Хмельницький національний університет
Факультет інформаційних технологій
Кафедра комп'ютерної інженерії та інформаційних систем

КВАЛІФІКАЦІЙНА РОБОТА

бакалавр
Освітній рівень

Програмно-технічні засоби підключення світлових приладів на основі одноплатної комп'ютерної системи Raspberry Pi4
Назва теми

КвРКІ 220043.22.01.51 ПЗ
Шифр

Галузь знань *12 «Інформаційні технології»*
Шифр, назва

Спеціальність *123 «Комп'ютерна інженерія»*
Шифр, назва

Освітня програма *«Комп'ютерна інженерія та програмування»*
Назва

Виконав: студент IV курсу, група КІ2с-22-2 _____ Анастасія ЧАЙКОВСЬКА
Підпис Ініціали, прізвище

Керівник _____ Сергій ЛИСЕНКО
Підпис, дата Ініціали, прізвище

Нормоконтролер _____ Тетяна КИСІЛЬ
Підпис, дата Ініціали, прізвище

До захисту допускаю:
зав. кафедри комп'ютерної інженерії та інформаційних систем _____ Ольга ПАВЛОВА
Підпис Ініціали, прізвище

«02» червня 2025 р.

Хмельницький 2025

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет Інформаційних технологій

Кафедра Комп'ютерної інженерії та інформаційних систем

Освітній рівень бакалавр

Галузь знань 12 Інформаційні технології

Спеціальність 123 Комп'ютерна інженерія

Освітня програма «Комп'ютерна інженерія та програмування»

ЗАТВЕРДЖУЮ

Зав. кафедри Ольга ПАВЛОВА

" 10 " 01 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ БАКАЛАВРА

Анастасії ЧАЙКОВСЬКІЙ
Прізвище, ім'я, по батькові студента

1. Тема проекту (роботи) Програмно-технічні засоби підключення світлових приладів на основі одноплатної комп'ютерної системи Raspberry Pi4
Керівник проекту (роботи) Сергій ЛИСЕНКО, д.т.н., проф.
Прізвище, ім'я, по батькові, науковий ступінь, вчене звання

Затверджена наказом ректора університету від 07.02.2025 р. № 23

2. Строк подання студентом проекту (роботи) на кафедру 01.06.2025 р.

3. Вихідні дані до проекту (роботи) Завдання на кваліфікаційну роботу

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити)
Аналіз програмно-технічні засобів реалізації PC-DMX інтерфейсу
Проєктування системи програмно-технічні засоби реалізації PC-DMX інтерфейсу на основі одноплатної комп'ютерної системи raspberry pi
Програмно-апаратна реалізація програмно-технічні засоби реалізації PC-DMX інтерфейсу на основі одноплатної комп'ютерної системи Raspberry pi

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслень)
Схема плати розширення
Плата розширення
Блок-схеми та діаграма класів

6. Консультанти розділів дипломного проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Тетяна КИСІЛЬ, доцент кафедри КІС		
Антиплагіат	Андрій НІЧЕПОРУК, доцент кафедри КІС		

7. Дата видачі завдання «_10_»_____01_____2025 р.

КАЛЕНДАРНИЙ ПЛАН

№з/п	Назва етапів (розділів) дипломного проекту (роботи)	Термін виконання етапів проекту (роботи)	Примітка
1	Вибір напряму дослідження та узгодження тематики кваліфікаційної роботи з керівником	10.01.2025	виконано
2	Ознайомлення з предметною областю; формулювання мети та задач дослідження; визначення об'єкта та предмета дослідження	01.02.2025	виконано
3	Робота над розділом 1 – дослідження предметної області та постановка задачі	01.03.2025	виконано
4	Робота над розділом 2 – вибір компонентів для проєктування системи адаптивного застосування моніторингових елементів розвідувального БПЛА	01.04.2025	виконано
5	Робота над розділом 3 – проєктування системи адаптивного застосування моніторингових елементів розвідувального БПЛА	29.04.2025	виконано
6	Оформлення пояснювальної записки згідно вимог	25.05.2025	виконано
7	Попередній захист ВКР	26.05.2025	виконано
8	Захист ВКР на засіданні ЕК	Червень 2025 року	

Студент _____ Анастасія ЧАЙКОВСЬКА
Підпис Ініціали, прізвище

Керівник роботи _____ Сергій ЛИСЕНКО
Підпис Ініціали, прізвище

№ рядка	формат	Позначення	Найменування	Кіл. листів	№ екз	Примітка
			Текстові документи			
1		КвРКІ 220043.22.01.51 ПЗ	Пояснювальна записка	59		
			Графічні матеріали			
2		КвРКІ 220043.22.01.51 Е8	Схема плати розширення	1		
3		КвРКІ 220043.22.01.51 Е8	Плата розширення	1		
4		КвРКІ 220043.22.01.51 Е8	Блок-схеми та діаграма	1		
			класів			

Зм	Арк	№ докум	Підпис	Дата	КвРКІ 220043.22.01.51 ВП		
Розробив		Чайковська		02.06.25	Відомість проекту	Літера	Аркуш / Аркушів
Перевір.		Лисенко		02.06.25		У	1 / 1
Н. контр.		Кисіль		02.06.25		ХНУ, КІ2с-22-2	
Затв.		Павлова		02.06.25			

АНОТАЦІЯ

Тема кваліфікаційної роботи: «Програмно-технічні засоби підключення світлових приладів на основі одноплатної комп'ютерної системи Raspberry Pi4».

Автор роботи: Анастасія ЧАЙКОВСЬКА.

Керівник роботи: Сергій ЛИСЕНКО.

Пояснювальна записка: 59 с., 19 рис., 4 табл., 4 дод., 69 джерел.

Графічна частина: 3 креслення.

PC-DMX ІНТЕРФЕЙС, RASPBERRY PI4, ОДНОПЛАТНА КОМП'ЮТЕРНА СИСТЕМА, ПРОГРАМНО-ТЕХНІЧНІ ЗАСОБИ.

Метою дипломної роботи єрозроблення і реалізація програмно-технічні засобів реалізації PC-DMX інтерфейсу на основі одноплатної комп'ютерної системи raspberry PI. Теоретичні дослідження та практична реалізація довели ефективність обраного підходу до обробки і керування інформаційними потоками з використанням цифрових інтерфейсів на основі Raspberry Pi4 та протоколу DMX512.

Об'єктом дослідження є програмно-технічні засоби підключення світлових приладів.

Предметом дослідження є програмно-технічні засоби підключення світлових приладів на основі одноплатної комп'ютерної системи Raspberry Pi4.

Під час проведення даного дослідження був використаний метод систематичного огляду літератури для вивчення і аналізу предметної області даного дослідження з текстових джерел інформації.

Підпис студента

30.05.2025

Дата

ЗМІСТ

ВСТУП

У сучасному світі стрімкий розвиток інформаційних технологій, мультимедійних систем та автоматизованих рішень зумовлює зростаючу потребу в цифровому керуванні світлом. Особливо актуальною ця потреба є в сферах, де освітлення виступає не лише функціональним елементом, а й важливим компонентом дизайну, атмосфери чи взаємодії з користувачем — таких як сцена, виставкові простори, інтерактивні інсталяції, архітектурне підсвічування та інтелектуальні системи освітлення.

Традиційні методи керування освітленням, як правило, базуються на аналогових сигналах або простих механічних перемикачах, що обмежують гнучкість, точність та можливості автоматизації. У відповідь на ці виклики з'являються цифрові протоколи, зокрема DMX512, які дозволяють здійснювати програмоване та сценарне керування великою кількістю світлових каналів. У цьому контексті PC-DMX інтерфейс виступає як ключовий інструмент для реалізації таких можливостей на базі персонального комп'ютера.

Завдяки використанню PC-DMX інтерфейсів забезпечується:

- Прецизійне керування кожним світловим елементом у системі з точністю до 256 рівнів яскравості на канал.

- Можливість створення складних світлових сценаріїв, які змінюються у часі або реагують на зовнішні події (звук, рух, температуру тощо).

- Гнучке налаштування та автоматизація за допомогою спеціалізованого програмного забезпечення (наприклад, QLC+, Freestyler, MagicQ).

- Миттєва адаптація конфігурацій без необхідності фізичної заміни обладнання або зміни кабельної інфраструктури.

- Інтеграція з іншими цифровими системами, такими як аудіо-відео комплекси, сенсори, системи управління «розумним будинком» або IoT-платформи.

Окрім цього, використання персонального комп’ютера як платформи для керування освітленням відкриває нові можливості для інтерактивних інсталяцій, де освітлення реагує на дії користувачів, створюючи нові форми естетичного або функціонального досвіду. Такий підхід дедалі частіше застосовується в музеях, шоурумах, креативних просторах і навіть у побуті.

З огляду на вищезазначене, можна зробити висновок, що потреба в цифровому керуванні світлом засобами PC-DMX обумовлена необхідністю гнучких, інтелектуальних і масштабованих рішень, які відповідають вимогам сучасного мультимедійного, сценічного, архітектурного та побутового середовища..

1 АНАЛІЗ ПРОГРАМНО-ТЕХНІЧНІ ЗАСОБІВ РЕАЛІЗАЦІЇ PC-DMX ІНТЕРФЕЙСУ

1.1 Актуальність застосування PC-DMX інтерфейсу

PC-DMX інтерфейс - це апаратно-програмний засіб, який дозволяє комп'ютеру керувати освітлювальним обладнанням або іншими пристроями за допомогою протоколу DMX512. Цей протокол, який був розроблений для керування сценічним світлом, забезпечує цифрову передачу керуючих сигналів по стандартній лінії зв'язку між керуючим пристроєм (у цьому випадку комп'ютером) і виконавчими пристроями, такими як дим-машини, прожектори, лазери чи інші ефекти [1].

Фізично PC-DMX інтерфейс зазвичай є пристроєм, який підключається до комп'ютера через USB або інший порт (наприклад, Ethernet), і виконує роль апаратного перетворювача між сигналами комп'ютера та сигналами DMX. На програмному рівні комп'ютер використовує спеціальне програмне забезпечення (наприклад, Freestyler, QLC+, DMXControl), яке дозволяє створювати світлові сцени, автоматизувати ефекти й формувати DMX-команди. Сам інтерфейс, у свою чергу, перетворює ці команди в стандартизований цифровий потік даних DMX, який надсилається по одному каналу або декількох каналах до світлових пристроїв.

PC-DMX інтерфейс забезпечує зв'язок між цифровим керуванням на комп'ютері та реальним фізичним керуванням освітлювальними пристроями. Він відіграє ключову роль у системах автоматизованого керування світловими шоу, театральним або концертним освітленням, дозволяючи зручно керувати великою кількістю каналів через єдину цифрову шину, підтримуючи точність, синхронізацію та масштабованість системи [2].

У сучасних умовах цифровізації та автоматизації технологій особливої актуальності набувають системи керування освітленням, які поєднують у собі гнучкість, масштабованість та інтеграцію з іншими цифровими платформами. Одним із ключових компонентів таких систем є PC-DMX інтерфейс, що забезпечує

взаємодію персонального комп'ютера з освітлювальним обладнанням, яке підтримує протокол DMX512 [3].

Протокол DMX512 є загальноприйнятим стандартом у сфері сценічного, архітектурного та декоративного освітлення. Його використання дає змогу точно та надійно керувати великою кількістю світлових приладів. Завдяки PC-DMX інтерфейсам, управління цими пристроями можна здійснювати безпосередньо з комп'ютера, що суттєво розширює можливості користувача: забезпечує підтримку програмного сценарного керування, автоматизацію дій, інтеграцію з мультимедійним контентом тощо.

Актуальність використання PC-DMX інтерфейсу також обумовлена його універсальністю та економічністю. Для реалізації складних світлових постановок більше не потрібно дорогого апаратного забезпечення. Достатньо мати ПК та доступний інтерфейс із відповідним програмним забезпеченням, таким як QLC+, Freestyler, DMXControl тощо. Це відкриває широкі можливості не лише для професійної індустрії, а й для малобюджетних проєктів, аматорських студій, навчальних закладів [4].

PC-DMX інтерфейси знайшли широке застосування в таких сферах:

- театральне та концертне освітлення;
- інсталяції в нічних клубах, галереях, музеях;
- архітектурне підсвічування;
- системи автоматизації освітлення в житлових приміщеннях;
- навчальні лабораторії та наукові дослідження.

Крім того, PC-DMX технології мають високий потенціал для подальшого розвитку завдяки можливості інтеграції з сучасними трендами — такими як Інтернет речей (IoT), штучний інтелект (AI), віртуальна та доповнена реальність (VR/AR). Це дозволяє створювати інтелектуальні системи освітлення, які адаптуються до поведінки користувача або навколишніх умов у реальному часі.

Варто також зазначити, що відкритість протоколів багатьох PC-DMX інтерфейсів надає можливість програмістам і розробникам створювати власні

програмні рішення або розширення для наявних систем. Це особливо цінно в контексті освітніх програм, де студенти можуть отримувати практичний досвід роботи з актуальними технологіями керування.

Застосування PC-DMX інтерфейсу є науково та практично обґрунтованим, сприяє ефективному керуванню освітлювальними системами, знижує витрати на реалізацію світлотехнічних рішень та відкриває широкі перспективи для інновацій у сфері автоматизованого освітлення.

1.2 Універсальність та стандартизація DMX512

Одним із ключових факторів, що обумовлюють широке впровадження PC-DMX інтерфейсів у системи керування освітленням, є підтримка ними стандарту DMX512 (Digital Multiplex 512). Цей протокол було розроблено Американською асоціацією театральної техніки (USITT) у 1986 році для стандартизованого керування сценічним освітленням і ефектами. Сьогодні він є загальновизнаним галузевим стандартом для цифрового керування освітлювальними пристроями, що значно підвищує універсальність та сумісність обладнання різних виробників [5-7].

Протокол DMX512 дозволяє передавати дані по одному кабелю до 512 каналів, кожен з яких відповідає за конкретний параметр (яскравість, колір, положення тощо) певного пристрою. Це дає змогу ефективно управляти великою кількістю приладів у реальному часі, що є критично важливим для динамічних інсталяцій, шоу-програм та інтерактивних систем.

Універсальність PC-DMX інтерфейсу проявляється в таких аспектах:

- Широка сумісність.
- Гнучкість програмного забезпечення.
- Масштабованість.

Більшість сучасних освітлювальних приладів підтримують протокол DMX512, що дозволяє легко інтегрувати інтерфейс у будь-яку систему керування.

Система може легко розширюватися завдяки підтримці кількох DMX-універсів (по 512 каналів у кожному).

Багато PC-DMX інтерфейсів підтримують популярні програми, такі як QLC+, Lightkey, Freestyler, DMXControl тощо, що забезпечує широкі можливості для налаштування сценаріїв та автоматизації [8-10].

Стандартизація протоколу DMX512 має ще одну важливу перевагу - інтероперабельність. Завдяки єдиним правилам передачі даних, користувачі можуть поєднувати обладнання від різних виробників без ризику несумісності пристроїв

Це особливо важливо для прокатних компаній, концертних організаторів і технічних спеціалістів, яким доводиться часто змінювати конфігурації обладнання.

Крім того, у поєднанні з персональним комп'ютером, PC-DMX інтерфейс виступає ефективною платформою для розробки кастомізованих апаратних рішень [11-13].

Завдяки відкритим бібліотекам та API, інженери й програмісти можуть створювати власне програмне забезпечення, яке враховує специфічні вимоги проєкту - від інтерактивних інсталяцій до автоматичних систем освітлення в інтелектуальних будівлях.

Стандартизація DMX512 у поєднанні з універсальністю PC-DMX інтерфейсів сприяє створенню гнучких, масштабованих та ефективних систем цифрового керування освітленням, що відповідають вимогам сучасної технічної та культурної інфраструктури.

