповідь сервера від запиту клієнта. У подальшому коді моєї програми будуть створені змінні, в яких буде зберігатися код HTML сторінок.

При запиті клієнта сервер відповідає тільки один раз. За цей раз він може передати тільки один тип документу. Сторінка формату html має тип text/html. Файл формату css має text/css.Зображення png мають формат image / png.

Синхронний сервер має наступний алгоритм відповіді для клієнта:

- початок з'єднання;
- передача заголовків відповіді;
- передача документа від сервера до клієнта;
- кінець з'єднання.

Для того, щоб, наприклад, завантажити на сторінці зображення, необхідно створити додатковий запит.

Синхронний сервер повинен обов'язково закрити з'єднання, так як за цей час він більше не зможе відповідати іншим клієнтам.

2.3 Режими Wi-Fi ESP32

Плата ESP32 може працювати як станція Wi-Fi, точка доступу або і те, і інше. Щоб встановити режим Wi-Fi, потрібно використовувати WiFi.mode () і встановіть бажаний режим як аргумент:

WiFi.mode (WIFI_STA) // режим станції: ESP32 з'єднується з точкою доступу

WiFi.mode (WIFI_AP) // режим точки доступу: станції можуть підключатися до ESP32

WiFi.mode (WIFI_STA_AP) // точка доступу і станція, підключена до іншої точки доступу

2.3.1 Станція Wi-Fi

Коли ESP32 налаштована як станція Wi-Fi, вона може підключатися до інших мереж (наприклад, до свого маршрутизатора)(рис. 2.1). У цьому випадку маршрутизатор призначає платі ESP унікальну IP-адресу. Також є можливість зв'язуватися з ESP з допомогою інших пристроїв (станцій), які також підключені до тієї ж мережі, використовуючи унікальну IP-адресу ESP. Пристрої, які підключаються до мереж Wi-Fi, називаються станціями (STA). Кожна точка доступу розпізнається за ідентифікатором SSID (Service Set IDentifier), який є ім'ям мережі, яку ви обираєте при підключенні пристрою(станції) до Wi-Fi.

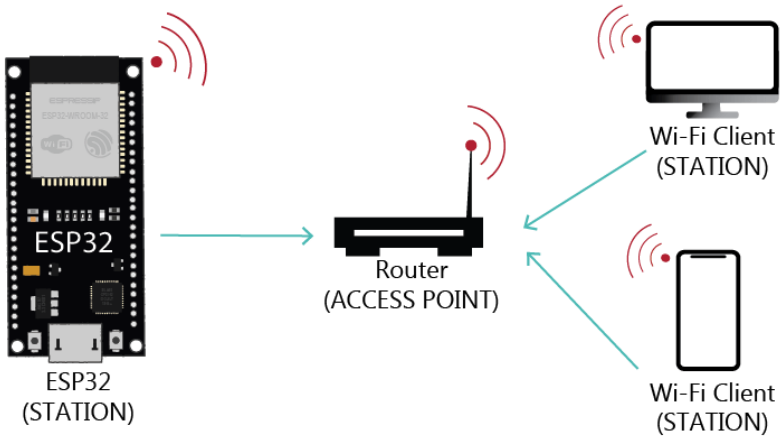

Рисунок 2.1 - Зв'язок з ESP32 у режимі станції

Маршрутизатор підключений до Інтернету, тому можна запитувати інформацію з Інтернету за допомогою плати ESP32, публікувати дані на онлайн-платформах, використовувати значки і зображення з Інтернету або включати бібліотеки JavaScript для створювання сторінки веб-сервера. У разі втрати з'єднання ESP32 автоматично перепідключитися до останньої використаної точки доступу, як тільки вона знову стане доступною. Те ж саме відбувається при перезавантаженні модуля. Це можливо, оскільки ESP зберігає облікові дані останньої використаної точки доступу у флеш-пам'яті (незалежній пам'яті).

2.3.2 Точка доступу

Якщо встановити плату ESP32 в якості точки доступу, то можна буде підключатися за допомогою будь-якого пристрою з можливостями Wi-Fi без підключення до маршрутизатора(рис. 2.2). Такий режим роботи називається програмної точкою доступу або ж soft-AP (soft Access Point). При установці ESP32 в якості точки доступу, створюється власна мережа Wi-Fi, і довколишні пристрої (станції) Wi-Fi можуть підключатися до неї, наприклад, смартфон або комп'ютер. Таким чином, не потрібно підключатися до маршрутизатора, щоб керувати ними. Максимальна кількість станцій, які можуть бути одночасно

підключені до soft-AP, може бути встановлено від 0 до 8, але за замовчуванням - 4.

Режим soft-AP часто використовується як проміжний крок перед підключенням ESP до Wi-Fi в режимі станції. Це коли SSID і пароль до такої мережі заздалегідь невідомі. ESP спочатку завантажується в режимі soft-AP, тому ми можемо підключитися до нього за допомогою ноутбука або мобільного телефону. Потім ми можемо надати облікові дані цільової мережі та ESP перемикається в режим станції і може підключатися до цільового Wi-Fi.Це також може бути корисно, якщо є необхідність, щоб кілька пристроїв ESP32 спілкувалися один з одним без маршрутизатора.

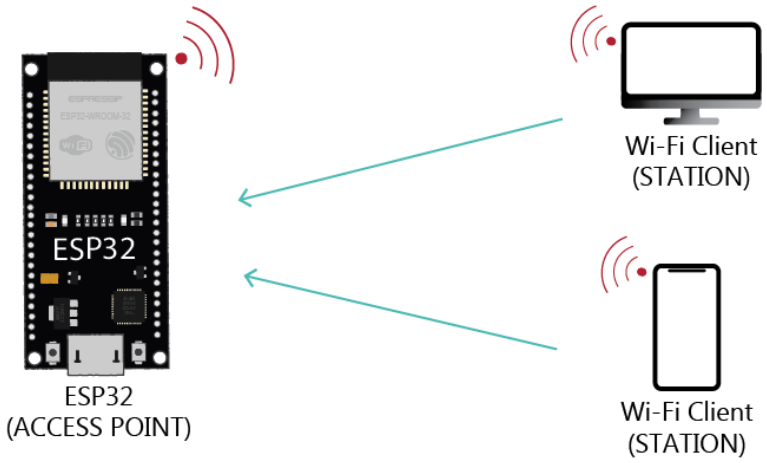

Рисунок 2.2 - Зв'язок з ESP32 у режимі точки доступу

У такому режимі не вийде зробити HTTP-запит до служб в Інтернеті для публікації показників датчиків в хмарі або використовувати служби в Інтернеті (наприклад, відправляти повідомлення електронної пошти).

Щоб встановити ESP32 в якості точки доступу, потрібно встановити режим Wi-Fi на точку доступу:

WiFi.mode (WIFI_AP)

А потім використати метод softAP ():

WiFi.softAP (ssid, password);

SSID -це ім'я, яке буде у точки доступу ESP32, а пароль – це змінна-пароль для точки доступу. Якщо пароль не потрібен, то потрібно встановити його на НУЛЬ.

Також в softAP () можна вказати інші необов'язкові параметри:

WiFi.softAP (const char * ssid, const char * password, int channel, int ssid_hidden, int max_connection)

SSID: ім'я точки доступу - максимум 63 символи;

пароль: мінімум 8 символів; якщо встановлений в НУЛЬ, то точка доступу буде відкритою;

канал: Номер каналу Wi-Fi (1-13)

ssid_hidden: (0 = широкомовний SSID, 1 = приховати SSID)

max_connection: максимальна кількість одночасно підключених клієнтів (1-4)[22]

2.3.2.1 Як обрати канал Wi-Fi для режиму точки доступу

Канали - це «підчастоти», на яких працюють Wi-Fi роутери і будь-яка інша техніка, яка використовує бездротове з'єднання(рис. 2.3). Є два основних діапазону частот - 2,4 і 5 ГГц, які поділені на канали. Плата ESP32 може працювати тільки на частоті 2,4 ГГц. Частоти поділені на канали, щоб до кожного каналу можна було підключити окремі пристрої, і вони не створили перешкоди один для одного. Канали завжди поділяються між клієнтами, які використовують його. Чим більше кількість активних клієнтів - тим менше швидкість отримає кожен з них. Точки доступу завжди чутливі до «сусідів», особливо якщо їх багато і вони працюють на пересічних каналах. Якість і швидкість роботи при цьому знижуються.Через перевантаженість каналу (внаслідок підключення до одного каналу безлічі інших мереж) відбуваються

часті обриви з'єднання по Wi-Fi, низька швидкість з'єднання, нестабільна робота і т.д.. Саме тому важливо обрати правильний канал для роботи плати. Щоб визначити потрібний канал необхідно просканувати всю мережу навколо і знайти вільні канали[23][24][25][26].