PC-DMX інтерфейс виступає як ключовий інструмент для реалізації таких можливостей на базі персонального комп'ютера [14-16].

Переваги цифрового керування засобами PC-DMX:

- Прецизійне керування кожним світловим елементом із точністю до 256 рівнів яскравості на канал.
- Створення складних сценаріїв освітлення, що змінюються у часі або реагують на зовнішні події (звук, рух, температура).

- Автоматизація процесів за допомогою спеціалізованого програмного забезпечення.
- Гнучке налаштування без потреби в апаратних змінах.
- Інтеграція з іншими цифровими системами, такими як аудіо-відео комплекси, системи IoT тощо.

Таблиця 2.1 - Приклади практичного застосування PC-DMX

Сфера застосування	Опис
Театральні постановки	Автоматизоване керування світлом у реальному часі згідно зі сценарієм вистави
Нічні клуби та шоу-програми	Динамічні світлові ефекти, синхронізовані з музикою
Музеї, інтерактивні виставки	Реакція освітлення на рух відвідувача або його дії
Розумне архітектурне підсвічування	Автоматичне регулювання освітлення залежно від часу доби чи погоди
Навчальні лабораторії	Вивчення основ цифрових систем керування та практичне програмування сценаріїв

Використання PC-DMX інтерфейсу для цифрового керування світлом є не лише доцільним, а й необхідним у багатьох сучасних галузях.

Такий підхід сприяє підвищенню ефективності, автоматизації, інтерактивності та гнучкості систем освітлення [17-20].

Таблиця 2.2 - Порівняльна таблиця традиційного та цифрового керування світлом

Ознака	Традиційне керування	Цифрове керування (PC-DMX)
Спосіб керування	Механічні перемикачі, аналогові димери	Програмне керування через ПК
Гнучкість системи	Обмежена	Висока
Кількість контрольованих пристроїв	Невелика	До сотень пристроїв через DMX-канали
Можливість сценарного управління	Відсутня або дуже обмежена	Повноцінне програмування сценаріїв
Можливість автоматизації	Мінімальна	Висока
Інтеграція з іншими системами	Відсутня	Можлива через API та протоколи
Вартість впровадження	Відносно висока (багато кабелів і комутаторів)	Знижується завдяки використанню ПК та USB-DMX інтерфейсів

1.3 Доступність та економічність PC-DMX інтерфейсу

Одним із ключових чинників, що сприяють поширенню технологій цифрового керування освітленням на основі PC-DMX, є їх доступність як у фінансовому, так і у технічному аспектах. У порівнянні з традиційними консолями освітлення або промисловими автоматизованими системами, рішення на основі PC-DMX інтерфейсу мають значно нижчу вартість впровадження та експлуатації, що робить їх привабливими для широкого кола користувачів - від невеликих студій і

навчальних лабораторій до великих сценічних комплексів і архітектурних проєктів [200-22].

Більшість комерційних або DIY PC-DMX інтерфейсів реалізуються у вигляді простих USB–DMX перетворювачів, які коштують у межах від $10 до $100, залежно від функціональності (однонапрямний чи двонапрямний обмін, підтримка кількох універсів, гальванічна розв'язка тощо). Це в десятки разів дешевше за класичні світлові пульти або DMX-консолі, які можуть коштувати від кількох сотень до тисяч доларів.

Системи PC-DMX не вимагають спеціалізованого апаратного середовища - достатньо стандартного персонального комп'ютера або ноутбука з доступним USB-портом. Це суттєво знижує вхідний бар'єр і дозволяє використовувати вже наявну техніку без додаткових витрат [23-25].

Існує велика кількість безкоштовного або умовно-безкоштовного ПЗ для керування DMX через ПК, зокрема:

- QLC+ (Q Light Controller Plus) — потужне, кросплатформне програмне забезпечення з відкритим кодом;

- Freestyler DMX — популярне рішення для Windows з широкими можливостями для шоу-програм;

- DMXControl, MagicQ, Lightkey (для macOS) тощо.

Ці програми підтримують більшість інтерфейсів, дозволяють створювати сценарії, ефекти, таймери, інтегрувати MIDI та OSC сигнали, що значно розширює можливості без додаткових витрат.

На відміну від аналогових систем керування або дорогих DMX-консолей, рішення на основі PC-DMX:

- потребують меншої кількості кабелів, завдяки цифровій шинній архітектурі;

- можуть керувати великою кількістю приладів з одного пристрою;

- легко масштабуються за рахунок підключення додаткових інтерфейсів або розширювачів.

Порівняння витрат на впровадження системи керування світлом подано в таблиці 1.3.

Таблиця 1.3 – Порівняння витрат на впровадження системи керування світлом

Компонент	PC-DMX система	Традиційна DMX-консоль
Апаратна частина керування	USB-DMX інтерфейс — від $20 до $60	Професійна консоль — від $500 до $3000
Комп'ютер для керування	Зазвичай наявний ПК/ноутбук (без витрат)	Не потрібен, але немає гнучкості
Програмне забезпечення	QLC+, Freestyler, MagicQ (безкоштовне)	Вбудоване у консоль (в ціні)
Кабелі DMX	$1–$2 за метр; загалом ~ $50	$1–$2 за метр; загалом ~ $50
Додаткове обладнання (розгалужувачі тощо)	При потребі — ~$30–$100	При потребі — ~$30–$100
Навчання та освоєння	Інтуїтивно, велика кількість безкоштовних гайдів	Часто вимагає спеціального навчання
Можливість масштабування	Висока — дешеве розширення	Обмежена, потребує нової консолі
Загальна вартість впровадження	$70–$200 *(без урахування ПК)*	$600–$3200+

PC-DMX система значно дешевша у впровадженні, особливо якщо вже є наявний комп'ютер. Вона також гнучкіша та легше розширюється, що робить її ідеальною для навчання, тимчасових сцен, невеликих шоу чи інтерактивних

інсталяцій. Традиційна DMX-консоль має сенс лише у великих інсталяціях або професійних театральних постановках, де потрібна повна автономність і миттєве керування «з кнопки», без участі ПК.

1.4 Можливості розширення і кастомізації PC-DMX інтерфейсу

Однією з ключових переваг цифрових рішень на базі PC-DMX інтерфейсу є гнучкість системи, яка забезпечується широкими можливостями розширення та кастомізації. Такий підхід дозволяє адаптувати систему до будь-яких потреб користувача — від простих статичних установок до динамічних мультимедійних шоу з інтеграцією сторонніх протоколів та пристроїв [24].

Розширення кількості керованих каналів передбачає базовий DMX512-потік забезпечує керування до 512 каналів (універс) — цього вистачає для невеликих сцен, студій або інсталяцій.

Для більших проєктів можна підключати кілька інтерфейсів або використовувати інтерфейси з підтримкою багатьох універсів (наприклад, 2×512, 4×512 тощо).

Сучасне ПЗ (наприклад, QLC+, MagicQ) підтримує паралельне керування кількома потоками, що дозволяє масштабувати систему практично без обмежень.

Кастомізація інтерфейсу керування надає можливість для більшості програм засобами PC-DMX використовувати візуальні редактори інтерфейсів, де користувач може створювати власні панелі керування, налаштовувати кнопки, слайдери, віджети.

Інтеграція з сенсорними екранами, MIDI-контролерами, клавіатурами, джойстиками дозволяє адаптувати керування під конкретного оператора чи завдання.

Підтримка зовнішніх протоколів та інтеграція для PC-DMX системи можуть бути інтегровані з іншими цифровими стандартами:

- MIDI — синхронізація з музичними пристроями;

- OSC (Open Sound Control) — дистанційне керування з мобільних пристроїв;

- Art-Net / sACN — передача DMX-потоків через Ethernet, масштабування у мережах.

Завдяки API або плагінам, деяке ПЗ дозволяє писати власні скрипти для інтерактивного керування (наприклад, запуск сцени при виявленні руху або зміні температури).

Існує можливість автоматизації сценаріїв, зокрема:

- Можливість створення таймерів, тригерів, умовних подій (IF-THEN логіки).

- Реалізація циклічних або реактивних сценаріїв (наприклад, реакція на звук, зміну освітлення в залі тощо).

- Сценарії можуть імпортуватися, експортуватися, змінюватися «на льоту» без перезапуску всієї системи.

Підтримка відкритого коду та DIY-рішень забезпечується набором інтерфейсів (наприклад, Enttec Open DMX або Arduino-проекти), які мають відкритий код. Це дозволяє:

- змінювати прошивку інтерфейсу;

- реалізовувати специфічні функції або протоколи;

- інтегрувати керування світлом у більші автоматизовані системи (наприклад, «розумний дім»).

PC-DMX інтерфейси забезпечують високий рівень масштабованості, кастомізації та автоматизації, що робить їх ідеальним рішенням для проєктів з динамічними вимогами. Вони легко адаптуються до змін у сценарії, апаратному середовищі чи потребах користувача без необхідності повного оновлення системи.

1.5 Інтеграція з новітніми технологіями

Системи цифрового керування світлом на основі PC-DMX інтерфейсу не є ізольованими - вони активно розвиваються завдяки інтеграції з новітніми

технологіями, що відкриває нові горизонти в області сценічного мистецтва, архітектурного освітлення, автоматизації, шоу-дизайну та кіберфізичних систем. Гнучкість програмного забезпечення та універсальність DMX-протоколу дозволяють легко поєднувати керування освітленням із іншими цифровими системами.

Інтеграція з Інтернетом речей (IoT) реалізується за рахунок таких можливостей: PC-DMX інтерфейс може використовуватись у складі систем розумного середовища, де керування світлом здійснюється через мережу IoT-пристроїв, наприклад, датчики руху, температури, вологості або розумні контролери можуть подавати сигнали на ПК, який через інтерфейс керує світловими приладами.

Також можливе застосування MQTT-протоколів для передачі даних у реальному часі між пристроями.

Хмарне керування та моніторинг реалізується за допомогою мережевих рішень (Art-Net, sACN) та віддалених серверів, системи освітлення можуть керуватися через Інтернет.

Це дає змогу створювати дистанційно керовані інсталяції (наприклад, світлові інсталяції в публічному просторі, керовані через веб-інтерфейс).

Також можлива реалізація запису логів, моніторингу стану приладів, діагностики несправностей.

Можливим є використання штучного інтелекту (AI) та машинного навчання. Сучасні дослідження демонструють використання AI для автоматичної генерації світлових шоу, які адаптуються до:

- ритму музики;
- характеру події;
- емоційного тону виступу.

Наприклад, нейромережі можуть аналізувати музичний потік у реальному часі й генерувати динамічні сцени освітлення, які автоматично передаються до PC-DMX системи.

У професійному шоу-дизайні використовуються віртуальні моделі сцени для попереднього програмування світлових ефектів.

Програмне забезпечення, сумісне з PC-DMX (наприклад, Capture, WYSIWYG, L8), дозволяє:

- моделювати сцену в 3D;

- в реальному часі тестувати сценарії освітлення;

- створювати візуалізації для доповненої реальності (AR), де режисер може віртуально «ходити» по сцені з планшетом.

Завдяки інтеграції з розумними помічниками (Alexa, Google Assistant) або системами розпізнавання жестів (Leap Motion, Kinect), користувач може керувати світлом без фізичного контакту з інтерфейсом.

Наприклад, команда «Затемни сцену» може передаватися через голосовий помічник у ПЗ QLC+, а жест - запускати сцену або ефект.

Також можлива інтеграція з мультимедійними системами синхронізації з відео, аудіо, піротехнікою тощо - стандартна функція сучасного програмного забезпечення. Наприклад, світло синхронізується з відеорядом або інтерактивною графікою, що створює повноцінне шоу з урахуванням динаміки подій.

PC-DMX інтерфейси відкривають потенціал для інтеграції з інноваційними технологіями, що дозволяє створювати інтелектуальні, адаптивні, віддалено керовані та інтерактивні системи освітлення нового покоління. Це робить їх не лише ефективним інструментом у сфері розваг, а й перспективною технологією в освітніх, наукових і промислових рішеннях.

1.6 Застосування Raspberry Pi для реалізації PC-DMX інтерфейсу

Застосування Raspberry Pi для реалізації PC-DMX інтерфейсу є перспективним напрямком розвитку програмно-техічних застобів, особливо для створення доступних, гнучких та програмованих рішень для керування освітленням та іншими пристроями, які використовують протокол DMX512 [26-29].

Розглянемо основні етапи реалізації PC-DMX інтерфейсу.

Для реалізації DMX-інтерфейсу на Raspberry Pi, можна використовувати додаткові апаратні компоненти, такі як:

- DMX-USB адаптери, які є спеціалізованими пристроями. Вони дозволяють Raspberry Pi спілкуватися з DMX-установками через USB-порт.

- GPIO порти, які входять до Raspberry Pi. Вони є універсальними портами введення/виведення (GPIO), які можна використовувати для генерування сигналу DMX, хоча це потребує складного програмування та врахування специфікацій протоколу.

Для роботи з DMX протоколом можна використовувати такі бібліотеки:

- OLA (Open Lighting Architecture), яка є відкритим фреймворком для керування освітленням через DMX, який підтримує Raspberry Pi. Вона дозволяє підключати пристрої та програмувати інтерфейси для керування світлом, використовуючи різні протоколи, включаючи DMX512 [30-33].

- RDM (Remote Device Management), яке є розширенням DMX для двостороннього зв'язку між контролерами та пристроями [34-37].

- Pi-DMX - бібліотека для роботи з DMX на Raspberry Pi, яка дозволяє здійснювати управління світлом [38-41].

Для реалізації апаратного забезпечення необхідні такі компоненти:

- DMX контролери, які необхідні для тестування та інтеграції можна підключити DMX сумісні пристрої (наприклад, світлові прилади).

- Роз'єми XLR, а саме 5-пінові XLR роз'єми для підключення пристроїв. На Raspberry Pi можна підключити зовнішній DMX-конвертер, що використовує ці роз'єми.

Для управління DMX сигналом на Raspberry Pi можна використовувати такі мови програмування [42-46]:

- Python, яка підтримує бібліотеки для роботи з GPIO та DMX, Python є популярним вибором для таких проектів.

- C/C++, яка забезпечує більшу гнучкість у розробці швидкодіючих програм для управління апаратним забезпеченням.

Для зручного використання програмно-технічних засобів можна створити графічний інтерфейс, що дозволить операторам або програмістам налаштовувати параметри освітлення або інших DMX-сумісних пристроїв безпосередньо через Raspberry Pi. Це може бути зроблено з допомогою веб-інтерфейсу або програмного забезпечення для ПК.

1.7 Висновки до першого розділу

У межах першого розділу було здійснено комплексний аналіз сучасних підходів до цифрового керування освітленням, зосереджений на застосуванні PC-DMX інтерфейсу. Було обґрунтовано доцільність використання протоколу DMX512 як основи для реалізації гнучкого й масштабованого керування світловими пристроями.

Визначено переваги його універсальності, відкритості, можливості інтеграції з новітніми протоколами, економічності та легкості у використанні.

Окремо розглянуто особливості застосування Raspberry Pi як апаратної платформи для створення доступного та ефективного програмно-технічного рішення, що підтримує кастомізацію та дистанційне налаштування через веб-інтерфейс.

2 ПРОЄКТУВАННЯ СИСТЕМИ ПРОГРАМНО-ТЕХНІЧНІ ЗАСОБИ РЕАЛІЗАЦІЇ PC-DMX ІНТЕРФЕЙСУ НА ОСНОВІ ОДНОПЛАТНОЇ КОМП'ЮТЕРНОЇ СИСТЕМИ RASPBERRY PI

2.1 Протокол DMX

Для виконання валіфікаційної роботи розглянемо використовувані протоколи та специфікації.

DMX використовується в кваліфікаційній роботі для позначення стандарту DMX512-A [EST13].

Типове налаштування DMX налаштування освітлення схематично показано на рисунку 2.1.