Для того, щоб визначити який канал Wi-Fi є найбільш незавантажений зараз, потрібно встановити додаток на комп'ютер або на телефон, який буде аналізувати мережі. І за допомогою програми проаналізувати який канал найбільш вільний.

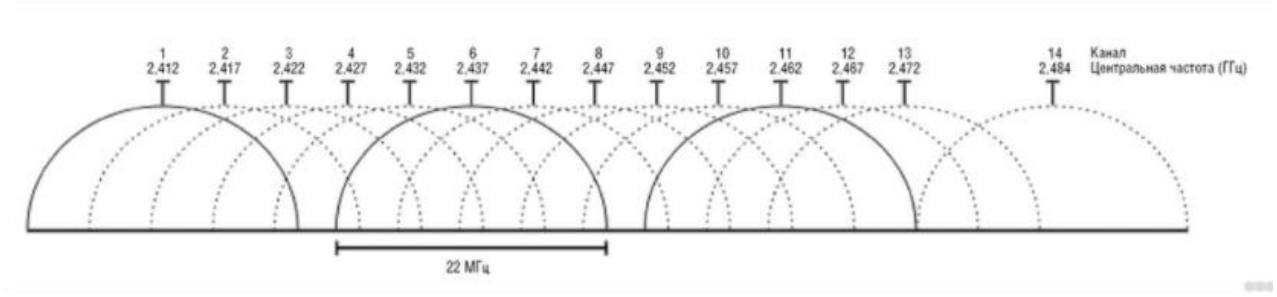

Рисунок 2.3 - Зображення каналів Wi-Fi

На частоті 2,4 ГГц в Україні дозволено використовувати з 1-го по 13-ий канали.

Ширина каналу 22 МГц. При цьому канали перетинаються між собою і через це мережі впливають один на одного, що може стати причиною низької швидкості мережі і інших проблем з мережею.

На рисунку виділені 3 канали - 1й, 6й і 11й. Ці три канали ніколи не перетинаються і саме тому рекомендовано їх використовувати в першу чергу, а далі обирати канал по завантаженості[27].

2.3.3 Станція Wi-Fi + точка доступу

ESP32 можна налаштувати одночасно як станцію Wi-Fi і точку доступу. Для цього потрібно встановити режим WIFI_AP_STA:

WiFi.mode (WIFI_AP_STA);

2.4 Базові функції бібліотеки Wi-Fi.h

2.4.1 Сканування мереж Wi-Fi

ESP32 може сканувати прилеглі мережі Wi-Fi в межах свого діапазону Wi-Fi. Це потрібно для перевірки, чи знаходиться мережа Wi-Fi, до якої виконується з'єднання, в межах досяжності нашої плати або інших додатків. Іноді проект на основі Wi-Fi може не працювати, тому що він не може підключитися до маршрутизатора через недостатню потужність Wi-Fi. Функціональність сканування і складання переліку доступних мереж в діапазоні реалізується класом Scan.

Є можливість перевірити доступні мережі, а також RSSI (індикатор рівня сигналу).

WiFi.scanNetworks () повертає кількість знайдених мереж:

int n = WiFi.scanNetworks ();

Після сканування можна отримати доступ до параметрів кожної мережі.

WiFi.SSID () друкує SSID для конкретної мережі:

Serial.print (WiFi.SSID (i));

WiFi.RSSI () повертає RSSI цієї мережі. RSSI позначає Received Signal Strength Indicator. Це оціночна міра рівня потужності, яку клієнтський пристрій отримує від точки доступу або маршрутизатора.

Serial.print (WiFi.RSSI (i));

2.4.2 Підключення до мережі Wi-Fi

Щоб підключити ESP32 до певної мережі Wi-Fi, потрібно знати її SSID і пароль. Крім того, ця мережа повинна бути в межах діапазону Wi-Fi ESP32.

Наприклад можна створити функцію initWiFi () для підключення ESP32 до мережі Wi-Fi:

```
void initWiFi () {
 WiFi.mode (WIFI_STA);
 WiFi.begin (ssid, password);
 Serial.print ("Connecting to WiFi ..");
 while (WiFi.status ()! = WL_CONNECTED) {
 Serial.print ( '.');
 delay (1000);
 }
 Serial.println (WiFi.localIP ());
}
```

Потім потрібно викликати функцію initWiFi () в setup ().

Спочатку в цій функції встановлюємо режим Wi-Fi. Якщо ESP32 буде підключена до іншої мережі, то вона повинен бути в режимі станції:

```
WiFi.mode (WIFI_STA);
```

Потім використовуємо WiFi.begin (), щоб підключитися до мережі. Потрібно передати в якості аргументів мережевий SSID і його пароль:

```
WiFi.begin (ssid, password);
```

SSID і пароль це змінні, який містять SSID і пароль мережі, до якої потрібно підключитися:

```
const char * ssid = "REPLACE_WITH_YOUR_SSID";
const char * password = "REPLACE_WITH_YOUR_PASSWORD";
```

Підключення до мережі Wi-Fi може зайняти деякий час, потрібно додати цикл while, який продовжує перевіряти, чи було з'єднання вже встановлено, за допомогою WiFi.status (). Коли з'єднання успішно встановлено, він повертає WL_CONNECTED:

while (WiFi.status ()! = WL_CONNECTED) {

2.4.3 Отримання статусу підключення до Wi-Fi

Щоб отримати статус підключення Wi-Fi, потрібно використовувати WiFi.status (). Цей метод повертає такі значення (перед двокрапкою вказано саме значення):

0: WL_IDLE_STATUS (тимчасовий статус. Він повертається, коли функція WiFi.begin() викликана і залишається активною. Якщо кількість спроб підключення буде вичерпано, цей статус змінюється на WL_CONNECT_FAILED, а якщо з'єднання буде успішно встановлено, то на WL_CONNECTED)

1: WL_NO_SSID_AVAIL (Коли SSID недоступні)

2: WL_SCAN_COMPLETED (Сканування мереж завершено)

3: WL_CONNECTED (При підключенні до мережі Wi-Fi)

4: WL_CONNECT_FAILED (Коли з'єднання втрачено для всіх спроб)

5: WL_CONNECTION_LOST (Коли зв'язок втрачено)

6: WL_DISCONNECTED (При відключенні від мережі)

Коли ESP32 налаштована як станція Wi-Fi, вона може підключатися до інших мереж (наприклад, до маршрутизатора). У разі роботи ESP32 в режимі станції маршрутизатор призначає унікальну IP-адресу платі. Щоб дізнатися IP-адресу плати, потрібно викликати метод localIP () після встановлення з'єднання з мережею:

Serial.println (WiFi.localIP ());

Наприклад, ось так можно використати метод WiFi.RSSI(), який допомагає дізнатися рівень потужності сигналу Wi-Fi, після підключення Wi-Fi(рис. 2.4) :

#include <WiFi.h>

```
const char * ssid = "REPLACE_WITH_YOUR_SSID";
const char * password = "REPLACE_WITH_YOUR_PASSWORD";

void initWiFi () {
 WiFi.mode (WIFI_STA);
 WiFi.begin (ssid, password);
 Serial.print ( "Connecting to WiFi ..");
 while (WiFi.status ()! = WL_CONNECTED) {
 Serial.print ( '.');
 delay (1000);
 }
 Serial.println (WiFi.localIP ());
}
void setup () {
 Serial.begin (115200);
 initWiFi ();
 Serial.print ( "RRSI:");
 Serial.println (WiFi.RSSI ());
}
void loop () { }
```

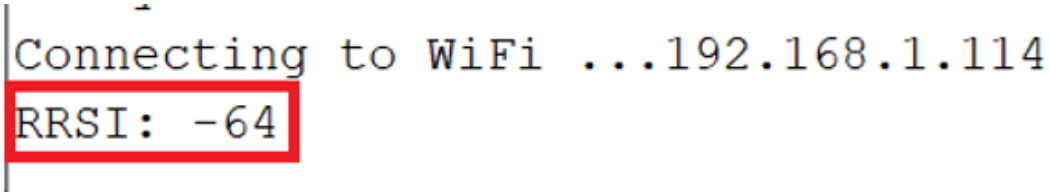

Рисунок 2.4-Результат виконання коду

RSSI може приймати значення від 0 до -100 дБм. Чим вище значення RSSI (ближче до 0), тим сигнал краще (більш потужний), і чим ближче до -100, тим

сигнал гірше (слабший). Якісним сигналом Wi-Fi можна вважати значення не нижче -65 дБм. [28][22][29]

2.5 Асинхронний сервер і його переваги

- асинхронний сервер може обробляти більше одного з'єднання одночасно;
- при написанні відповіді є можливість обробляти інші з'єднання, в той час як сервер піклується про відправку відповіді у фоновому режимі;
- обробка будь-якого типу контенту;
- висока швидкість.

Це повністю асинхронний сервер, тому він не працює в потоці циклу. Сервер самостійно визначає коли закрити з'єднання і звільнити ресурси. Також, як і в синхронному сервері, можна відправити тільки одну відповідь на один запит.

Асинхронний сервер дозволяє встановити з'єднання і передавати необхідну кількість даних, не закриваючи з'єднання з клієнтом. Сервер може відповідати одночасно кільком клієнтам.[30]

ESP32 може працювати в одному з режимів, як синхронний або асинхронний сервер.Для цього необхідно використовувати бібліотеку «ESPAsyncWebServer».

Бібліотеці «ESPAsyncWebServer» для роботи необхідна бібліотека «AsyncTCP». Бібліотека AsyncTCP для ESP32 Arduino-це повністю асинхронна бібліотека TCP, призначена для забезпечення безперебійного мережевого середовища з декількома підключеннями для мікроконтролерів Espressif ESP32.Ця бібліотека є базою для ESPAsyncWebServer. [31]

За допомогою бібліотеки «ESPAsyncWebServer» ми можемо налаштувати веб-сервер так, щоб він прослуховував вхідні HTTP-запити на різні URL і, в залежності від отриманого запиту, виконував ті чи інші функції. Для цього ми будемо використовувати метод on () на об'єкті «server». [32]

server.on () - команда формує відповідь від сервера при відповідному запиті. Що повинен відповісти або не відповісти сервер клієнту.

server.on ("URL", Method, Request) // Створюємо відповідь сервера на запит. Відповідей у сервера може бути багато, в залежності від запиту клієнта буде обиратися певна відповідь (за допомогою обробника запитів).

Тобто, коли клієнт звертається сервер може виконати певні дії за допомогою request, а потім можна сформувати відповідь від сервера до клієнта за допомогою server.on ().

Асинхронний веб-сервер прослуховує з'єднання, стежить за клієнтами і самостійно очищує пам'ять.

За допомогою такого коду можна перевірити чи є певний параметр GET:

```
if (request->hasParam("brightness")) {
        int brightness =request->getParam("brightness");
```

https://github.com/me-no-dev/ESPAsyncWebServer#print-to-response

2.5.1 Відповіді (Responses)

– ці об'єкти використовуються для передачі відповідних даних назад клієнту;

– залежно від типу відповіді використовуються різні методи для відправки даних в пакетах, які повертаються майже негайно і відправляють наступний пакет, коли він отриманий. Будь-який час між ними витрачається на запуск призначеного для користувача циклу і обробку інших мережевих пакетів[33].

Такий приклад може вивести відповідь користувачеві на єкран:

```
AsyncResponseStream *response = request-> beginResponseStream ("text/html");
```

```
response->addHeader("Server","ESP Async Web Server");
response->printf("<!DOCTYPE html><html><head><title>Webpage at %s</title></head><body>", request->url().c_str());
response->print("<h2>Hello ");
response->print(request->client()->remoteIP());
response->print("</h2>");
response->print("<h3>General</h3>");
response->print("<ul>");
response->printf("<li>Version: HTTP/1.%u</li>", request->version());
response->printf("<li>Method: %s</li>", request->methodToString());
response->printf("<li>URL: %s</li>", request->url().c_str());
response->printf("<li>Host: %s</li>", request->host().c_str());
response->printf("<li>ContentType: %s</li>", request->contentType().c_str());
response->printf("<li>ContentLength: %u</li>", request->contentLength());
response->printf("<li>Multipart: %s</li>", request->multipart()?"true":"false");
response->print("</ul>");
response->print("</body></html>");
request->send(response);
```

https://github.com/me-no-dev/ESPAsyncWebServer#print-to-response

2.5.2 Запити

Команда request-> send формує відповідь при запиті. Як повинен реагувати сервер на такий запит (відкрити файл, редирект, вимкнутися, авторизувати і т.д.).

Формуємо відповідь на запит від клієнта:

request-> send (code, doc_type, var) // Request є вказівником, тому використовується оператор -> для методу send () для передачі змінних. code-

код сервера в якості відповіді (200 ОК), doc_type -тип документа, var- змінна з контентом, який повинен відображатися.

Такий рядок коду перенаправить клієнта на інший URL:

request->redirect("/login"); //на локальний url

request->redirect("http://esp32.com"); //на зовнішній url

https://github.com/me-no-dev/ESPAsyncWebServer

Ось приклад базової відповіді з HTTP-кодом:

request-> send (404); // Відправляє 404 файл не знайдений

Ось приклад базового відповіді зі строковим вмістом:

request-> send (200, "text / plain", "Hello World!"); [30]

2.6 Робота з сервером в моїй програмі

У моєму коді присутні такі налаштування для роботи з сервером:

Підключаємо бібліотеки, які потрібні для роботи асинхронного сервера (рис. 2.5):

```
#include "WiFi.h"
#include "ESPAsyncWebServer.h"
```

Рисунок 2.5 – Фрагмент коду

Також перед setup вказуємо ідентифікатор і пароль від своєї WiFi-мережі(рис. 2.6):

```
const char *ssid = "MyWi-Fi";
const char *password = "MyPassword";
```

Рисунок 2.6 – Фрагмент коду

AsyncWebServer server (80); // Створюємо екземпляр класу «AsyncWebServer» під назвою «server» і задаємо йому номер порту «80»

Serial.begin (115200); // відкриває послідовний порт для налагодження та встановлюємо швидкість 115200 біт / с.

Підключення до мережі Wi-Fi з SSID і паролем. Перевіряємо статус підключення. Якщо немає з'єднання, то виводимо повідомлення про підключення(рис. 2.7) :

```
WiFi.begin(ssid, password);
while (WiFi.status() != WL_CONNECTED) {
  delay(1000);
  Serial.println("Connecting to WiFi..");
}
```

Рисунок 2.7 – Фрагмент коду

Serial.println (WiFi.localIP ()); // Виводимо в послідовний порт IP-адресу плати.

Веб-сервер буде прослуховувати вхідні HTTP-запити на різні URL і - в залежності від отриманого запиту - виконувати ті чи інші функції. При натисканні на кнопки і посилання браузер відправляє HTTP-запит GET з різними параметрами і на основі цієї URL-адреси ESP змінює параметри на стрічці.

URL для кореневої сторінки веб-серверу(рис. 2.8) :

```
server.on("/", HTTP_GET, [](AsyncWebServerRequest *request){
  request->send(200, "text/html", option);
});
```

Рисунок 2.8 – Фрагмент коду

server.on ("/", HTTP_GET, [] (AsyncWebServerRequest * request) {

// Створюємо відповідь сервера на запит за допомогою методу метод on () на об'єкті «server».

request-> send (200, "text / html", option); // формуємо відповідь, з кодом, типом даних і файлом для відкриття.

Відповідь сервера для сторінки з яскравістю, яка в коді записана в форматі HTML-сторінки і знаходиться в рядку під назвою option(рис. 2.9) :

```
server.on("/option.html", HTTP_GET, [](AsyncWebServerRequest *request){
  request->send(200, "text/html", option);
  if (request->hasParam("brightnes")) {
          int brightnes =request->getParam("brightnes")->value().toInt();
          FastLED.setBrightness(brightnes);
          Serial.println(brightnes);
      }
});
```

Рисунок 2.9 – Фрагмент коду

if (request-> hasParam ("brightnes")) {

int brightnes = request-> getParam ("brightnes") -> value (). toInt ();

FastLED.setBrightness (brightnes);

Serial.println (brightnes);

} // Отримуємо параметр brightnes для зміни яскравості, цей параметр приходить у вигляді рядка, за допомогою методу toInt () перетворюємо рядок в ціле число і записуємо в змінну brightnes. Встановлюємо яскравість за допомогою методу setBrightness() і виводимо яскравість в послідовний порт для налагодження .

HTTP-запит і відповідь для сторінки з вбудованими режимами(рис. 2.10) :

```
server.on("/custom.html", HTTP_GET, [](AsyncWebServerRequest *request){
  strip.modeStr=1;
  if (request->hasParam("mode")) {
    strip.builtIn.mode=request->getParam("mode")->value().toInt();
  }
  request->send(200, "text/html", custom);
});
```

Рисунок 2.10 – Фрагмент коду

strip.modeStr = 1; // вказуємо вид режимів (Вбудовані/Змінні/Стробоскоп), які зберігаються в класі Strip у змінній modeStr.

if (request-> hasParam ("mode")) {

strip.builtIn.mode = request-> getParam ("mode") -> value (). toInt ();

} // отримуємо параметр у вигляді рядка для вибору одного з режимів, які зберігаються в класі BuiltIn, перетворюємо рядок в ціле число і записуємо його в змінну mode.

request-> send (200, "text / html", custom); // формуємо відповідь, з кодом, типом даних і файлом для відкриття, який записаний в форматі HTML-сторінки і знаходиться в рядку під назвою custom.