Воно виглядає наступним чином: DMX пульт виводить DMX сигнал, який надсилається по кабелю на перший освітлювальний прилад (в даному випадку простий диммер).

Потім сигнал зациклюється на наступний світильник, в даному випадку - світлодіодний налобний ліхтар.

Останнім світильником у цьому прикладі є рухома голова, що рухається. Її вихідний DMX-сигнал не використовується в інших світильниках, тому кінцевий резистор (тут це чорно-жовта «паличка») завершує DMX-лінію.

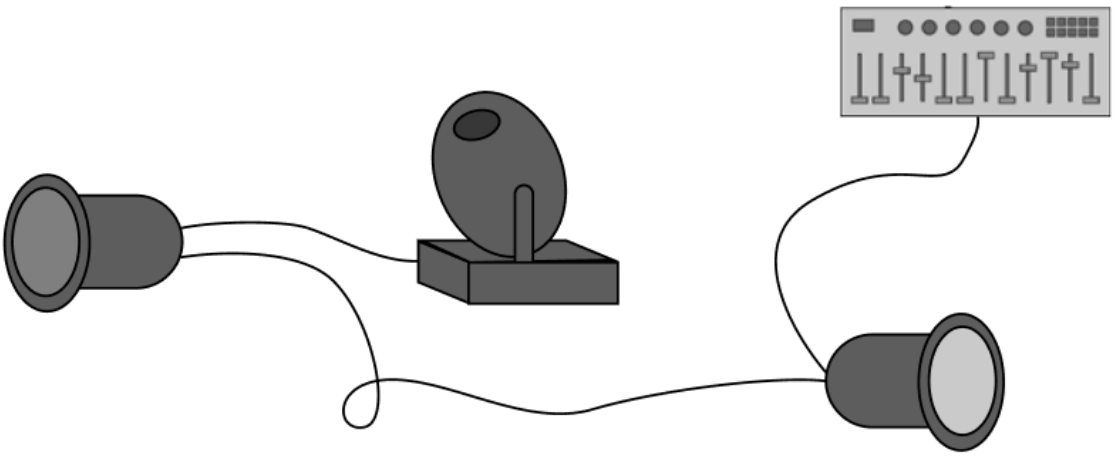


Рисунок 2.1 - Схематичний приклад налаштування освітлення за допомогою DMX

Сигнал DMX складається з 512 цілих чисел від 0 до 255, кожне з яких передає значення одного каналу.

Кожен світильник f в лінії DMX прослуховує певну кількість nf цих каналів (встановлену виробником), щоб керувати своїми функціями.

Тепер користувач повинен присвоїти кожному приладу адресу af, щоб позначити перший канал, який він повинен прослуховувати, наприклад, за допомогою DIP-перемикача або невеликого дисплея на кожному з підключених приладів.

Порядок присвоєння адрес не обов'язково повинен збігатися з порядком розташування пристроїв у лінії.

Щоб показати, як це працює на практиці, приклад продовжено.

Спочатку світлодіодна фара отримує адресу aLED = 1. Вона потребує nLED = 3 канали, щоб окремо керувати червоним, зеленим і синім (RGB) компонентами, тому слухає канали 1, 2 і 3.

Щоб світлодіод світився жовтим кольором, нам потрібно встановити для каналів 1 і 2 (червоний і зелений) значення 255, а для каналу 3 (синій) - 0.

Наступна адреса, яку можна можемо призначити без конфлікту, - це aLED + nLED = 4.

Необхідно задавати цю адресу диммеру (adim = 4), якому для керування яскравістю потрібен лише один канал.

Тепер необхідно резервувати адресу 5 для майбутнього використання іншого диммера, так що два канали диммера будуть знаходитися поруч один з одним на настільній консолі.

Рухомій голівці присвоюється адреса amov = 6. Її RGB-значеннями, панорамуванням / нахилом і колесом можна керувати, а отже, потрібно 6 каналів.

512 каналів, що передаються по одній лінії, називаються DMX-всесвітом. Пульт може виводити кілька універсумів, що дозволяє керувати більшою кількістю приладів.

Замість фізичного DMX пульта, його завдання може виконувати програмне забезпечення. Комп'ютер, на якому вона працює, підключається до DMX-лінії через фізичний інтерфейс.

2.2 DMX-розгалужувачі

Існує два апаратних компоненти, які можна використовувати в DMX-лінії, щоб зробити проводку між приладами та джерелами DMX більш гнучкою:

- DMX-розгалужувач використовується як Т-подібний перехідник для передачі одного вхідного сигналу на дві (або більше) вихідні лінії.

- DMX-мікшер об'єднує сигнали з двох вхідних ліній A і B (рідко більше) в один вихідний сигнал.

Компоненти мають такі режими роботи:

- Резервний, тобто поки A є дійсним сигналом, зациклюйте його, інакше використовуйте B.

- Злиття, коли використовувати перші x каналів A, потім заповнити решту 512 мінус x каналів першими каналами B.

- LTP («останній має перевагу»).

- HTP («найвищий має перевагу»), де для кожного з 512 каналів використовувати найбільше значення обох відповідних каналів у A і B.

Протокол DMX визначає послідовний сигнал (показаний на рисунку 2.2) зі швидкістю передачі 250000 біт на секунду.

Він складається з окремих пакетів, кожен з яких ініціюється послідовністю скидання (як завгодно довгий низький сигнал BREAK, за яким слідує висока позначка після BREAK (MAB) і слот 0).

Слот містить 8-розрядні дані (молодший біт першим), перед якими стоїть низький стартовий біт, за яким слідують два високих стопових біти.

Дані в слоті 0 називаються стартовим кодом, він завжди дорівнює 0x00 для DMX-пакетів.

Інші стартові коди можуть використовуватися для позначення специфічної інформації виробника або спеціальних функцій; ці пакети ігноруються стандартними DMX-приймачами, тому в цьому проекті вони не розглядаються.

Після послідовності скидання, значення каналів всесвіту передаються по одному слоту.

Оскільки канали передаються послідовно, світильники в DMX-лінії можуть підраховувати отримані канали і починати прослуховування, як тільки їхня адреса збігається з номером поточного каналу.

Тому номер каналу не потрібно передавати окремо. Слоти можна розділити за допомогою високої позначки між слотами (MBS).

Після останнього каналу висока позначка перед BREAK (MBB) або наступним BREAK вказує на кінець пакета.

Принаймні один пакет на секунду повинен передаватися, хоча бажано швидше оновлення, щоб забезпечити швидку реакцію світильників і, зокрема, плавне згасання світла.

Щоб збільшити частоту оновлення, не всі 512 каналів повинні передаватися в пакеті.

Часову діаграму DMX подано на рисунку 2.2.

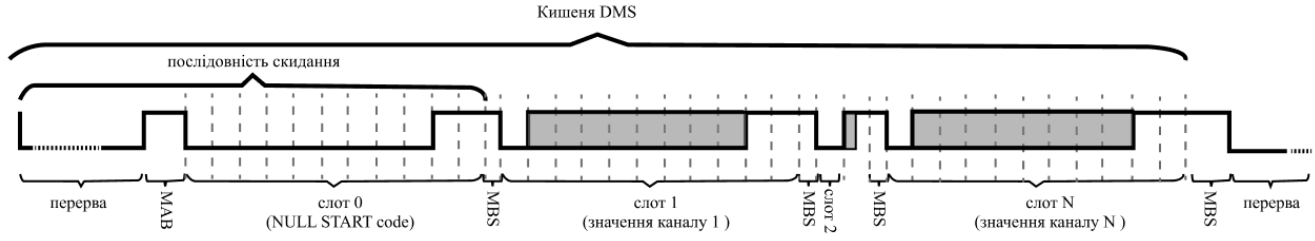


Рисунок 2.2 – Часова діаграма DMX

Якщо розглядати пакети з усіма 512 каналами, то можна побачити, що мінімальний час пакета становить 21,57 мс, або максимальна частота оновлення 45 Гц.

Розширенням DMX, яке не буде важливим для цієї роботи, але про яке варто згадати, є віддалене керування пристроями (Remote Device Management, RDM).

Воно дозволяє встановлювати DMX-адреси приладів та інші параметри за допомогою контролера RDM, який є розширеним DMX пультом або програмним забезпеченням.

Він працює шляхом чергування односпрямованого DMX-сигналу з двоспрямованими RDM-пакетами.

У DMX використовуються електричні специфікації, визначені в промисловому стандарті EIA-485 (також відомому як RS-485), який описує збалансовану передачу сигналів по лінії зв'язку.

«Збалансований» означає, що біти даних кодуються за допомогою різниці потенціалів між витою парою Data + і Data -.

Це зменшує завадостійкість, оскільки шум додається до обох ліній даних однаково - фактично нівелюється в різниці - і, таким чином, робить можливим використання ліній довжиною до 1,1 км. Однак для ліній DMX рекомендується не перевищувати 300 м.

На рисунку 2.3 показана електрична схема такої шини. Зліва - передавач, який перетворює вихідний сигнал у диференціальний.

Всі приймачі внизу (допускається до 32) роблять те ж саме в іншому напрямку.

Як на ближньому кінці (з боку передавача), так і на дальньому (після останнього приймача) встановлюється кінцевий резистор 120 Ом, щоб мінімізувати відбиття, які можуть заважати сигналу.

Стандарт DMX вимагає використання 5-контактних роз'ємів XLR, за винятком випадків, коли їх фізично неможливо встановити.

Незважаючи на це, більшість DMX-апаратури оснащена 3-контактним XLR-роз'ємом замість нього або додатково.

Це пов'язано з тим, що потрібно лише 3 контакти, а 3-контактні XLR-кабелі поширені в івент-технологіях, оскільки вони також використовуються для мікрофонів.

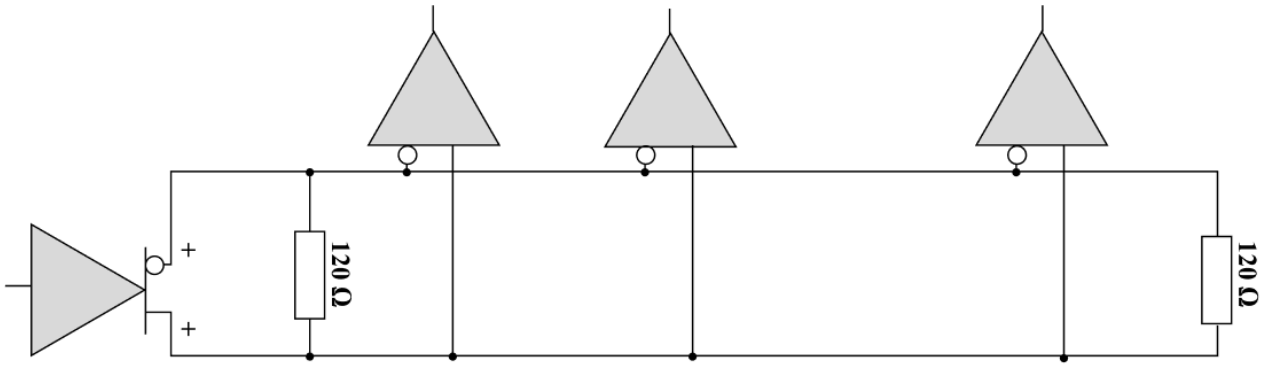


Рисунок 2.3 – Шина EIA-485 з одним передавачем і до 32 приймачами

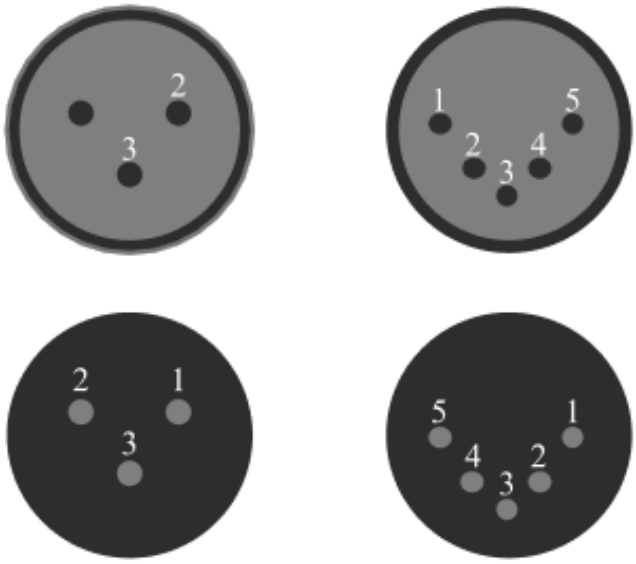


Рисунок 2.4 – Роз'єми XLR, що використовуються для DMX

2.1.3 Протоколи Art-Net та sACN

DMX дозволяє лише один або два всесвіти на лінію, що може зробити прокладання кабелю непрактичним у деяких випадках використання

Щоб подолати цю проблему, компанія з виробництва світлового обладнання Artistic Licence створила протокол передачі DMX через UDP під проєктною назвою Art-Net. Він дозволяє надсилати кілька DMX-всесвітів через стандартну IP-мережу і, таким чином, значно розширює гнучкість систем освітлення, оскільки один канал Ethernet або бездротова мережа можуть бути використані для значної частини транспортного шляху.

Існують контролери освітлення і світильники, які працюють безпосередньо з Art-Net (виключно або поряд з традиційним DMX), всі інші можуть бути підключені через Art-Net Node, який перетворює DMX в IP і з нього.

Часто протокол використовується для зв'язку між програмним забезпеченням для освітлення на комп'ютері та одним з таких вузлів, що діє як джерело DMX для світильників, як на рисунку 2.5.

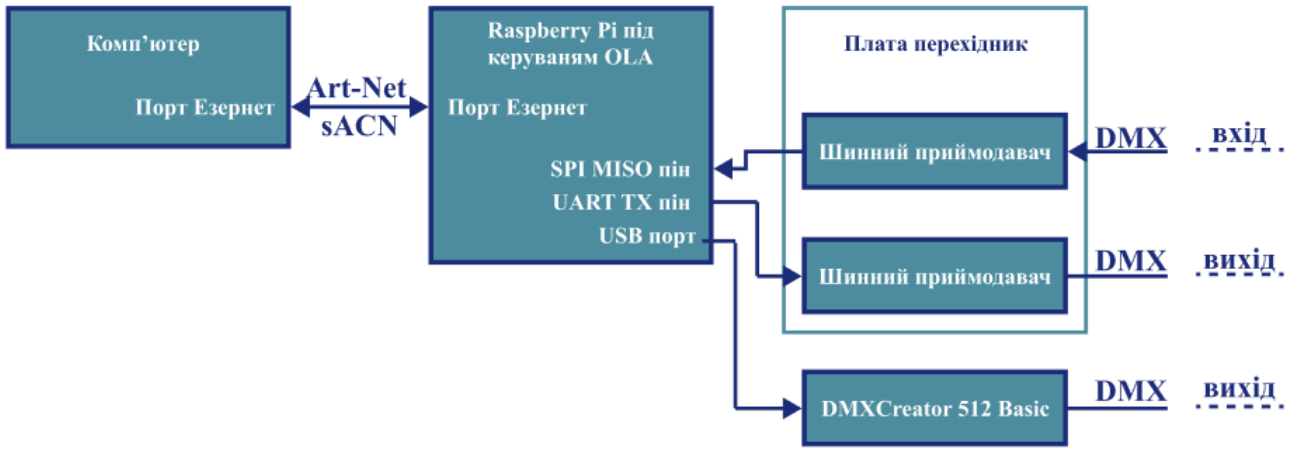


Рисунок 2.5 – Схематичне підключення вузла Art-Net

Потоковий ACN (sACN), який був стандартизований як ANSI E1.31 є відкритим протоколом ESTA з тими ж цілями і поділяє більшість високорівневих властивостей з Art-Net.

2.2 UART

Універсальний асинхронний приймач-передавач (UART) - це інтерфейс, присутній у багатьох мікроконтролерах, який дозволяє здійснювати зв'язок через послідовну шину.