Далі показаний фрагмент коду, який відповідає за зміну кольору світлодіодів на сторінці «Змінні режими»(рис. 2.11) :

```
if (request->hasParam("color")) {
    char red[5] = {0};
    char green[5] = {0};
    char blue[5] = {0};
    red[0] = green[0] = blue[0] = '0';
    red[1] = green[1] = blue[1] = 'X';
    String color;
    color=request->getParam("color")->value();
    red[2] = color[1];
    red[3] = color[2];
    green[2] = color[3];
    green[3] = color[4];
    blue[2] = color[5];
    blue[3] = color[6];
    strip.creator.colorRed = strtol(red, NULL, 16);
    strip.creator.colorGreen = strtol(green, NULL, 16);
    strip.creator.colorBlue = strtol(blue, NULL, 16);
}
```

Рисунок 2.11 – Фрагмент коду

При отриманні параметра color створюється масив з 5-ти чисел для кожного колірного каналу (red, green, blue):

```
char red [5] = {0};
char green [5] = {0};
char blue [5] = {0};
```

Перші два елемента у всіх каналів 0 і X відповідно, для створення префікса, що вказує на заснування шестнадцатеричної системи числення:

```
red [0] = green [0] = blue [0] = '0';
red [1] = green [1] = blue [1] = 'X';
```

Параметр color приходить у вигляді рядка, тому ми створюємо новий рядок color. Далі записуємо отриманий параметр в створений рядок:

```
String color;
color = request-> getParam ( "color") -> value ();
```

Дані про колір приходять в шістнадцятковому форматі #rrggbb, тому записуємо дані з рядка про кожний канал за допомогою такого коду:

```
red [2] = color [1];
red [3] = color [2];
green [2] = color [3];
green [3] = color [4];
blue [2] = color [5];
blue [3] = color [6];
```

Передаємо ці дані в змінні з класу creator:

```
strip.creator.colorRed = strtol (red, NULL, 16);
strip.creator.colorGreen = strtol (green, NULL, 16);
strip.creator.colorBlue = strtol (blue, NULL, 16);
```

Функція strtol () перетворює строкове представлення числа, яке зберігається в рядку string, в довге ціле (long int) і повертає результат:

long int strtol (const char * string, char ** endptr, int basis);

endptr-даний параметр не використовується, тому він записаний нульовим вказівником.
string-це сі-рядок для виконання перетворення.
basis-заснування системи числення.В даному випадку це 16, так як перетворення в 16ричную систему. Якщо basis дорівнює 16 - префікс 0x або 0X. [34] [35]

Також з веб-сторінки «Стробоскоп» ми можемо отримати два параметри: для зміни яскравості (такий параметр описувався вище в розділі) і також для зміни частоти миготіння(рис. 2.12) :

```
if (request->hasParam("frequency")) {
    int thisfreq=request->getParam("frequency")->value().toInt();
      strip.objStrobo.frequency=round((60000-thisfreq*20)/thisfreq);
      Serial.println(thisfreq);
    }
```

Рисунок 2.12 – Фрагмент коду

Завдяки цьому фрагменту коду ми отримуємо параметр frequency для зміни частоти миготіння. Цей параметр приходить у вигляді рядка, за допомогою методу toInt () перетворюємо рядок в ціле число і записуємо в змінну frequency з класу Strobo попередньо обробляючи дані. Для перетворення з bpm (beats per minute) в мілісекунди я використовувала функцію round, яка округлює в найближчу сторону і також математичні розрахунки:

round ((60000-thisfreq * 20) / thisfreq);

В кінці виводимо частоту в послідовний порт для налагодження:

Serial.println (thisfreq);

server.begin (); // В кінці setup використовуємо метод begin () на об'єкті «server», щоб запустити сервер і він почав прослуховувати вхідних клієнтів.

3 СТВОРЕННЯ ВЕБ-СТОРІНОК

Абревіатура HTML розшифровується як «HyperText Markup Language» («мова гіпертекстової розмітки») і це найпоширеніша мова розмітки для створення веб-сторінок. Веб-браузери вміють читати HTML-файли та HTML-теги повідомляють веб-браузеру, як він повинен показати контент на веб-сторінці.

3.1 Базові теги HTML

Першим рядком будь-якого HTML-документа завжди є <! DOCTYPE html>. Він каже браузеру, що цей документ - HTML-файл.

Структура веб-сторінки повинна перебувати між тегами <html> і </ html>. Тег <html> вказує на початок веб-сторінки, а </ html> - на її кінець.

HTML-документ складається з двох головних частин: заголовка і тіла. Заголовок позначається тегами <head> і </ head>, а тіло - тегами <body> і </ body>.

У заголовку знаходяться дані про HTML-документ, які кінцевий користувач безпосередньо бачити не буде, але які додають в веб-сторінку різний функціонал на кшталт заголовка документа, скриптів, стилів і т.д. - це називається метаданими. У тілі є вміст веб-сторінки, як заголовки сторінки, тексту, кнопок, таблиць і т.д.

3.1.1 Тег title (заголовок HTML-документа)

Цим тегом задається текст, що показується у вкладці веб-браузера. Цей текст повинен розташовуватися між тегами <title> і </ title>, які в свою чергу повинні знаходитися між тегами <head> і </ head>.

3.1.2 Теги h1, h2 і т.д. (Заголовки тексту в документі)

Ці теги використовуються для того, щоб структурувати текст на веб-сторінці. Тег заголовка починається з букви «h», після якої йде число, що позначає важливість заголовка. Наприклад, теги <h1> і </ h1> використовуються для заголовка 1 рівня (це найважливіший заголовок), теги <h2> і </ h2> - для заголовка 2 рівня (це другий за важливістю заголовок) і т.д. до заголовка 6 рівня. Ці теги текстових заголовків повинні бути між тегами <body> і </ body>.

3.1.3 Тег p (абзац)

Цей тег використовується для структурування тексту. Кожен абзац повинен знаходитися між тегами <p> і </ p>.

3.1.4 Тег a (гіперпосилання)

HTML-посилання називають «гіперпосиланнями». Їх можна додати в текст, зображення, кнопки і багато інших HTML-елементів. Для додавання гіперпосилання використовуються теги <a> і . Ось так:

 element

Між тегами <a> і поміщається HTML-елемент, до якого потрібно додати гіперпосилання.

Атрибут «href» вказує на те, куди повинно вести це посилання.

3.1.5 Атрибут «class»

Цей атрибут потрібен для того, щоб задати для HTML-елемента одне або більше імен класу. Вони використовуються, наприклад, щоб задати CSS-стилі. Наприклад, щоб задати стиль для групи HTML-елементів, ми можемо перед цим поставити для всіх них одне і теж ім'я класу.

Щоб привласнити HTML-елементу ім'я класу, знадобиться наступний синтаксис:

<Element class = "classname">

3.1.6 Тег <meta> (метадані)

У заголовку файлу HTML також часто використовують тег <meta>, вміст якого адаптує веб-сторінку під будь-який веб-браузер Наступний тег <meta> змушує веб-сторінку реагувати в будь-якому браузері (ноутбуці, планшеті або смартфоні):

<Meta name = "viewport" content = "width = device-width, initial-scale = 1">

У тезі <meta> задаються метадані про HTML-документ. На самій сторінці вони не показуються, але надають браузеру корисну інформацію про те, як відображати веб-контент.

Також можна використовувати наступний рядок:

<Link rel = "icon" href = "data:,">

Він потрібен, щоб браузер не робив до ESP32 запитів про іконку сайту. В тому випадку коли немає значка. Значок - це значок веб-сайту, який відображається поруч із заголовком на вкладці веб-браузера. Якщо такого рядка не буде, то ESP32 буде отримувати запит значка кожен раз при зверненні до веб-сервера[36].

3.2 CSS стилі

Абревіатура CSS означає «каскадна таблиця стилів» - це мова, яка використовується для опису того, як HTML-елементи повинні виглядати на веб-сторінці. З його допомогою можна описати певну частину сторінки на кшталт окремого тега або групи тегів. CSS-код можна додати в HTML-файл або в окремий файл, на який буде посилатися HTML-файл. У своєму коді я додала CSS-код безпосередньо в HTML-файл для зручності подальшої роботи в Arduino IDE.

CSS-код в HTML-файлі повинен знаходитися між тегами <style> і </ style>, які в свою чергу повинні знаходитися в заголовку HTML-документа.

У мові CSS використовуються так звані «селектори». Вони вказують на HTML-елемент, для якого потрібно задати певний стиль. У селектора є властивості, а у властивостей - значення. Кожному селектору можна задати багато властивостей. Ось так виглядає синтаксис:

```
selector {
 property: value;
}
```

Також існує універсальний селектор. Його використовують у випадках, коли потрібно встановити одночасно один стиль для всіх елементів веб-сторінки, наприклад, задати шрифт або накреслення тексту. Для позначення

універсального селектора застосовується символ зірочки (*) і в даному випадку ми прибираємо відступи і поля для всіх елементів[37][38] (рис. 3.1) :

```
*{
  margin: 0;
  padding: 0;
}
```

Рисунок 3.1 – Фрагмент коду

3.3 Структура моєї програми для створення веб-сторінок

Для зберігання веб-сторінки я створила змінну, яка буде зберігати багатостроковий текст в такому вигляді:

char index_html [] = R "(HTML CODE)";

R-raw параметр зберігання необробленого тексту і читання його як зрозумілий string текст для C ++. Без R необхідно було б писати весь текст в один рядок уникаючи додаткових "" і зворотних слешів \ t, \ n та іншого.

Коренева сторінка з налаштуваннями яскравості має назву «option».

У <head> ми записуємо такі налаштування:

Вказуємо кодування документа. В даному випадку-це UTF-8:

<Meta charset = "UTF-8">

Також потрібно задати тег <meta>, вміст якого адаптує веб-сторінку під будь-який веб-браузер:

<Meta name = "viewport" content = "width = device-width, initial-scale = 1.0">

Властивість width визначає розмір вікна перегляду. Він може бути встановлений на певну кількість пікселів, наприклад, width = 600 або на спеціальне значення device-width, яке означає ширину екрану в пікселях CSS в масштабі 100%, тобто вказує браузеру, що потрібна ширина області перегляду така ж, як і ширина екрану

(Є також відповідні значення height і device-height, які можуть бути корисні для сторінок з елементами, які змінюють розмір або положення на основі висоти вікна перегляду).

Властивість initial-scale контролює рівень масштабування при першому завантаженні сторінки. Властивості maximum-scale, minimum-scale і user-scalable визначають, як користувачам дозволено змінювати масштабування. initial-scale = 1.0 - встановлює початковий масштаб сторінки. В даному випадку 1, тобто 100%[39] [40].

Для того, щоб підключити нестандартний шрифт, потрібно додати в код такий рядок:

<Link href = "https://fonts.googleapis.com/css?family=Comfortaa|Oswald|Russo+One&display=swap" rel = "stylesheet">

Тег link встановлює зв'язок із зовнішнім документом, наприклад файлом зі стилями або з шрифтами. На відміну від тега <a>, тег <link> розміщується завжди всередині контейнера <head> і не створює посилання. Атрибут href вказує шлях до зовнішнього файлу, атрибут rel визначає відносини між поточним документом і файлом, на який робиться посилання. В даному випадку цей рядок вказує на те, що цей документ підключає шрифти[41].

3.3.1 Підключення стилів

Також потрібно підключити до нашої сторінки CSS стилі. Є 3 способу підключення.

3.3.1.1 Варіант 1 - Глобальний CSS

Глобальний CSS поміщається в контейнер <head> конкретної сторінки. При такому варіанті підключення класи і ідентифікатори (ID) можуть бути використані для звернення до CSS коду, проте, вони будуть активні тільки на цій конкретній сторінці. CSS стилі підключені таким шляхом завантажуються при кожній повторному завантаженні сторінки, тому вони можуть вплинути на швидкість її завантаження.

Переваги глобальних CSS:

- таблиця стилів впливає тільки на одну сторінку;
- у глобальному CSS можуть бути використані класи і ідентифікатори (ID);
- немає необхідності завантажувати декілька файлів. HTML і CSS можуть бути в одному і тому ж файлі.

Недоліки глобальних CSS:

- збільшений час завантаження сторінки;
- підключається тільки до однієї сторінки - неефективно, якщо треба використовувати один і той же CSS для декількох сторінок.

Документ HTML з глобальним CSS повинен виглядати приблизно так(рис. 3.2) :

```
<!DOCTYPE html>
<html>
<head>
<style>
body {
    background-color: blue;
}
h1 {
    color: red;
    padding: 60px;
}
</style>
</head>
<body>

<h1>Пример</h1>
<p>Сам текст</p>

</body>
</html>
```

Рисунок 3.2 – Фрагмент коду

3.3.1.2 Варіант 2 - Зовнішній CSS

В цьому випадку всі зміни, зроблені в зовнішньому CSS файлі, будуть відображатися на сайті. Посилання на зовнішній CSS файл поміщається в контейнер <head> сторінки:

<Head>

<Link rel = "stylesheet" type = "text / css" href = "style.css" />

</ Head>

Переваги зовнішніх CSS:

- менший розмір сторінки HTML і чистіша структура файлу;
- велика швидкість завантаження;

– для різних сторінок може бути використаний один і той же .css файл.

Недоліки зовнішніх CSS:

– сторінка може некоректно відображатися до повного завантаження зовнішнього CSS.

3.3.1.3 Варіант 3 - Внутрішній CSS

Внутрішній CSS використовується для конкретного тега HTML. Атрибут <style> використовується для налаштування цього тега. Цей варіант підключення CSS не найпопулярніший, так як в цьому випадку необхідно налаштовувати кожен тег HTML окремо. До того ж управління сайтом може стати досить важким, якщо використовувати тільки внутрішній CSS. Однак в деяких випадках цей спосіб може бути вельми корисним. Наприклад, в разі якщо немає доступу до CSS файлів, або необхідно застосувати правила тільки для одного елемента. Приклад HTML сторінки з внутрішнім CSS повинен виглядати так:

```
<! DOCTYPE html>
<Html>
<Body style = "background-color: black;">
<H1 style = "color: white; padding: 30px;"> Приклад 1 </ h1>
<P style = "color: white;"> Приклад 2 </ p>
</ Body>
</ Html>
```

Переваги внутрішнього CSS:

– корисний для перевірки і попереднього перегляду змін;

– корисний для швидких виправлень;

– менше HTTP запитів.

Недоліки внутрішнього CSS:

– внутрішній CSS повинен бути застосований для кожного елементу окремо[42].

У моєму коді я буду використовувати глобальний CSS всередені тегу <head> для кожної веб-сторінки окремо.

Після правил CSS також всередені тегу <head> потрібно прописати код для отримання параметра яскравості(рис. 3.3) :

```
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>
<script>


            function bri(pos) {
            $.get("/option.html?brightnes=" + pos);
           }

</script>
```

Рисунок 3.3 – Фрагмент коду

У цьому фрагменті коду я використовувала jQuery.

jQuery - це швидка, легка і багатофункціональна JavaScript бібліотека. Зменшення обсягу JavaScript коду за допомогою бібліотеки jQuery здійснюється за рахунок мінімізації. Мінімізація - це процес, який полягає у видаленні з вихідного коду всього зайвого (коментарів, незначущих прогалин, переносів рядків, символів табуляції) і заміни імен функцій і змінних на більш короткі.

Підключення jQuery на сторінку здійснюється також, як і будь-якого іншого JavaScript файлу. Тобто за допомогою додавання в HTML тега <script> з атрибутом src, в якому необхідно задати повний або відносний шлях до файлу.

CDN (Content Delivery Network) - це технологія, яка дозволяє збільшити швидкість доставки контенту кінцевим користувачам.

Складається вона з великої кількості серверів, географічно розташованих в різних точках світу, на кожному з яких розташовується кеш контенту. При

цьому його доставка кінцевому користувачеві здійснюється зазвичай з того сервера, який ближче інших розташований до нього. В результаті чого скорочується час його завантаження, прискорюється відгук, збільшується продуктивність сайту, а також знижується навантаження на оригінальний сервер.

Тобто CDN надає ще один спосіб підключити бібліотеку jQuery. При цьому безпосередньо завантажувати його собі на сервер не потрібно, він буде братися з CDN.

Завантаження jQuery з CDN надають безліч компаній, наприклад, таких як Google, Microsoft, Cloudflare, jQuery, Yandex і т.д.

Підключити jQuery з CDN дуже просто. Для цього потрібно вставити script з атрибутом src, в якому прописати шлях до цієї бібліотеки.

Код для онлайн підключення jQuery з Google CDN:

<Script src = "https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"> </ script>

За допомогою такого рядка коду:

function bri (pos) {

$ .get ("/ option.html? Brightnes =" + pos);

}

ми записуємо в параметр brightnes, за допомогою якого в подальшому коді буде змінюватися яскравість, значення pos (в цей параметр при зміні положення повзунка записалося нове значення яскравості) .Тобто ми тут вказуємо що нам поверне сервер, коли зміниться значення повзунка.Сервер нам поверне параметр brightnes з новим значенням яскравості.

jQuery функція $ .get () дозволяє завантажити дані з сервера за допомогою HTTP запиту методом GET.Важно позначити, що функція $ .get () є скороченою версією функції $ .ajax ()[43].

Завдяки використанню ajax-запиту здійснюється запит до сервера методом GET без перезавантаження сторінки.

Ajax-підхід до побудови призначених для користувача інтерфейсів веб-додатків, що полягає в «фоновому» обміні даними браузера з веб-сервером. В результаті при оновленні даних веб-сторінка не перезавантажується повністю і веб-додатки стають швидше і зручніше.