Тут немає додаткового тактового сигналу, приймач синхронізується за допомогою фіксованого формату даних. Біти даних передаються послідовно, оформлені в слоти з низьким стартовим і високим стоп-бітом. Якщо сигнал залишається низьким довше, ніж один слот, виконується умова переривання.

Щоб уникнути непорозумінь, різні параметри повинні бути зафіксовані як на приймачі, так і на передавачі: швидкість передачі, кількість бітів даних у слоті (зазвичай від 5 до 9), нумерація бітів (старший або молодший біт першим), кількість використовуваних стоп-бітів (один або два) і чи повинен кожен слот додатково містити біт парності.

Оскільки протокол синхронізації DMX є спеціалізацією цієї специфікації, надсилання та отримання даних DMX через UART можливе. Єдиним застереженням є нестандартна швидкість передачі даних 250 кбіт/с.

2.3 SPI

Послідовний периферійний інтерфейс (SPI) - це інтерфейс синхронної передачі даних між головним і декількома slave-пристроями, розроблений компанією Motorola.

Тут буде розглянуто лише конфігурацію незалежного підлеглого пристрою, яку показано на рисунку 2.6.

Подаючи низький рівень сигналу на один з виводів Slave select / Chip enable (CE), ведучий може повідомити відповідний ведений про те, що він хоче встановити зв'язок з ним.

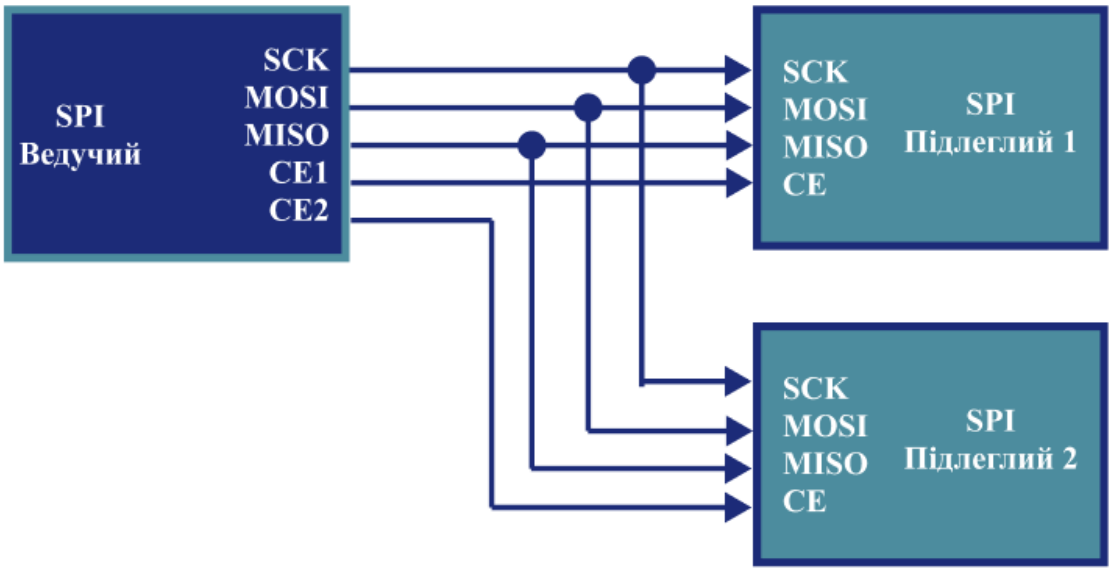


Рисунок 2.6 – Схема ведучого (master) та підлеглий (slave), пристроїв SPI

Після цього ведучий генерує тактовий сигнал на виводі SCK (Serial clock) і одночасно зчитує на виводі MISO (Master In, Slave Out) і передає на виводі MOSI (Master Out, Slave In) по одному біту за один тактовий цикл. Після завершення передачі даних (наприклад, один байт надіслано підлеглому, який потім може відповісти одним байтом, але це залежить від протоколу, встановленого між пристроями), ведучий скидає всі виводи в неактивний стан. Існує декілька режимів SPI, які визначають, коли повинна відбуватися операція читання/запису бітів по відношенню до тактового сигналу.

Для простоти тут показано лише режим 0, в якому стан холостого ходу SCK є низьким, а передача даних починається з першим фронтом тактового сигналу (див. рисунок 2.7).

2.4 Аналіз вимог та проектування системи

Вимоги розроблені для конкретного випадку використання невеликих об'єднань, таких як технічні команди в молодіжних групах (тобто не професійних компаній з організації заходів тощо).

Інтерфейс PC-DMX повинен пропонувати функції для користувачів, але при цьому бути доступним за ціною.

Інтерфейс повинен підтримувати це, будучи сумісним з якомога більшою кількістю з них.

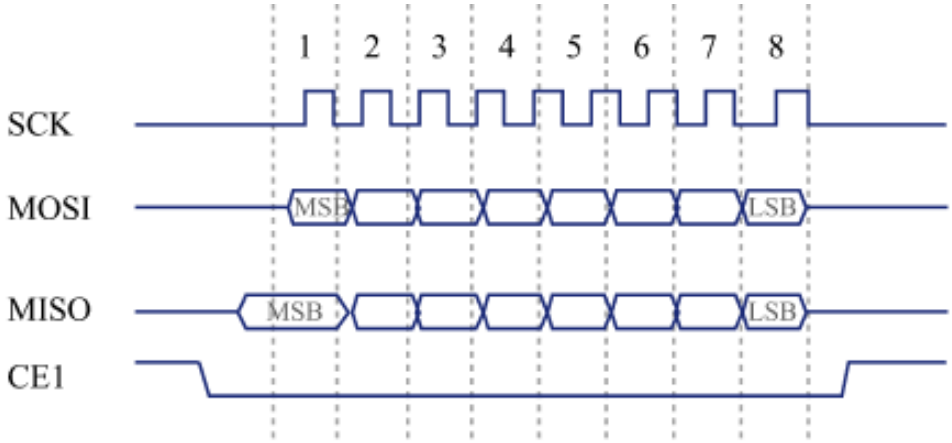


Рисунок 2.7 – Часова діаграма SPI. MSB та LSB - це скорочення від старшого та молодшого значущого біта відповідно\

Підключення до комп'ютера має бути можливим через Ethernet, щоб можна було продовжити кабель через стандартне мережеве обладнання, таке як комутатори, маршрутизатори та точки доступу Wi-Fi. USB-з'єднання не є достатнім, оскільки USB-пристрої потребують спеціальних драйверів та конфігурації, що зробить інтерфейс менш портативним, наприклад, у разі необхідності швидкої заміни комп'ютера в екстрених ситуаціях.

Серед протоколів, що використовуються для передачі даних, принаймні один повинен бути відкритим, тобто підтримуватися або sACN, або Art-Net10.

Інтерфейс повинен підтримувати виведення щонайменше двох DMX-універсумів, щоб мати змогу керувати достатньо великою кількістю світильників, і введення щонайменше одного, щоб забезпечити тактильне керування DMX-каналами за допомогою мікшерного пульта.

Спосіб обробки вхідних сигналів DMX повинен бути конфігурованим:

- або значення DMX-каналу надсилаються через Art-Net до керуючого програмного забезпечення (за замовчуванням Art-Net Node-подібний режим), наприклад, для керування програмними функціями за допомогою апаратних фейдерів мікшерного пульта;
- або він діє як DMX-розгалужувач, перенаправляючи свій вхідний DMX-сигнал на обидва DMX-виходи;
- або вхідні канали DMX об'єднуються зі значеннями каналів, що передаються мережею, в один з вихідних універсумів за допомогою одного з режимів об'єднання.

Всі вхідні та вихідні DMX-сигнали повинні оброблятися з високою частотою оновлення, щоб затримки залишалися низькими і було можливим плавне згасання світла.

Використання інтерфейсу має бути максимально простим для кінцевих користувачів. Це означає, що для його використання не потрібні ні глибокі знання про протокол DMX або комп'ютерні мережі, ні спеціальні ноу-хау про цей специфічний інтерфейс.

Однак передбачається, що кінцеві користувачі знають, як працювати з програмним та апаратним забезпеченням DMX в цілому.

Більш складні функції, такі як згадане вище гнучке відображення входів, повинні бути достатньо тривіальними, щоб бути зрозумілими за короткий проміжок часу.

Вся установка повинна коштувати менше 100е і бути розширюваною, тобто майбутні потреби, такі як потреба в більшій кількості DMX-всесвітів, повинні бути легко реалізовані без перепроектування і перебудови всього апаратного і програмного забезпечення.

Крім того, було б добре, якби і апаратне, і програмне забезпечення були з відкритим вихідним кодом, щоб дозволити іншим розширювати і покращувати інтерфейс.

Оскільки обмеження в 100е звужує спектр професійного обладнання до різних USB-DMX адаптерів та одного Art-Net Node від Eurolite - всі вони підтримують лише один вихідний універсум, і жоден з них не має DMX входу - залишаються лише проекти «зроби сам».

DMXControl Projects' Node U1, також може бути підключений лише через USB. Таким чином, він повинен явно підтримуватися програмами освітлення.

Тим не менш, він не відповідає вимогам мережевого підключення.

Протокол, який відповідає більшості вимог, - це Quad Art Net Box , який підтримує чотири DMX-виходи, один з яких можна перемикати на вхід.

Вихідні коди та схеми доступні в Інтернеті, а також можна замовити монтажний набір.

Однак немає інформації про те, чи можна гнучко об'єднати вхідний DMX-сигнал в один вихідний DMX-всесвіт, чи він завжди перенаправляється на вихід Art-Net.

Замість того, щоб виконувати всю роботу в одному мікроконтролері, як у попередніх проектах, можна використовувати набагато потужніший Raspberry Pi з додатковим співпроцесором на платі розширення (іноді її називають щитом), який підключається до контактів GPIO (General Purpose Input / Output).

Таким чином, потрібно замінити лише програмне забезпечення (перепрошивши SD-карту), щоб воно відповідало конкретному випадку використання.

USB, Art-Net, sACN, Open Sound Control і MIDI можуть бути перетворені в DMX за допомогою правильного образу SD-карти.

Апаратне забезпечення плати розширення підтримує лише один вхідний і один вихідний універсум.

Програмне забезпечення Open Lighting Architecture (OLA) є універсальним перекладачем протоколів для DMX-сигналів, підтримуючи різні пристрої за допомогою плагінів. Його можна встановити на Raspberry Pi, і вже доступний плагін, що забезпечує власний вихід DMX через порт UART.

2.5 Raspberry Pi як адаптер системи

Одноплатний комп'ютер Raspberry Pi є основою адаптера. Він має порти Ethernet та USB, а також процесор, достатньо потужний для роботи OLA [47-50].

Необхідно розробити плату розширення, підключену до його GPIO виводів. Вона має бути обладнана двома трансиверами шини EIA-485 для забезпечення одного входу DMX і, через вищезгаданий плагін UART, одного виходу [51-54].

Другий вихід буде забезпечуватися адаптером USB-DMX, який був доступний мені, DMXCreator 512 Basic. Його протокол потрібно переробити і включити в USB-плагін OLA. Початкове спостереження за протоколом показало доцільність такого підходу [55-59].

Також необхідно розрбоити плагін OLA, який дозволяє прямий вхід DMX на Raspberry Pi (рис. 2.8).

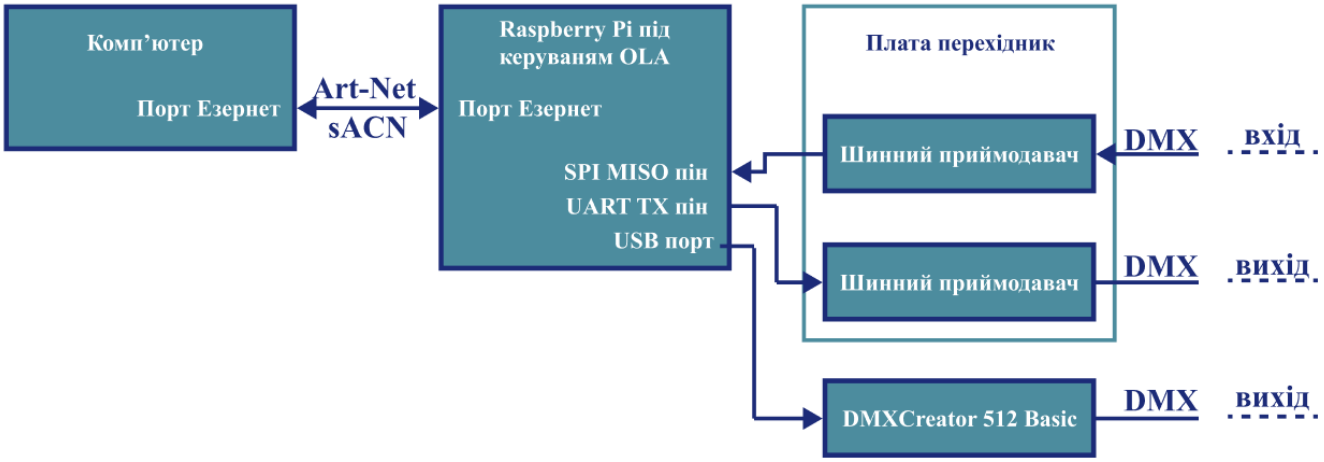


Рисунок 2.8 – Схема запланованого інтерфейсу PC-DMX

2.6 Open Lighting Architecture як системне програмне забезпечення для реалізації проєкту

Проект Open Lighting Architecture (OLA) – це безкоштовне програмне забезпечення з відкритим вихідним кодом. Вона працює на Linux або Mac і здатна

взаємодіяти з обладнанням USB DMX512, DMX512 по IP-протоколам і GPIO-шпильками Raspberry Pi.

Застосунок має веб-інтерфейс для легкого створення, моніторингу та налаштування DMX-всесвітів. OLA є частиною більшого проекту Open Lighting Project, метою якого є створення високоякісного, вільного програмного забезпечення для індустрії розважального освітлення.

В OLA широко використовуються деякі ключові слова:

- Порт - це точка, куди передається максимум 512 значень DMX-каналу (вихідний порт) або зчитується (вхідний порт). Він може бути як фізичним, так і віртуальним (як у Art-Net).
- Пристрій групує порти разом, він складається принаймні з одного порту.
- Плагін забезпечує підтримку розпізнавання, підключення та комунікації з одним або декількома пристроями. Він має бути скомпільований разом з OLA (тобто не може бути завантажений і підключений пізніше) і, таким чином, має бути частиною проекту. Під час виконання плагіни можна вмикати та вимикати незалежно.
- Всесвіт OLA - це внутрішній набір з 512 значень каналів DMX. Користувач може вносити зміни до вхідних портів для отримання нових даних та/або вихідних портів для передачі поточних значень каналів.

Плагін Art-Net Плагін ArtNet20 реалізує протокол Art-Net, який підтримує максимум чотири вхідні та чотири вихідні універсуми на одну IP-адресу.

Цей плагін створює вхідні та вихідні порти відповідно, які можуть бути підключені до OLA-універсумів і, таким чином, передавати DMX-дані до/від зовнішніх світлових програм.

Оскільки протокол Art-Net розроблений з урахуванням зворотної сумісності, як новіші, так і старіші клієнтські програми можуть спілкуватися з OLA.

Цей плагін не потребує додаткового обладнання, він використовує вже наявні мережеві порти.

2.7 Плагін UART

Плагін Native UART DMX Plugin створює один вихідний порт, який безпосередньо генерує DMX-сигнал через порт UART головного пристрою, зазвичай Raspberry Pi.

Цей сигнал на виводі GPIO повинен бути пропущений через мікросхему шинного трансивера для перетворення його в збалансований сигнал EIA-485 з допустимою різницею потенціалів.

DMX-512 використовує послідовні сигнали переривання. Вони не можуть бути надіслані простим записом символів з послідовного порту. Інтерфейс termios2 надає методи для запуску та завершення BREAK.

Між ними стандартний виклик сплячого режиму перериває потік надсилання на вказаний час; неточності в цьому випадку не мають значення. DMX-512 має відносно жорсткі часові вимоги до різних елементів сигналу.

2.8 Плагін USB

Плагін USB DMX забезпечує підтримку різноманітних адаптерів USB-DMX. Кожен з них контролюється «субплагіном», який розширює загальну базову реалізацію.

Це спрощує доступ до бібліотеки libusb і зменшує дублювання коду.

Кожен субплагін отримує сповіщення про новий підключений USB-пристрій і може претендувати на нього, якщо ідентифікатор виробника, ідентифікатор пристрою і, можливо, інша інформація збігаються із заздалегідь визначеними значеннями.

Після цього він відповідає за створення портів і зв'язок з пристроєм.

2.9 Плагін SPI

Плагін SPI дозволяє безпосередньо керувати світлодіодними стрічками за допомогою SPI-керованих драйверів світлодіодів, таких як WS2801 або LPD8806.

Розширені функції, такі як використання апаратних SPI-мультиплексорів і декількох піксельних смуг, доступні, але виходять за рамки цього пояснення.

2.10 Висновки до другого розділу

У другому розділі проведено системне проектування кіберфізичної системи адаптивного застосування моніторингових елементів. Визначено її апаратну та програмну архітектуру, описано компоненти системи та їхню взаємодію. Зокрема, проаналізовано реалізацію передачі даних через UART, USB та SPI, застосування Open Lighting Architecture (OLA) як системного ПЗ, а також адаптацію плагінів для підтримки специфічних каналів вводу/виводу.

Важливе місце займає формалізація функціонального призначення модулів, зокрема DMX-вузлів, що відіграють ключову роль у керуванні світловими пристроями та обробці даних.

3 ПРОГРАМНО-АПАРАТНА РЕАЛІЗАЦІЯ ПРОГРАМНО-ТЕХНІЧНІ ЗАСОБИ РЕАЛІЗАЦІЇ PC-DMX ІНТЕРФЕЙСУ НА ОСНОВІ ОДНОПЛАТНОЇ КОМП'ЮТЕРНОЇ СИСТЕМИ RASPBERRY PI

3.1 Початкове налаштування OLA на Raspberry Pi

Розглянемо кроки, необхідні для встановлення OLA на Raspberry Pi 4.

Для реалізації проєкту було завантажено образ Raspbian Lite з домашньої сторінки Raspberry Pi і прошито його на карту пам'яті microSD, з якої завантажується Raspberry Pi. Розмір карти microSD повинен бути не менше 8 ГБ. Доступ до захищеної оболонки (SSH) в Raspbian за замовчуванням вимкнено. Оскільки SSH необхідний для віддаленого підключення до Raspberry Pi, його було увімкнути, створивши новий (порожній) файл з назвою ssh у кореневому каталозі картки.

Після завантаження Raspberry Pi з новою прошитою картою microSD і підключення його до мережі за допомогою кабелю Ethernet, необхідно дізнатися IP-адресу, щоб відкрити захищену оболонку. У цій оболонці виконуються всі наступні команди.

Перш ніж продовжити, слід оновити всі програмні пакети, прошивку та ядро до останніх версій:

```
sudo apt update
sudo apt upgrade
sudo rpi-update
```

3.2 Збірка та встановлення OLA

Для збирання СПЗ OLA потрібні програмні пакети, які потрібно встановити за допомогою apt . Після цього завантажується найновіший вихідний код з GitHub, збирається, а отримані двійкові файли копіюються у правильні папки (Додаток А).

Було використано перехресну компіляцію OLA на потужнішому комп'ютері.

Після завершення інсталяції демон OLA можна запустити за допомогою olad і отримати доступ до його веб-інтерфейсу через порт 9090.

OLA має запускатися автоматично, як тільки Raspberry Pi завантажиться, що можна зробити за допомогою скрипта init (Додаток А).

3.3 Увімкнення UART

У файлі /boot/config.txt значення enable_uart=0 потрібно змінити на enable_uart=1, щоб зробити порт придатним для використання.

Максимальна швидкість передачі даних становить 115к біт/с (менше необхідних 250 кбіт/с), тому до цього ж файлу слід додати ще один рядок init_uart_clock=16000к для збільшення ліміту.

За замовчуванням, під час послідовного з'єднання виводяться повідомлення оболонки та ядра.

Таку поведінку слід вимкнути за допомогою raspi-config . Нарешті, щоб дозволити доступ до порту UART, користувача pi за замовчуванням слід додати до групи dialout:

```
sudo usermod -a -G dialout pi
```

Плагін UART OLA потрібно увімкнути і налаштувати так, щоб він використовував правильний порт UART. Це можна зробити, змінивши вміст файлу /home/pi/.ola/ola-uartdmx.conf як показано в скрипті Додатку А.

3.4 Налаштування USB

Розпізнані USB-пристрої є доступними для учасників групи plugdev, тому там слід додати pi, як описано вище.

Щоб розпізнати усі підтримувані OLA USB-пристрої, правила udev OLA імпортуються за допомогою скрипта, описаного в Додатку А.

3.5 Мережеві налаштування

Щоб полегшити безпосереднє підключення інтерфейсу PC-DMX до комп'ютерних систем ів, на яких не працює DHCP-сервер (що, можливо, стосується більшості систем кінцевих користувачів), йому присвоюється статична IP-адреса.

Після цього IP-адреса комп'ютера має бути лише в тій самій підмережі, щоб мати змогу спілкуватися з інтерфейсом. До файлу /etc/dhcpcd.conf було додано скрипт, поданий в додатку А.

За замовчуванням веб-інтерфейс OLA доступний через порт 9090, який можна змінити за допомогою параметра командного рядка.

Однак, оскільки порти нижче 1024 не можуть бути відкриті без привілеїв root, а olad відмовляється запускатися від імені root, порт 80 для веб-серверів не може бути використаний.

Ці команди встановлюють правила переадресації з порту 80 на 9090 як рішення у вигляді скрипта, поданого в Додатку А.

Щоб ці правила зберігалися після перезавантаження, до файлу /etc/rc.local було додано рядки коду:

```
sysctl -w net.ipv4.conf.all.route_localnet=1
iptables-restore < /etc/iptables/rules.v4
```

3.5 Проєктування плати розширення

Наступним кроком було проєктування плати розширення, яка містить обидві мікросхеми прийомопередавача для виходу UART та входу SPI і підключається через виводи GPIO Raspberry Pi.

Було розроблено схему, показану на рисунку 3.1. Схема розроблена таким чином, що виводи прийому (R), передачі (D) і дозволу передачі (DE) обох мікросхем трансиверів IC1 і IC2 - а не тільки один напрямок для кожної мікросхеми - підключаються до Raspberry Pi.

Підтримується один напрямок для кожної мікросхеми.

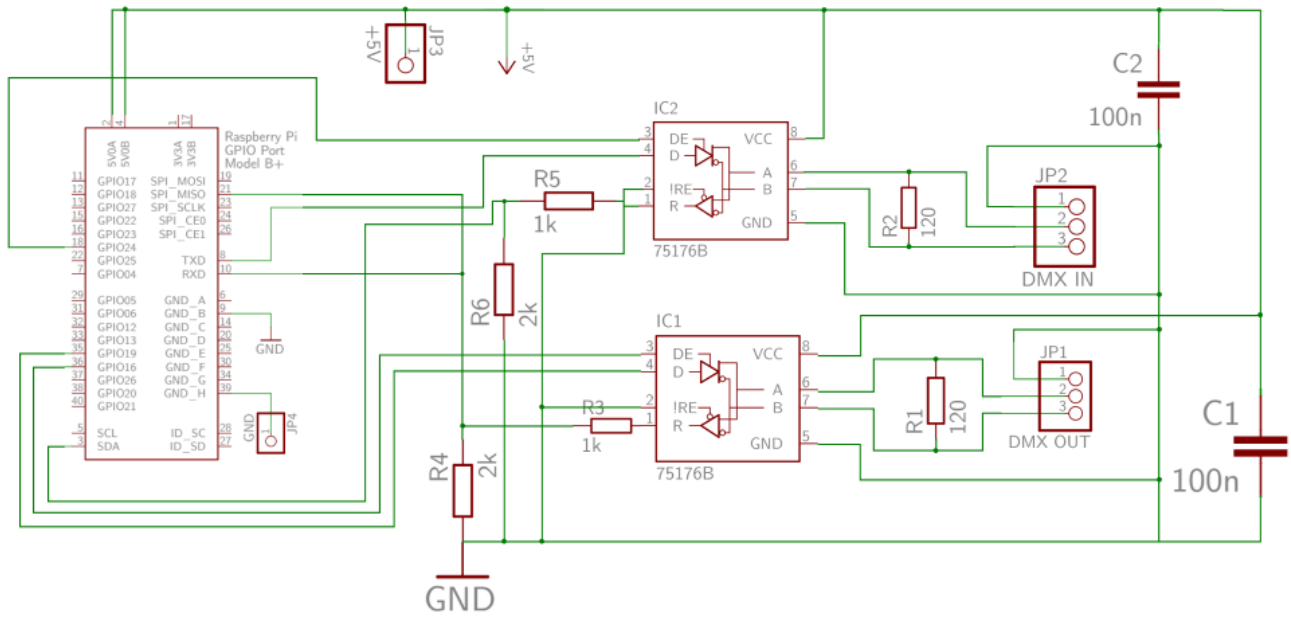


Рисунок 3.1 – Схема плати розширення

Отримані дані на виводах R передаються на виводи GPIO через дільники напруги (резистори R3/R4 і R5/R6) для зменшення вихідного сигналу 5В до дозволених 3,3В для входів Raspberry Pi.

Отримані дані IC2 (з вхідної лінії DMX) до виводу MISO (Master In, Slave Out) SPI надходять також на вивід RXD (прийом) UART, оскільки це була моя перша спроба зробити так, щоб прийом DMX-даних на Raspberry Pi працював.

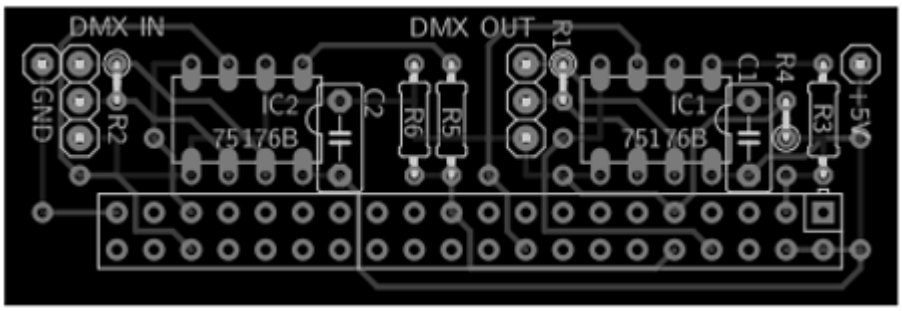
Отримані дані IC1 (з вихідної лінії DMX) надходять на вивід SDA, щоб залишити можливість використовувати шину I2C для розбору даних; в іншому випадку його можна використовувати як простий вивід GPIO.

Резистори R1 і R2 є кінцевими резисторами DMX відповідно до стандарту DMX. Виводи напруги живлення (VCC) обох мікросхем з'єднані з землею через невеликі конденсатори (C1 і C2), щоб пом'якшити піки напруги джерела живлення.

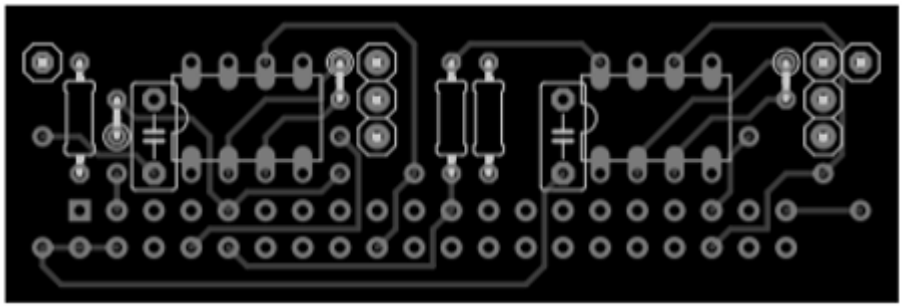
Всі інші частини схеми є штекерними роз'ємами; JP1 і JP2 йдуть на роз'єми DMX XLR, JP3 і JP4 дозволяють подавати зовнішню напругу для живлення Raspberry Pi через плату розширення замість вбудованого порту micro USB.

Потім ця схема була перетворена на двосторонній макет, який можна або надрукувати для створення друкованої плати, або припаяти вручну на просвердленій платі, що я і зробив.

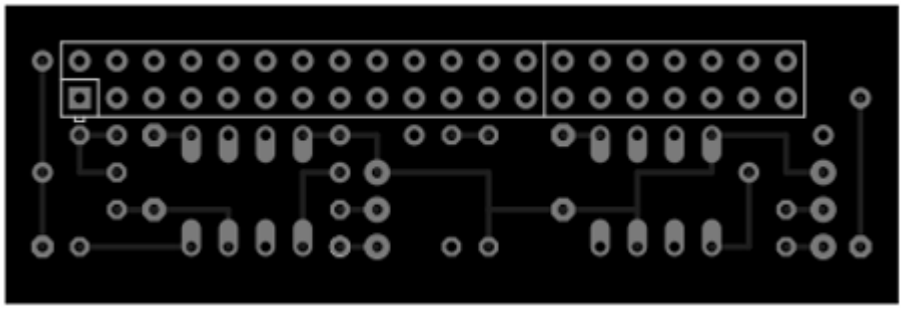
Макет показаний на рисунку 3.2, а готова плата - на рисунку 3.3.



а)



б)



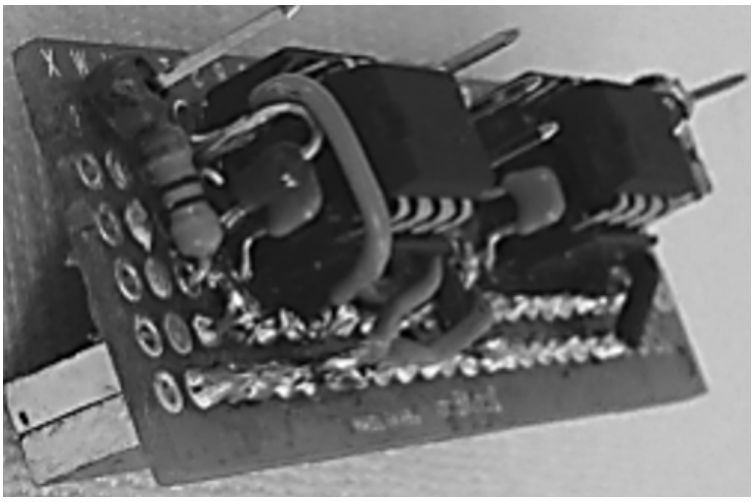
в)

Рисунок 3.2 - Розташування плати розширення

Загальний вигляд, вигляд зверху без етикеток, вигляд знизу без етикеток (дзеркальне відображення).

а)

б)



в)

Рисунок 3.3 - Готова плата розширення: а) вид зверху; б) вид знизу; в) вид збоку

3.6 Реалізація протоколу DMXCreator 512

Після встановлення плати розширення на виводи GPIO Raspberry Pi, світильниками, підключеними через DMX-лінію IC1, можна керувати, змінюючи значення каналів OLA Universe у веб-інтерфейсі.

Вивід Drive enable мікросхем трансивера жорстко підключений до VCC. Другий вихідний порт DMX забезпечується адаптером DMXCreator 512 Basic USB-

to-DMX35. Він підтримується програмним забезпеченням для освітлення DMXCreator.