Порівняння стандартного підходу і AJAX:

У класичній моделі веб-додатки:

- користувач заходить на веб-сторінку і натискає на який-небудь її елемент;
- браузер формує і відправляє запит серверу;
- у відповідь сервер генерує абсолютно нову веб-сторінку і відправляє її браузеру. Після чого браузер повністю перезавантажує всю сторінку.

При використанні AJAX:

- користувач заходить на веб-сторінку і натискає на який-небудь її елемент;
- скрипт (на мові JavaScript) визначає, яка інформація необхідна для оновлення сторінки;
- браузер відправляє відповідний запит на сервер;
- сервер повертає лише ту частину документа, на яку прийшов запит;
- скрипт вносить зміни з урахуванням отриманої інформації (без повного перезавантаження сторінки)[44].

Далі йде тег <body> і перше що в ньому знаходиться-це назва видів режимів, яке знаходиться в окремому контейнері і в переліку, що складається з посилань на інші веб-сторінки(рис. 3.4) :

```
<div class="container">
  <ul>
    <a href="option.html" class="current"><li>Яркость</li></a>
    <a href="custom.html"><li>Встроенные режимы</li></a>
    <a href="create.html"><li>Изменяемые режимы</li></a>
    <a href="strobo.html"><li>Стробоскоп</li></a>
  </ul>
</div>
```

Рисунок 3.4 – Фрагмент коду

Сторінка, на якій в даний момент знаходиться користувач поміщається в клас current (поточний). Це правило діє і для всіх інших веб-сторінок.

Основний контент сторінки знаходиться в окремому контейнері всередені тегу <form>:

<form>

<Input type = "range" name = "brightnes" value = "" class = "range" min = "0" max = "250" onchange = "bri (this.value)">

</form>

Тег <form> встановлює форму на веб-сторінці. Форма призначена для обміну даними між користувачем і сервером. Область застосування форм не обмежена відправкою даних на сервер, за допомогою клієнтських скриптів можна отримати доступ до будь-якого елементу форми, змінювати його і застосовувати на власний розсуд.

Документ може містити будь-яку кількість форм, але одночасно на сервер може бути відправлена тільки одна форма. З цієї причини дані форм повинні бути незалежні один від одного.

Коли форма відправляється на сервер, управління даними передається програмі. Попередньо браузер готує інформацію у вигляді пари «ім'я =

значення», де ім'я визначається атрибутом name тега <input>, а значення введено користувачем або встановлено в поле форми за замовчуванням.

Тег <input> є одним з різнобічних елементів форми і дозволяє створювати різні елементи інтерфейсу і забезпечити взаємодію з користувачем. Головним чином <input> призначений для створення текстових полів, різних кнопок, перемикачів і прапорців. Хоча елемент <input> не потрібно поміщати всередину контейнера <form>, що визначає форму, але якщо введені користувачем дані повинні бути відправлені на сервер, де їх обробляє серверна програма, то вказувати <form> обов'язково. Те ж саме відбувається і в разі обробки даних за допомогою клієнтських додатків, наприклад, скриптів на мові JavaScript.

Допускається всередину контейнера <form> поміщати інші теги, при цьому сама форма ніяк не відображається на веб-сторінці, видно тільки її елементи і результати вкладених тегів[45][46]

Для управління яскравістю всередені тегу input використовується type = "range" (повзунок).

Повзунок призначений для введення чисел у вказаному діапазоні. Тут min - мінімальне число в діапазоні (0), max - максимальне число (250), step - крок зміни чисел (за замовчуванням 1), value - початкове значення. За замовчуванням value обчислюється як середнє арифметичне між min і max. Самі повзунки рідко застосовуються в «чистому» вигляді, оскільки не забезпечують необхідну зворотний зв'язок з користувачем, а ось в поєднанні з JavaScript це стає потужним і зручним елементом інтерфейсу[47] [48].

У цьому фрагменті коду при управлінні повзунком спрацьовує подія onchange. Особливість даної події в тому, що викликається вона після того, як користувач відпустив повзунок. Подія не дозволяє відстежити всі маніпуляції, які робить користувач, поки обирає потрібне значення. Коли користувач

пересуне повзунок буде викликана функція bri, нове значення яскравості запишеться в параметр pos і далі буде оброблено в функції [49].

3.3.2 Вбудовані режими

Також, як і на інших веб-сторінках, на вбудованих режимах є перелік з назвами всіх веб-сторінок. Всі блоки для вибору режиму знаходяться в головному контейнері з класом container, а кожен конкретний режим перебуває у власному контейнері (рис. 3.5) :

```
<div class="container">
  <h1>Встроенные режимы</h1>
<div class="bloc_led">
      <h2>Радуга</h2>
    <a href="?mode=1"><div class="button">Выбрать</div></a>
</div>
<div class="bloc_led">
        <h2>Смена цветов всей ленты</h2>
      <a href="?mode=2"><div class="button">Выбрать</div></a>
</div>
<div class="bloc_led">
        <h2>Появление случайных цветов</h2>
      <a href="?mode=3"><div class="button">Выбрать</div></a>
</div>
<div class="bloc_led">
        <h2>RGB пропеллер</h2>
      <a href="?mode=4"><div class="button">Выбрать</div></a>
</div>
<div class="bloc_led">
        <h2>Случайные вспышки</h2>
      <a href="?mode=5"><div class="button">Выбрать</div></a>
</div>

</div>
```

Рисунок 3.5 – Фрагмент коду

Це рядок для створення кнопки, яка передає параметр mode (режим) і має клас button:

 Вибрати </ div>

В даному випадку тег <a>, який призначений для створення посилань, використовується для передачі параметра.

3.3.2 Змінні режими

Для вибору режиму серед змінних режимів використовується такий код:

<Label> <input type = "radio" name = "mode" value = "1" class = "radio" checked> Бігаючий вперед </ label>

Основний контент сторінки знаходиться всередені тегу <form>.

<Input> елементи типу radio зазвичай використовуються в радіогрупах - наборах радіокнопок, що описують набір пов'язаних опцій. Одночасно можна вибрати тільки один перемикач в даній групі. Радіокнопки зазвичай відображаються у вигляді маленьких кружечків, які при виборі заповнюються або підсвічуються.

Радіогрупа визначається призначенням однакових радіокнопок в групі name. Як тільки радіогрупа встановлена, вибір будь-якої радіокнопки в цій групі автоматично скасовує вибір будь-якої обраної радіокнопки в тій же групі.

name -ім'я групи перемикачів для ідентифікації поля. Оскільки перемикачі є груповими елементами, то ім'я у всіх елементів групи має бути однаковим.

value -задає, яке значення буде відправлено на сервер. Тут кожен елемент повинен мати своє унікальне значення, щоб можна було ідентифікувати, який пункт був обраний користувачем[50][51].

Також тут використовується <label> - цей елемент являє собою заголовок для елемента в призначеному для користувача інтерфейсі. Текст мітки не тільки візуально пов'язаний з відповідним введенням тексту, а й програмно.

В HTML зв'язування label з input виконується одним з 2х способів:

1. За допомогою розміщення елемента input в label:

<Label> <input type = "radio" name = "mode" value = "1" class = "radio"> В середину </ label> <

2. За допомогою задання елементу input атрибута id, а label - for з таким самим значенням як у id.

<Input type = "radio" id = "first" name = "mode" value = "1" class = "radio">

<Label for = "first"> В середину </ label> [52][53]

Також тут є елемент типу color для вибору кольору світлодіодів(рис. 3.6) :

```
<div class="container">
  <div class="bloc_led">
  <h2>Цвет</h2>
  <div id="ColorPalet">
    <input type="color" name="color" value="#ffffff" class="color">
  </div>
</div>
</div>
```

Рисунок 3.6 – Фрагмент коду

<Input> елементи типу color надають елемент інтерфейсу для користувача , який дозволяє користувачеві вказати колір або за допомогою візуального інтерфейсу вибору кольору, або шляхом введення кольору в текстове поле в шістнадцятковому форматі #rrggbb. Дозволені лише прості кольори (без альфа-каналу), хоча кольори у CSS мають більше форматів, наприклад назви кольорів, функціональні позначення і шістнадцятковий формат з альфа-каналом.

Подання елемента може істотно відрізнятися від одного браузера і / або платформи до іншої - це може бути просте текстове введення, яке автоматично

перевіряє правильність введення інформації про колір у відповідному форматі, або стандартний для платформи вибір кольору, або якесь призначене для користувача вікно вибору кольору.

value елемента <input> типу color містить 7-символьний рядок, що задає колір RGB в шістнадцятковому форматі. Хоча можна ввести колір у верхньому або нижньому регістрі, він буде збережений у вигляді нижнього регістра. Value ніколи не буває в будь-якій іншій формі і ніколи не буває порожнім. Value встановлює значення за замовчуванням, колірна заливка буде попередньо заповнена кольором за замовчуванням (в даному випадку це буде білий колір).

Для цього елемента використовується ідентифікатор name = "color"[54].

Для відправки даних на сервер призначена спеціальна кнопка Submit. Атрибут name для цього типу кнопки можна не писати. Якщо не вказати значення value, то браузер автоматично додасть текст, він різниться в залежності від браузера:

<Input type = "submit" name = "" class = "submit" value = "Застосувати"> [55]

3.3.3 Стробоскоп

Сторінка стробоскоп розділена на 4 блокових елемента-контейнера :

- контейнер з назвами видів режимів;
- для регулювання яскравості;
- для регулювання частоти;
- для вибору кольору.

Для управління частотою миготіння всередені тегу input використовується type = "range" (повзунок):

<Input type = "range" name = "Frequency" value = "120" class = "range" min = "20" max = "200" onchange = "frequency (this.value)">

<Span id = "bpm"> bpm: 120 </ span>

Тут min - мінімальне число в діапазоні (20), max - максимальне число (200), step - крок зміни чисел (за замовчуванням 1), value (початкове значення) - 120.

Коли користувач пересуне повзунок буде викликана функція frequency, нове значення частоти запишеться в параметр frequency і далі буде оброблено в функції.

Тег призначений для визначення рядкових елементів документа. Він позначає «просто текстовий блок» і в ньому записаний текст «bpm» (що означає одиницю виміру для частоти миготіння) з id = "bpm"[56][57].

```
<script>
        function bri(pos) {
        $.get("/strobo.html?brightnes=" + pos);
        }

         function frequency(pos) {
        $.get("/strobo.html?frequency=" + pos);
         document.getElementById("bpm").innerHTML="bpm: "+pos;
        }
</script>
```

За допомогою такого коду ми обробляємо параметр яскравості аналогічно з тим, як описано раніше у тексті і також обробляємо параметр частоти миготіння.Сервер нам поверне параметр frequency з новим значенням частоти.

```
document.getElementById ( "bpm"). innerHTML = "bpm:" + pos;
```

Цей фрагмент коду звертається по id до рядка bpm: 120 і за допомогою властивості innerHTML змінює 120, яке стоїть за умовчанням для частоти миготіння при запуску сервера, на поточне значення частоти.

Завдяки медіа-запитами мої веб-сторінки адаптивні і можуть підлаштовувати інтрефейс сторінки під розмір екрану або вкладки.

На рис. 3.7 показаний вигляд сторінки при розмірі більше 890 пікселів.

На рис. 3.8 показаний вигляд сторінки при розмірі менше 890 пікселів.

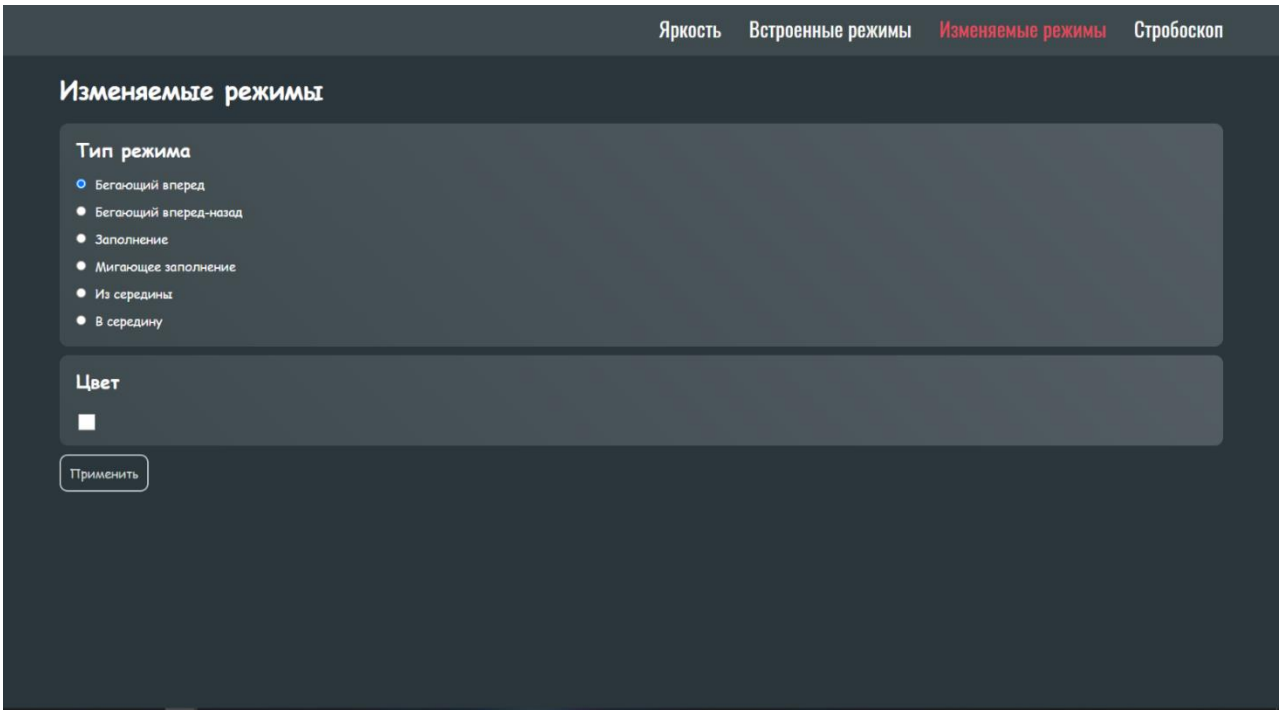

Рисунок 3.7 – Скріншот сайту

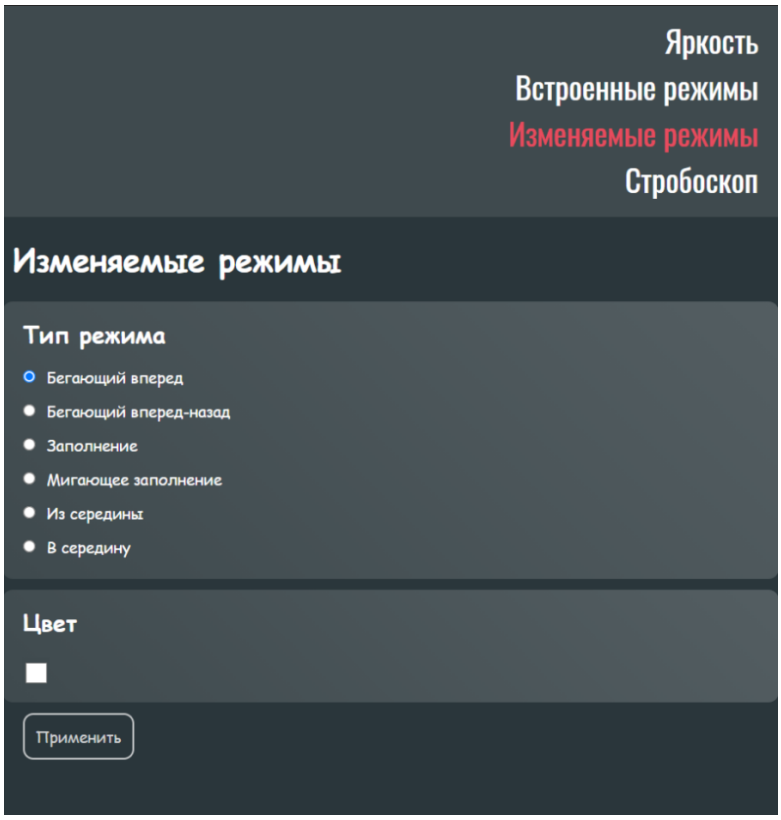

Рисунок 3.8 – Скріншот сайту

Ось так виглядає мій код з медіа-запитами(рис. 3.9) :

```
@media (max-width: 1400px) {
  h1{
    padding-left: 10px;
  }
  nav{
    padding-right: 20px;
  }
  .submit{
    margin-left:20px;
  }
}


@media (max-width: 890px){
  nav li{
  display: block;
  font-size: 30px;
  margin-left: 30px;
  }
  }
```

Рисунок 3.9 – Фрагмент коду

ВИСНОВКИ

Даний проект вирішує ряд проблем у сфері управління світловими ефектами та освітленням.