Щоб перехопити трафік між програмним забезпеченням DMXCreator та USB-адаптером, було встановлено мережевий аналізатор Wireshark з підтримкою USB у Windows.

Файл перехоплення dmxcreator.pcap здійснює такі команди керування:

1. При зміні вихідних DMX-даних (наприклад, при згасанні слайдера) можна спостерігати USB-трафік. Адаптер сам генерує DMX-сигнал з правильною частотою оновлення. Це було перевірено шляхом відключення живлення підключеного приладу, який миттєво відновив правильний колір, коли його знову підключили.

2. При кожній зміні DMX-даних один пакет з постійним рядком байт надсилається на кінцеву точку USB 0x01 пристрою, а потім один або два пакети даних (з 256 байтами корисного навантаження кожен) надсилаються на кінцеву точку 0x02:

a) якщо зміна відбувається в першій половині всесвіту (канали з 1 по 256), байтовий рядок має вигляд 0x80 0x01 0x00 0x00 0x00 0x00 0x01 і надсилається лише один пакет даних;

b) якщо зміна відбувається у другій половині всесвіту (канали з 257 по 512), байтовий рядок має вигляд 0x80 0x01 0x00 0x00 0x00 0x00 0x02 і надсилається два пакети даних.

3. Корисне навантаження пакетів даних складається з половини значень DMX-каналів всесвіту у вигляді одного байтового рядка. Отже, два пакети даних потрібні, якщо змінений канал знаходиться у другій половині.

4. Ідентифікатор виробника USB-пристрою - 0x0a30, ідентифікатор продукту - 0x0002 (міститься у пакеті 88; його також можна отримати, запустивши lsusb у Linux).

3.7 Розширення плагіна OLA's USB DMX Plugin

Реінжиніринговий протокол було включено до плагіна OLA's USB DMX Plugin.

Користувачеві, який запускає olad, потрібно надати дозволи на зв'язок з USB-адаптером.

Цього можна досягти, додавши наступне правило до правил udev у файлі /etc/udev/rules.d/o.rul і перезавантаживши правила. Це дозволить усім членам групи plugdev мати доступ до USB-пристроїв, ідентифікованих за вказаними ідентифікаторами виробника та продукту.

Також було додано правило до файлу debian/ola.udev для OLA, щоб його було включено до випущених пакунків OLA .deb. Під час встановлення цих пакунків правила буде розпаковано у потрібне місце.

Для створення нового підплагіна USB DMX Plugin було модифіковано заводський клас, а саме метод DeviceAdded, який викликається щоразу, коли виявляється новий USB-пристрій.

Цей метод повертає false, якщо пристрій не повинен або не може бути використаний цим субплагіном.

У випадку DMXCreator єдиними ідентифікаційними атрибутами USB-адаптера є ідентифікатори виробника та продукту (Додаток А).

USB-адаптер не повертає серійний номер, тому, оскільки неможливо відрізнити різні пристрої, підтримується лише один пристрій за раз.

У випадку, якщо пристрій має бути затребуваним, метод створює новий екземпляр віджета, передає його методу AddWidget BaseWidgetFactory (розташованому у WidgetFactory.h), який намагається його ініціалізувати, і повертає результат. OLA підтримує виявлення гаряче підключених пристроїв (тобто пристроїв, підключених після запуску OLA) в асинхронному режимі базової бібліотеки libusb за замовчуванням, але також має бути реалізовано резервну версію з використанням синхронних методів libusb. Таким чином, два класи

AsynchronousDMXCreator512Basic та SynchronousDMXCreator512Basic реалізовано як дочірні класи класу віджету DMXCreator512Basic .

Протокол дозволяє надсилати цілий універсум, тому у DMXCreator512Basic.cpp було оголошено константний масив з байтами, які слід надсилати на кінцеву точку USB 0x01.

І синхронна, і асинхронна реалізації DMXCreator512Basic використовують програмний патерн Facade, тобто виклики методів класу віджету передаються через дочірній клас ThreadedUsbSender або AsyncUsbSender відповідно.

Асинхронна версія надає ту саму функціональність, використовуючи асинхронні методи libusb та функції зворотного виклику.

Для ініціалізації необхідно відкрити USB-пристрій та отримати доступ до нього.

За допомогою отриманого дескриптора пристрою було створено дочірній клас ThreadedUsbSender і зберегти вказівник на нього у змінній екземпляра m_sender (Додаток А).

Кожного разу, коли нові DMX-дані мають бути надіслані через цей USB-адаптер, викликається метод SendDMX віджета, який просто пересилає об'єкт DmxBuffer до m_sender, якщо він вже доступний (тобто, якщо функція Init була коректно викликана раніше), нескінченний цикл якого, в свою чергу, викликає TransmitBuffer в наступній ітерації.

TransmitBuffer, наданий DmxBuffer спочатку порівнюється з останнім переданим. Якщо вони однакові, то нічого не потрібно передавати взагалі.

В іншому випадку обидві половини всесвіту копіюються у m_universe_lower та m_universe_upper . Якщо наданий DmxBuffer не містить усіх 512 каналів (змінна length встановлюється рівною кількості скопійованих каналів), то решта заповнюється нулями.

Після цього status_buffer надсилається до кінцевої точки USB 0x01, а обидві половини послідовно надсилаються до кінцевої точки 0x02. Якщо будь-яка

операція завершується невдачею, повертається false, потік зупиняється, а дескриптор пристрою закривається.

У кількох місцях нові класи потрібно було інтегрувати в розроблений USB DMX Plugin.

Розглянемо відповідні класи.

AsyncPluginImpl.cpp є екземпляром фабричного класу в методі Start; перевантажений метод NewWidget AsyncPluginImpl.h є перевантаженим методом NewWidget.

SyncPluginImpl.cpp є екземпляр фабричного класу в конструкторі; перевантажений метод NewWidget SyncPluginImpl.h є перевантаженим методом NewWidget.

SyncronizedWidgetObserver. h [sic] є перевантаженим методом NewWidget.

WidgetFactory.h є перевантаженим віртуальним методом NewWidget.

Крім того, до опису плагіна UsbDmxPlugin.cpp було включено USB-адаптер, а нові файли було додано до Makefile.mk, щоб дозволити перекомпіляцію за допомогою make та make install .

Тепер підключення нового базового USB-пристрою DMXCreator 512 Basic до універсуму у веб-інтерфейсі OLA та надсилання DMX-даних через нього працює належним чином.

3.8 Реалізація OLA Native SPI DMX плагіна

Два вихідних порти надаються плагінами UART та USB DMX, Наступним кроком було розширення плагіну UART, щоб він також підтримував вхід DMX.

Прийняти і розпізнати сигнал BREAK складно, оскільки він пересилається в додаток як нульовий байт і, таким чином, його неможливо відрізнити від каналів даних, які просто встановлені на нуль. У вхідних прапорах c_iflag інтерфейсу termios C є опції для вирішення цієї задачі.

Лістинг системного програмного забезпечення реалізації OLA Native SPI DMX плагіна подано в Додатку А.

Це дозволило отримати можливість декодувавати байти даних. Наступним кроком була реалізація можливості перевірки, чи правильно сигнал був захоплений UART.

Тому необхідним було забезпечити отримання даних, який дозволяє доступ до «сирого» DMX-сигналу, і можливості виконання синтаксичного аналізу для забезпечення повного контролю над ним.

3.9 Бітове зчитування за допомогою бібліотеки pigpio

Наступна кроком було здійснення постійного опитування значення одного виводу GPIO і отриманні таким чином необробленого сигналу.

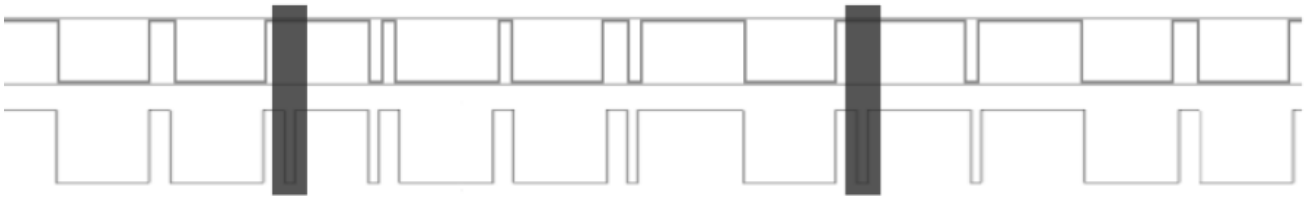
Ця техніка відома як «бітове зчитування» у бібліотеці pigpio42.

Оскільки Linux не є операційною системою реального часу і планування може затримувати операції зчитування так, що вони вже опитують вивід, коли на нього подається наступний біт. На швидкості 250 кбіт/с ці затримки можуть бути значними.

pigpio надає інструмент діагностики під назвою piscope, який відображає сигнал биття бітів.

Тому було згенеровано DMX-дані за допомогою зовнішнього DMX-інтерфейсу і перевірено сигнал, захоплений piscope.

Приклад захопленого зображення разом з очікуваним сигналом показано на рисунку 3.4.



Риунок 3.4. – Сигнали pigpio

Це показало, що часто короткий імпульс, тобто швидкий перехід від низького рівня до високого або навпаки, взагалі не було видно, наприклад, у слотах 4 і 7 на рисунку 3.4.

Крім того, вибірка відбувалася кожні 4 мкс, оскільки стоп-біти слотів (2 старші біти) тривали іноді 4 мкс, іноді 11 мкс, але ніколи не 8 мкс, як очікувалося раніше.

На закінчення, «бітове зчитування» виводу GPIO не є достатньо точним для сигналу DMX зі швидкістю 250 кбіт/с.

3.10 Використання SPI для вибірки DMX

Для реалізації використання SPI для вибірки DMX було використано можливості Raspberry Pi StackExchange43.

Під час дослідження швидкості бітової вибірки GPIO було здійснено вибірку рідного порту SPI для довільних даних DMX.

Для цього лінію DMX було підключено (через шинний трансивер) до виводу MISO (Master In, Slave Out) Raspberry Pi, а інші виводи SPI залишено непідключеними.

Це пов'язано з тим, що SPI розрахований на набагато вищі швидкості, ніж UART, тому стабільна тактова частота дуже важлива.

SPI-контролер Raspberry Pi (який працює як SPI-майстер) має основну частоту 250 МГц, яку можна поділити на будь-яке парне число.

Мета полягає в тому, щоб дискретизувати підключений DMX-сигнал 8 разів на біт, щоб мати достатню толерантність, якщо іноді час дискретизації потрапляє точно на межу, тому необхідна частота дискретизації становить 250 кбіт/с - 8/біт = 2 МГц.

Необхідний дільник тактової частоти 250МГц / 2МГц = 125 є непарним, тому замість нього було використано знвчення 124 (непарні дільники округлюються вниз).

Таким чином, став можливим розбір вибірки бітів, щоб врахувати цю неточність.

Однак, оскільки правильний DMX-приймач повинен приймати будь-який сигнал з бітрейтом від 245 до 255 кбіт/с, гнучкість має бути забезпечена у будь-якому випадку.

3.11 Увімкнення SPI

Увімкнення SPI було реалізовано за допомогою виконаних налаштувань в системному файлі raspi-config.

Для збільшення розміру буфера (тобто кількості байт, які можна отримати/передати за одну операцію) до параметрів ядра у файлі /boot/cmdline.txt слід додати spidev.bufsiz=65536.

Після перезавантаження отриманого значення, повернуте системною командою cat /sys/module/spidev/parameters/bufsiz, повинно відповідати запитуваному 65536.

Для перевірки конфігурації можна з'єднати виводи MISO і MOSI разом для тестування зворотного зв'язку.

Перелік виконаних команд для увімкнення SPI подано в додатку А.

Крім того, було здійснено запуск серії тестів шляхом підключення виводу MISO до +3,3 В та заземлення або залишаючи його непідключеним.

Як і очікувалося, на виході були тільки FF s, тільки 00s і знову тільки 00s, відповідно.

3.12 Отримання даних DMX

На основі використання розробленого системного програмного забезпечення для тестування системи зі зворотним зв'язком, наведеної вище, було реалізовано додаткове системне програмне забезпечення в spi-rec.c, яка здійснює чотириразове

зчитування 8192 байтів з шини SPI MISO і виконує виведенння їх у двійковому форматі на консоль.

Як і тестовий код зі зворотним зв'язком та існуючий плагін SPI, він використовує інтерфейс spidev, який уможливлює зв'язок SPI з користувацького простору (а не як частина ядра).

Одиничне передавання даних через SPI за допомогою spidev налаштовується за допомогою структури spi_ioc_transfer, як показано нижче.

Сама передача виконується за допомогою ioctl(fd,SPI_IOC_MESSAGE(1), &tr).

Лістинг розробленого системного програмного забезпечення для отримання даних DMX подано в Додатку А.

Функція передачі, в якій створюється вищезгадана структура і яка використовується для ініціювання передачі даних SPI, викликається чотири рази поспіль.

Після цього функція printBinary виводить дані з буфера rx у вигляді двійкових чисел. Вона використовує макрос BYTE_TO_BINARY для деконструкції байтів у 8 одиниць та нулів (Додаток А.).

Також було виконано апробацію розробленого системного програмного забезпечення на прикладі DMX-сигналу і збереження отриманих двійкових даних у файлі dm-x-sp-dat.txt.

Потім ці дані були завантажені і побудовані в GNU Octave за допомогою наступних команд. Скріншот отриманого графіка показано на рисунку 3.5.

Така процедура була повторена кілька разів, і графіки демонстрували позитивний результат: не було пропущено жодного короткого імпульсу, а завдяки передискретизації час також був дуже точним.

Єдина проблема, яка була помічена, це те, що два послідовних фрагменти (тобто отримані в результаті декількох операцій передачі) розділені неврахованим пропуском, навіть якщо в системному коді здійснені виклики передачі йдуть один за одним.

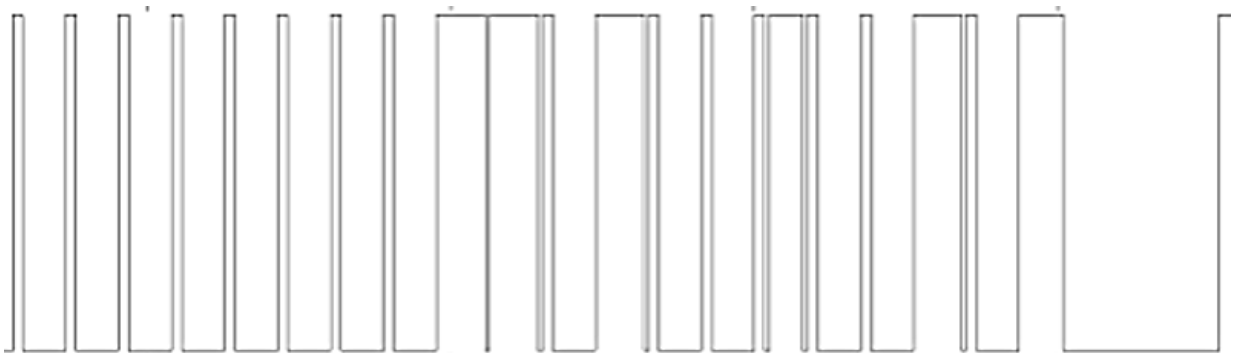


Рисунок 3.5 – Отримані SPI дані

Отримані результати свідчать про те, що результуючі байти не можуть бути проаналізовані як (можливо, нескінченно) довгий потік бітів, а скоріше кожен фрагмент повинен бути проаналізований окремо.

В результаті, деякі фрагменти містять лише початок DMX-пакету, а деякі - лише кінець.

Вони не приносять користі, оскільки незрозуміло, скільки каналів було передано до цього.

Таким чином, частота оновлення нижча для вищих каналів. Проблема візуалізована на рисунку 4.6.

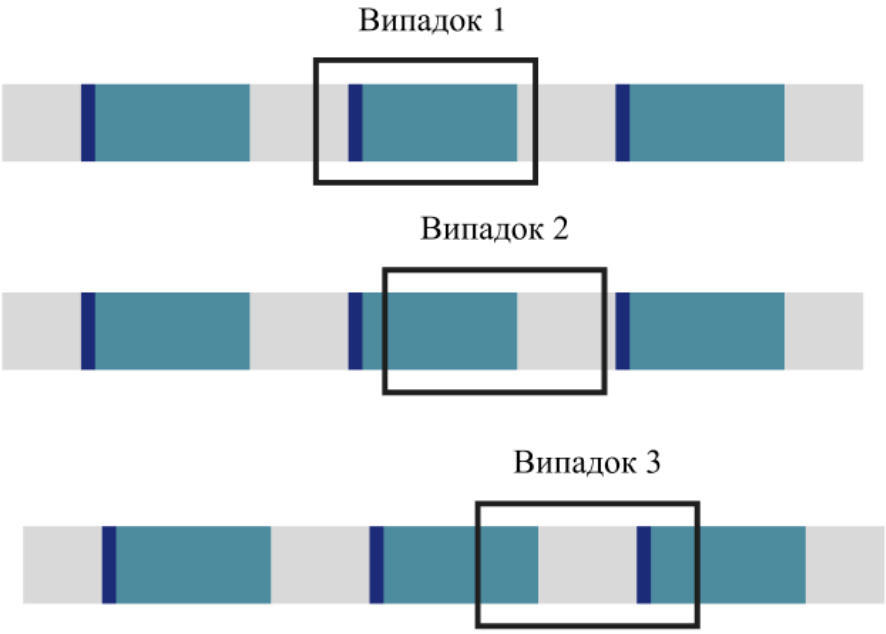


Рисунок 3.6 – Отримані фрагменти SPI у порівнянні з потоком DMX-сигналів

DMX-пакети можуть бути повністю вкладені в один SPI-чанк (випадок 1; це оптимальний випадок) або лише частково.

Якщо міститься лише кінець DMX-пакету (випадок 2), то отриманий фрагмент марний.

Якщо є початок DMX-пакета (випадок 1 і випадок 3), то всі значення наявних каналів до кінця отриманого фрагмента можна співвіднести з відповідними номерами каналів.

Оскільки ймовірність того, що DMX-пакет почнеться на початку фрагмента, менша, ніж десь посередині, вищі канали оновлюються рідше.

3.13 Розбір отриманих фрагментів SPI

Отримання необробленого сигналу вимагає самостійного розбору значень DMX-каналів з відібраних даних. Цей розбір повинен підкорятися часовим обмеженням протоколу DMX.

Розбір отриманих фрагментів SPI здійснювався шляхом розроблення машини станів, яка здатна обробляти вибірку даних біт за бітом, і може переходити до наступного стану, якщо отримані дані відповідають протоколу DMX, і повертається до початкового стану в протилежному випадку.

Блок-схема машини станів зображена на рисунку 3.7.

Після виявлення стартового біта DMX-слота для побудови значення каналу завжди вибирається середина наступних 8 бітів DMX (= 8 «байт SPI»).

Наступні стопові біти вирішують, як діяти далі. Розглянемо типові ситуації виконання.

Обидва стоп-біти мають високий рівень, і далі слідує довільна кількість високих бітів як мітка між слотами / мітка перед BREAK. Потім, на наступному спадаючому фронті, синтаксичний аналізатор зберігає сконструйоване значення DMX-каналу в правильному положенні і продовжує роботу в стані стартового біта даних для наступного слоту.

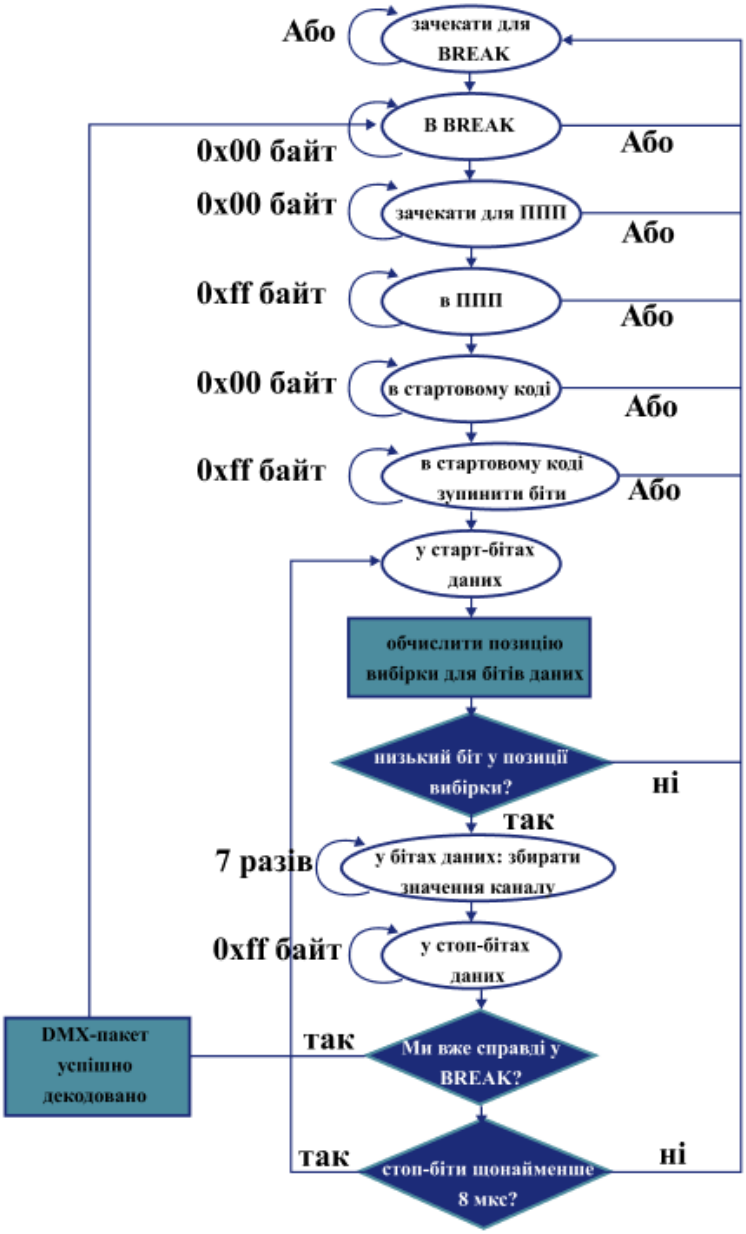


Рисунок 3.7 – Блок-схема машини станів

Винятком є останній канал, де якщо щойно збережене значення DMX було записано в канал 512, то зрозуміло, що більше слотів не може бути, тому DMX-пакет завершується, а стан змінюється на in BREAK.

Існують молодші біти там, де мають бути стоп-біти. Якщо сконструйоване значення каналу також дорівнює нулю, то насправді це був не слот даних, а початок BREAK. Отже, всі значення каналів з цього моменту встановлюються на нуль, DMX-пакет завершується, і автомат переходить у стан BREAK.

Для розпізнавання спадаючих та зростаючих ребер я написав дві допоміжні функції. DetectFallingEdge повертає кількість нулів, якщо переданий байт має вигляд $1^n 0^{8-n}$ (n 2 f0, . . . , 7g}), i -1 в іншому випадку.

Необхідно врахувати, що цей випадок може мати місце, якщо байт містить або тільки одиниці, або випадкові шипи.

DetectRisingEdge працює еквівалентно. Розглянемо функцію WaitForMab як приклад простого обробника стану.

Вона шукає перший передній фронт після BREAK, щоб перейти у стан IN_MAB (Додаток А).

Іншим важливим обробником стану є InDataStartbit, оскільки він обчислює позицію вибірки бітів даних DMX.

Ця позиція вибірки завжди має бути в середині байта, а отже, залежить від state_bitcount, тобто кількості «бітів SPI», які належать до поточного стану, встановленого попереднім обробником стану (тобто InStartcodeStopbits або InDataStopbits). Можливо, середній біт вже містився в останньому обробленому байті, тому в цьому випадку поточний байт скидається до попереднього (Додаток А). Всі інші функції обробників станів можна переглянути у файлі SPIDMXParser.cpp. Метод ParseDmx, якому передається фрагмент SPI для аналізу, ітераційно переглядає байти фрагмента і викликає відповідні обробники станів.

Щоразу, коли виявляється кінець досліджуваного DMX-пакета, здійснюється виклик PacketComplete, який, у свою чергу, викликає функцію зворотного виклику, якщо вона була встановлена раніше за допомогою методу SetCallback у SPIDMXParser.h.

3.14 Обгортання коду у плагін OLA

Останнім кроком було створення нового плагіна OLA, який надає вхідний порт для кожного порту SPI, знайденого у системі.

Потім цей порт може бути виправлений на OLA-універсум, який перенаправляється через Art-Net до програмного забезпечення для освітлення або безпосередньо до вихідного порту DMX.

Плагін OpenSoundControl для OLA містить інформацію про процес його розробки, що був основою для реалізації нового плагіна.

Всі класи системного програмного забезпечення мають префікс SDMX; їхній простір імен, а також назва каталогу - sdmx.

Було взято за основу загальну структуру плагіна на основі плагінів розроблених UART.

Клас Plugin (SDMXPl.h) підключається до інфраструктури плагінів OLA і шукає пристрої SPI.

Для кожного знайденого SPI-пристрою створюється новий OLA-пристрій (SPIDMXDevice.h), який, у свою чергу, створює по одному екземпляру кожного з наступних класів:

Клас Widget (SDMXWg.h), який абстрагує необхідні виклики spidev.

Клас Thread (SDMXTr.h), який багаторазово викликає метод ReadWrite віджету і зберігає отримані дані. Сам потік створює новий парсер (SPIDMXParser.h) для декодування вихідних даних SPI.

Клас InputPort (SPIDMXPort.h) з'єднує нитку з механізмом плагінів OLA, пересилаючи дані DMX і повідомляючи нитку про те, що її було виправлено або не виправлено до/з всесвіту.

Діаграма класів UML на рисунку 3.8 пояснює ці відносини між вищеописаними класами.

Щоразу, коли синтаксичний аналізатор виявляє кінець DMX-пакету, його функція PacketComplete викликає функцію зворотного виклику, яка була встановлена у конструкторі, що викликається з SDMXTh.cpp.

Функція зворотного виклику, у свою чергу, встановлюється у методі PrStUns у SDMXP.h, який завжди викликається при зміні універсуму, до якого підключено порт.

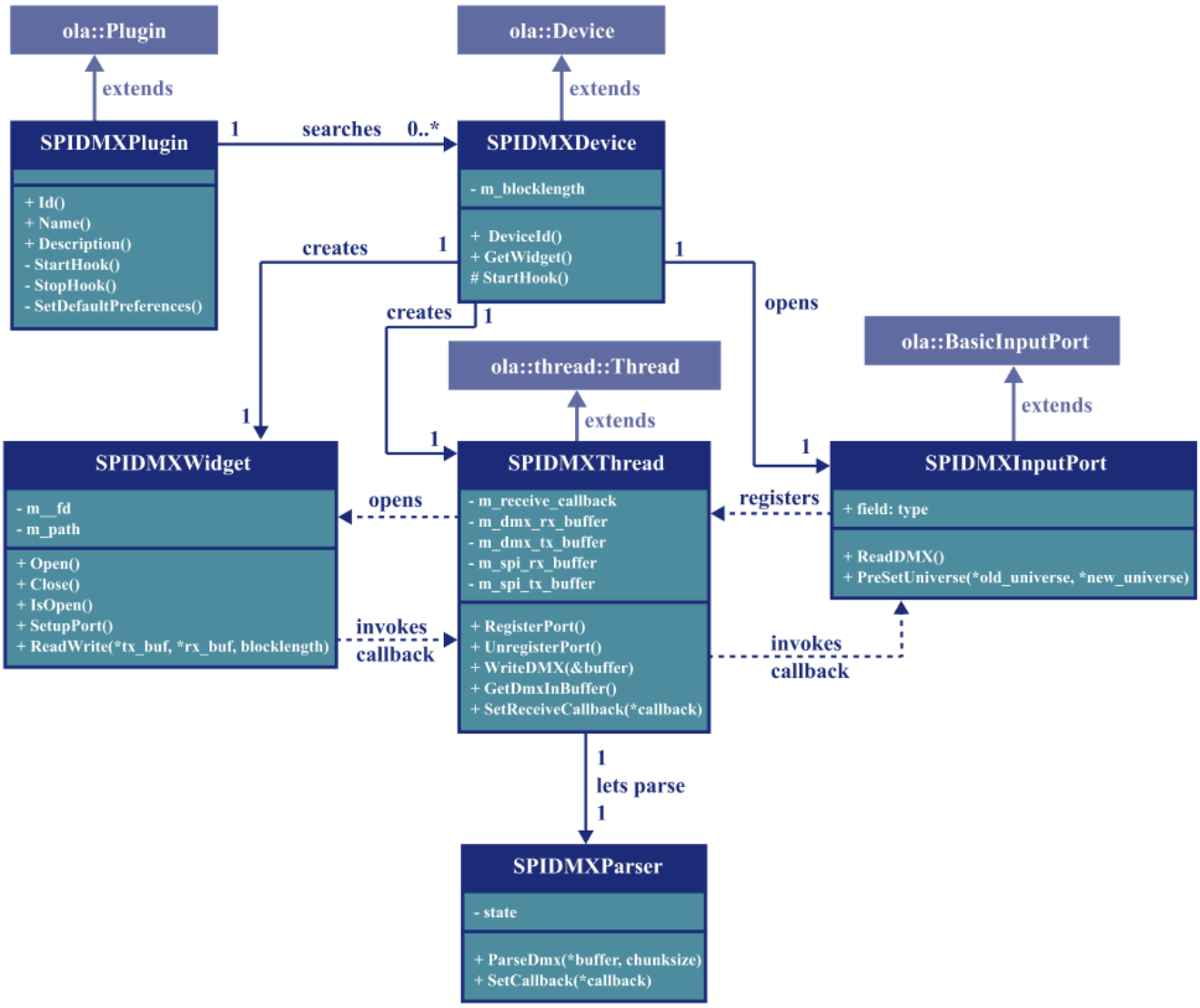


Рисунок 3.8 – Даграма класів UML

Цей ланцюжок викликів потребує особливої уваги, оскільки потрібно пам'ятати, які вказівники все ще можуть посилатися на змінну зворотного виклику, особливо якщо це може бути змінна в іншому потоці. Помилки тут легко призводять до помилок сегментації.

Оскільки плагіни можна вмикати/вимикати під час збирання, інструментарій автоінструментів має бути поінформовано про новий плагін та його залежності від операційної системи (а саме spidev), що було зроблено шляхом додавання одного рядка до configure.ac.

Крім того, з файлу plugins/Makefile.mk було включено Makefile.mk плагіна, який є подібним до Makefile'ів інших плагінів.

Під час виконання, плагін завантажується у olad/DynlnLdr.cpp, де константа збірки USE_SMX визначає, чи був плагін увімкнений під час збірки.

Кожному плагіну призначено константу ідентифікатора плагіна у файлі con/prot/Ol.prt, у цьому випадку OLA_PLUGIN_SPIDMX = 22.

Під час розробки вона тимчасово дорівнювала 99000 і була змінена на остаточне значення незадовго до злиття з основною гілкою.

Після інтеграції нового плагіна в існуючу інфраструктуру знадобилася повна перебудова проекту.

Готовий SPI DMX плагін був протестований з декількома джерелами DMX і працює чудово.

Вищі канали оновлюються рідше, що призводить до більшої затримки.

Отже, їх не слід використовувати для затухання світла, яке має бути плавним за визначенням.