Управління по бездротовому інтерфейсу Wi-Fi дає користувачеві більше можливостей для реалізації творчих ідей. Віддалений доступ з досить великим радіусом дії дає можливість розмістити стрічку в будь-якому місці свого будинку або будь-якого приміщення, що дає свободу для дизайнерських рішень в інтер'єрі. Клієнт завжди буде мати швидкий доступ до сервера з будь-якого телефону або ПК. Якщо допрацювати та поліпшити конструкцію можна розширити область її застосування . Якщо зробити захисний корпус для плати і силіконову оболонку для стрічки, то можна буде розміщувати стрічку із зовнішнього боку будівлі або ж на високих вуличних арках для прикраси і освітлення вулиць. Також можна додати режим еквалайзеру, що буде особливо актуально для місць, де грає музика, наприклад, в нічних клубах, кафе або магазинах. До того ж, можна додати варіант живлення від акумуляторів, в разі, якщо поруч не буде розеток.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. ESPAsyncWebServer [Електронний ресурс] — Режим доступу: https://github.com/me-no-dev/ESPAsyncWebServer
2. Сравнение персональных протоколов беспроводной сети [Електронний ресурс] — Режим доступу: http://digitrode.ru/articles/1355-sravnenie-personalnyh-protokolov-besprovodnoy-seti-wi-fi-bluetooth-zigbee.html#sel=
3. Wi-Fi и Bluetooth [Електронний ресурс] — Режим доступу: http://www.jaans.ru/wifi-bluetooth-rasnitsa.html
4. Управление светодиодной лентой через Bluetooth[Електронний ресурс] — Режим доступу: https://arduinoplus.ru/upravlyaem-svetodiodnoi-lentoi-cherez-bluetooth/
5. Адресная светодиодная лента Ардуино [Електронний ресурс] — Режим доступу: https://xn--18-6kcdusowgbt1a4b.xn--p1ai/ws2812b-%D0%B0%D1%80%D0%B4%D1%83%D0%B8%D0%BD%D0%BE/
6. Адресная светодиодная лента [Електронний ресурс] — Режим доступу: https://electroinfo.net/poluprovodniki/chto-takoe-adresnaja-svetodiodnaja-lenta.html#i-5
7. Кто быстрее, FastLED или Adafruit NeoPixel? [Електронний ресурс] — Режим доступу: https://alexgyver.ru/fastled_vs_neopixel/
8. FastLED [Електронний ресурс] — Режим доступу: https://github.com/FastLED/FastLED
9. FastLED overview [Електронний ресурс] — Режим доступу: https://github.com/FastLED/FastLED/wiki/Overview
10. FastLED [Електронний ресурс] — Режим доступу: http://fastled.io/
11. High performance math [Електронний ресурс] — Режим доступу: https://github.com/FastLED/FastLED/wiki/High-performance-math#saturating
12. Pixel reference [Електронний ресурс] — Режим доступу: https://github.com/FastLED/FastLED/wiki/Pixel-reference#colors
13. Random() [Електронний ресурс] — Режим доступу: http://arduino.ru/Reference/Random

14. HTTP [Електронний ресурс] — Режим доступу: ru.wikipedia.org/wiki/HTTP
15. Методы HTTP запроса [Електронний ресурс] — Режим доступу: https: //qastart.by/class-2/21-metody-http-zaprosa
16. Методы HTTP запроса [Електронний ресурс] — Режим доступу: https://zametkinapolyah.ru/servera-i-protokoly/tema-7-opredelenie-metodov-http-http-method-definitions-metody-http-zaprosov.html
17. Руководство по HTTP и REST[Електронний ресурс] — Режим доступу: https://code.tutsplus.com/ru/tutorials/a-beginners-guide-to-http-and-rest--net-16340
18. HTTP [Електронний ресурс] — Режим доступу: https://ru.wikipedia.org/wiki/HTTP
19. WiFi [Електронний ресурс] — Режим доступу: https://doc.arduino.ua/ru/prog/WiFi
20. ESP8266WiFi library [Електронний ресурс] — Режим доступу: https://arduino-esp8266.readthedocs.io/en/latest/esp8266wifi/readme.html
21. Arduino:Библиотеки/WiFi [Електронний ресурс] — Режим доступу: http://wikihandbk.com/wiki/Arduino:%D0%91%D0%B8%D0%B1%D0%BB%D0%B8%D0%BE%D1%82%D0%B5%D0%BA%D0%B8/WiFi
22. ESP32 Useful Wi-Fi Library Functions (Arduino IDE) [Електронний ресурс] — Режим доступу: https://randomnerdtutorials.com/esp32-useful-wi-fi-functions-arduino/
23. Настройка домашнего Wi-Fi [Електронний ресурс] — Режим доступу: https://meduza.io/slides/kak-sdelat-chtoby-vse-letalo-vybrat-5-ggts-ili-2-4-ggts-pomogite-nastroit-domashniy-wi-fi
24. Как улучшить работу домашнего Wi-Fi [Електронний ресурс] — Режим доступу: https://3dnews.ru/975001
25. [Електронний ресурс] — Режим доступу: https://weblance.com.ua/143-wi-fi-v-massy-osnovnye-ponyatiya-i-nastroyki-dlya-nachinayuschih.html
26. Настройка Wi-Fi [Електронний ресурс] — Режим доступу: https://help-wifi.com/sovety-po-nastrojke/kak-najti-svobodnyj-wi-fi-kanal-i-smenit-kanal-na-routere/
27. Каналы Wi-Fi [Електронний ресурс] — Режим доступу: https://wifigid.ru/poleznoe-i-interesnoe/shirina-kanala
28. Мощность сигнала и зона покрытия беспроводной сети Wi-Fi [Електронний ресурс] — Режим доступу: https://cutt.ly/entQGmJ

29. ESP8266WiFi library[Електронний ресурс] — Режим доступу: https://arduino-esp8266.readthedocs.io/en/latest/esp8266wifi/readme.html
30. ESPAsyncWebServer [Електронний ресурс] — Режим доступу: https://github.com/me-no-dev/ESPAsyncWebServer
31. AsyncTCP [Електронний ресурс] — Режим доступу: https://github.com/me-no-dev/AsyncTCP
32. ESP32:Примеры[Електронний ресурс] — Режим доступу: https://cutt.ly/entQAGu
33. ESPAsyncWebServer [Електронний ресурс] — Режим доступу: https://github.com/me-no-dev/ESPAsyncWebServer/blob/master/README.md
34. [Електронний ресурс] — Режим доступу: http://cppstudio.com/post/816/
35. [Електронний ресурс] — Режим доступу: http://www.c-cpp.ru/content/strtol-strtoul
36. ESP32 Async Web Server – Control Outputs with Arduino IDE [Електронний ресурс] — Режим доступу: https://randomnerdtutorials.com/esp32-async-web-server-espasyncwebserver-library/
37. ESP32:Примеры[Електронний ресурс] — Режим доступу: https://cutt.ly/5ntQKbq
38. Universal selector [Електронний ресурс] — Режим доступу: http://htmlbook.ru/css/selector/universal
39. Meta tag [Електронний ресурс] — Режим доступу: https://developer.mozilla.org/ru/docs/Mozilla/Mobile/Viewport_meta_tag
40. Habr QaA[Електронний ресурс] — Режим доступу: https://qna.habr.com/q/568278
41. Link [Електронний ресурс] — Режим доступу: http://htmlbook.ru/html/link
42. Подключение css стилей [Електронний ресурс] — Режим доступу: https://www.hostinger.com.ua/rukovodstva/podklyucheniye-css-stiley
43. Jquery method get [Електронний ресурс] — Режим доступу: https://basicweb.ru/jquery/jquery_method_get.php
44. AJAX [Електронний ресурс] — Режим доступу: https://ru.wikipedia.org/wiki/AJAX
45. Html form [Електронний ресурс] — Режим доступу: http://htmlbook.ru/html/form

46. Html input [Електронний ресурс] — Режим доступу: http://htmlbook.ru/html/input
47. Input type range[Електронний ресурс] — Режим доступу: https://shra.ru/2017/07/obrabotka-izmenenijj-v-input-typerange/
48. Ползунок [Електронний ресурс] — Режим доступу: http://htmlbook.ru/samhtml5/formy/polzunok
49. Onchange [Електронний ресурс] — Режим доступу: http://htmlbook.ru/html/attr/onchange
50. Переключатели[Електронний ресурс] — Режим доступу: http://htmlbook.ru/samhtml5/formy/pereklyuchateli
51. Input radio[Електронний ресурс] — Режим доступу: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/input/radio
52. Input radio[Електронний ресурс] — Режим доступу: https://itchief.ru/html-and-css/styling-checkbox-and-radio
53. Element label [Електронний ресурс] — Режим доступу: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/label
54. Input color [Електронний ресурс] — Режим доступу: https://developer.mozilla.org/ru/docs/Web/HTML/Element/Input/color
55. Кнопки в формах [Електронний ресурс] — Режим доступу: http://htmlbook.ru/samhtml5/formy/knopki
56. Главный текстовый тег — span [Електронний ресурс] — Режим доступу: https://htmlacademy.ru/courses/43/run/1?utm_source=google&utm_medium=cpc&utm_campaign=wrld_dsa&keyword=&gclid=CjwKCAjwv_iEBhASEiwARoemvDQ_04MAAQYDtEAWzn4e_d1WySwu2iGu54b3x0ZYdqKkqg0Wcu6LaBoCn9UQAvD_BwE
57. Span [Електронний ресурс] — Режим доступу: http://htmlbook.ru/html/span