Розмір фрагмента однієї передачі - який має найбільший вплив на це - налаштовується у файлі налаштувань плагіна OLA.

Однак, значення 8к байт видається добрим компромісом між високою ймовірністю включення вищих каналів у фрагмент, з одного боку, і не надто великими неперехопленими проміжками між фрагментами, з іншого боку.

3.15 Збірка шасі

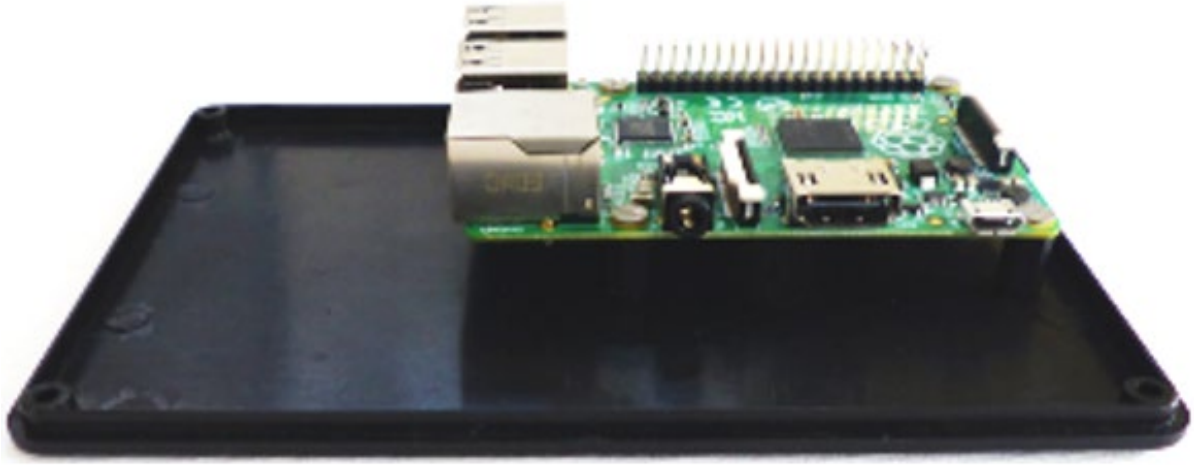
Щоб зробити DMX-інтерфейс надійним і портативним, було розміщено Raspberry Pi 4 разом з платою розширення в пластиковому корпусі.

Всі зовнішні роз'єми (2 роз'єми XLR, 1 роз'єм XLR, 1 роз'єм Ethernet і 1 роз'єм живлення) підключаються за допомогою подовжувачів, що з'єднуються.

Це забезпечує модульну установку і легку заміну несправних деталей.

Щоб встановити USB-адаптер DMXCreator 512 Basic в шасі, необхідно було зняти його корпус, замінити кабель на більш гнучкий і помістити його в термоусадочну трубку.

а)



б)

Рисунок 3.9 – Корпус системи

Оскільки індикаторні світлодіоди Raspberry Pi корисні для того, щоб знати, чи правильно він увімкнувся і завантажився, було встановлено прозорі пластикові шнури, які діють як оптичні хвилеводи.

Реалізація програмно-технічні засоби реалізації pc-dmx інтерфейсу на основі одноплатної комп’ютерної системи raspberry PI подано на рисунку 3.11.

а)

Рисунок 3.10 – Конектори

Рисунок 3.11 - Реалізація програмно-технічні засоби реалізації pc-dmx інтерфейсу на основі одноплатної комп’ютерної системи raspberry PI

3.16 Валідація

Реалізація інтерфейсу PC-DMX - як апаратна, так і програмна - повинна працювати надійно. Це означає, зокрема, що вона повинна відповідати всім заявленим вимогам, за винятком випадків, коли зовнішні обставини, наприклад, втрата живлення, перешкоджають цьому.

Перевірка якості системного програмнго забезпечення та модульні тести можуть допомогти як автоматичні інструменти для забезпечення цього.

Проект Open Lighting Architecture надає тестову інфраструктуру та різні існуючі тести.

Було реалізовано систему юніт-тестів та внесення змін та доповнення до коду, щоб перевірити, чи не відбуваються збої.

Тестування нового системного програмнго забезпечення з реальною апаратною підтримкою є складним процесом.

Сигнал DMX потрібно було відправляти через вихідний порт назад у вхідний порт, щоб перевірити його затримку і цілісність даних.

Таким чином, було здійснено тестування функціональності інтерфейсу PC-DMX, підключивши освітлювальні прилади і порівнявши очікувану і фактичну світловіддачу.

Було виявлено, що інтерфейс був успішно розгорнутий на кількох запусках, які тривали близько 10 годин.

Протягом усього цього часу він працював стабільно, що свідчить про те, що в реалізації системи не було допущено помилок, які б помітно вплинули на його поведінку.

3.17 Якість системного програмнго забезпечення. Виконання вимог

В результаті виконаного проєкту було перевірено задовольняє розроблене системне програмне забезпечення поставленим вимогам.

Це стосується підтримки різних програм керування освітленням з використанням як пропрієтарних, так і відкритих мережевих протоколів, простоти використання через веб-інтерфейс та відкритого вихідного коду. Крім того, розроблене системне програмне забезпечення безпосередньо дозволяє гнучко під'єднувати входи і виходи DMX до універсумів, що дає змогу реалізувати всі варіанти використання. Наразі підтримуються два DMX-виходи та один DMX-вхід, більше можна додати, просто підключивши відповідне обладнання через USB.

Частота вихідного сигналу DMX є задовільною, що було перевірено вручну шляхом підключення світильників за різними DMX-адресами та їх плавного згасання за допомогою програмного забезпечення для управління підключеним освітленням. Вхідний сигнал DMX також приймається досить часто, щоб забезпечити плавне згасання приблизно для нижніх чотирьохсот каналів.

Вартість Raspberry Pi 4 Model B+, карти пам'яті microSD на 8 ГБ, друкованої плати, електронних компонентів, шасі, штекерних роз'ємів і блоку живлення становить близько 4000 грн. USB-адаптер DMXCreator входить до комплекту поставки.

3.6. Висновки до третього розділу

У третьому розділі реалізовано програмно-апаратну частину проєкту. Детально описано процес налаштування Raspberry Pi4, встановлення й конфігурації OLA, реалізацію електромонтажу та інтеграцію з USB- і SPI-пристроями. Розроблено механізм обробки вхідного і вихідного DMX-сигналу, зокрема, використано бітове зчитування та реалізацію машини станів для декодування SPI-даних.

Було доведено можливість масштабування та ефективності системи, включаючи підтримку кількох DMX-виходів і можливість розширення шляхом додавання нових пристроїв через USB.

Система забезпечує стабільне й точне керування освітленням на рівні до 400 каналів, підтверджене валідаційними випробуваннями.

ВИСНОВКИ

У роботі за результатами виконаних теоретичних та практичних досліджень було розроблено і реалізовано програмно-технічні засоби реалізації PC-DMX інтерфейсу на основі одноплатної комп'ютерної системи raspberry PI. Теоретичні дослідження та практична реалізація довели ефективність обраного підходу до обробки і керування інформаційними потоками з використанням цифрових інтерфейсів на основі Raspberry Pi4 та протоколу DMX512.

Система характеризується високою гнучкістю, масштабованістю та відкритістю архітектури, що дозволяє адаптувати її до різних сценаріїв застосування. Результати проекту демонструють практичну придатність і технічну зрілість запропонованого рішення, що може бути використане як у військових, так і в цивільних задачах моніторингу та автоматизованого керування.

У першому розділі проведено аналіз сучасних підходів до цифрового керування освітленням, зосереджений на застосуванні PC-DMX інтерфейсу. Було обґрунтовано доцільність використання протоколу DMX512 як основи для реалізації гнучкого й масштабованого керування світловими пристроями. Визначено переваги його універсальності, відкритості, можливості інтеграції з новітніми протоколами, економічності та легкості у використанні. Окремо розглянуто особливості застосування Raspberry Pi як апаратної платформи для створення доступного та ефективного програмно-технічного рішення, що підтримує кастомізацію та дистанційне налаштування через веб-інтерфейс.

У другому розділі проведено проведено системне проектування кіберфізичної системи адаптивного застосування моніторингових елементів. Визначено її апаратну та програмну архітектуру, описано компоненти системи та їхню взаємодію. Зокрема, проаналізовано реалізацію передачі даних через UART, USB та SPI, застосування Open Lighting Architecture (OLA) як системного ПЗ, а також адаптацію плагінів для підтримки специфічних каналів вводу/виводу.

У третьому розділі реалізовано програмно-апаратну частину проєкту. Детально описано процес налаштування Raspberry Pi4, встановлення й конфігурації OLA, реалізацію електромонтажу та інтеграцію з USB- і SPI-пристроями. Розроблено механізм обробки вхідного і вихідного DMX-сигналу, зокрема, використано бітове зчитування та реалізацію машини станів для декодування SPI-даних. Було доведено можливість масштабування та ефективності системи, включаючи підтримку кількох DMX-виходів і можливість розширення шляхом додавання нових пристроїв через USB. Система забезпечує стабільне й точне керування освітленням на рівні до 400 каналів, підтверджене валідаційними випробуваннями.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Bodrogi P, Brückner S, Khanh TQ, Winkler H. Visual assessment of light source color quality. *Color Research and Application*. 2013. Vol. 38. pp. 4–13.

2. Dangol R, Islam M, Hyvärinen M, Bhusal P, Puolakka M, Halonen L. Subjective preferences and colour quality metrics of LED light sources. *Lighting Research and Technology*. 2013. Vol. 45. pp. 666–688.

3. Davis W, Ohno Y. Color quality scale. *Optical Engineering*. 2010. Vol. 49. pp. 1–16.

4. Fotios S. Lighting in offices: lamp spectrum and brightness. *Coloration Technology*. 2011. Vol. 127. pp. 114–120.

5. Freyssinier J, Rea M. A two-metric proposal to specify the color-rendering properties of light sources for retail lighting. *Proceeding of SPIE*. 2010. Vol. 7784. pp. 77840V–1.

6. Freyssinier J, Rea M. A two-metric proposal to specify the color-rendering properties of light sources for retail lighting. *Proceeding of SPIE*. 2010. Vol. 7784. pp. 77840V–1.

7. Islam M, Dangol R, Hyvärinen M, Bhusal P, Puolakka M, Halonen L. Investigation of user preferences for LED lighting in terms of light spectrum. *Lighting Research and Technology*. 2013. Vol. 45. pp. 641–665.

8. Ju J, Dahua C, Yandan L. Effects of correlated color temperature on spatial brightness perception. *Color Research and Application*. 2012. Vol. 37. pp. 450–454.

9. Linhart F, Scartezzini J-L. Evening office lighting – visual comfort vs. energy efficiency vs. performance? *Journal of Building and Environment*. 2011. Vol. 46. pp. 981–989.

10. Lin R.F, Chou C, Wang Y.T, Tu H.W. Effects of LED color temperature on office workers. *Proceedings of the 2nd Southeast Asian Network of Ergonomics Societies Conference*, Langkawi, Malaysia. 2012. July 9–12.

11. Pousset N, Obein G, Razet A. Visual experiment on LED lighting quality with color quality scale colored sample. *Proceedings of CIE 2010: Lighting Quality and Energy Efficiency*, Vienna, Austria. 2010. March 14–17. pp. 722–729.

12. Rea M.S, Freyssinier J.P. Color rendering: Beyond pride and prejudice. *Color Research and Application*. 2010. Vol. 35. pp. 401–409.

13. Shakir I, Narendran N. White LEDs in landscape lighting application. *Solid State Lighting II, Proceedings of SPIE*. 2002. Vol. 4776. Retrieved 19 May 2013.

14. Smet K, Ryckaert W, Pointer M, Deconinck G, Hanselaer P. Memory colours and colour quality evaluation of conventional and solid-state lamps. *Optics Express*. 2010. Vol. 18. pp. 26229–26244.

15. Spaulding J.M, Maria R.T, Robert E.L. Human preference in tunable solid state lighting. *Proceeding SPIE*. 2011. Vol. 7954. pp. 795403–795403. doi:10.1117/12.876301. Retrieved 23 May 2013. https://doi.org/10.1117/12.876301.

16. Szabo F, Schanda J, Bodrogi P, Radkov E. A comparative study of new solid state light sources. Retrieved 23 May 2013. http://cie2.nist.gov/TC1-69/whiteled-manuscript-ver8.pdf.

17. Chavan M.S., Patil V.P., Chavan S., Sana S., Shinde C. Design and implementation of IoT-based real-time monitoring system for aquaculture using Raspberry Pi. *International Journal on Recent and Innovation Trends in Computing and Communication*. 2018. Vol. 6. pp. 159–161.

18. Atiqur R., Li Y. Automated smart car parking system using raspberry Pi 4 and iOS application. *International Journal of Reconfigurable and Embedded Systems (IJRES)*. 2020. Vol. 9. pp. 229–234.

19. Zhao C.W., Jegatheesan J., Loon S.C. Exploring IoT application using Raspberry Pi. *International Journal of Computer Networks and Applications*. 2015. Vol. 2. pp. 27–34.

20. Noor M.b.M., Hassan W.H. Current research on internet of things (IoT) security: A survey. *Computer Networks*. 2019. Vol. 148. pp. 283–294.

21. Pawar P., Kumar M.T., Vittal K.P. An IoT based intelligent smart energy management system with accurate forecasting and load strategy for renewable generation. *Measurement*. 2020. Vol. 152.

22. Abate F., Carratù M., Liguori C., Paciello V. A low cost smart power meter for IoT. *Measurement*. 2019. Vol. 136. pp. 59–66.

23. Sridharan M., Devi R., Dharshini C.S., Bhavadarani M. IoT based performance monitoring and control in counter flow double pipe heat exchanger. *Internet of Things*. 2019. Vol. 5. pp. 34–40.

24. Pereira R.I.S., Dupont I.M., Carvalho P.C.M., Jucá S.C.S. IoT embedded Linux system based on Raspberry Pi applied to real-time cloud monitoring of a decentralized photovoltaic plant. *Measurement*. 2018. Vol. 114. pp. 286–297.

25. Perumal V.S.A., Baskaran K., Rai S.K. Implementation of effective and low-cost building monitoring system (BMS) using Raspberry Pi. *Energy Procedia*. 2017. Vol. 143. pp. 179–185.

26. Arasteh H., et al. IoT-based smart cities: A survey. *2016 IEEE 16th International Conference on Environment and Electrical Engineering (EEEIC)*. 2016. pp. 1–6.

27. Li Y., Chu L., Zhang Y., Guo C., Fu Z., Gao J. Intelligent transportation video tracking technology based on computer and image processing technology. *Journal of Intelligent & Fuzzy Systems*. 2019. Vol. 37. pp. 3347–3356.

28. Saarika P.S., Sandhya K., Sudha T. Smart transportation system using IoT. *2017 International Conference On Smart Technologies For Smart Nation (SmartTechCon)*. 2017. pp. 1104–1107.

29. Vidyasagaran S., Devi S.R., Varma A., Rajesh A., Charan H. A low cost IoT based crowd management system for public transport. *2017 International Conference on Inventive Computing and Informatics (ICICI)*. 2017. pp. 222–225.

30. Shete R., Agrawal S. IoT based urban climate monitoring using Raspberry Pi. *2016 International Conference on Communication and Signal Processing (ICCSP)*. 2016. pp. 2008–2012.

31. Khan H.A., Abdulla R., Selvaperumal S.K., Bathich A. IoT based on secure personal healthcare using RFID technology and steganography. *International Journal of Electrical and Computer Engineering (IJECE)*. 2021. Vol. 11. pp. 3300–3309.

32. Kumar R., Rajasekaran M.P. An IoT based patient monitoring system using raspberry Pi. *2016 International Conference on Computing Technologies and Intelligent Data Engineering (ICCTIDE'16)*. 2016. pp. 1–4.

33. Kaur A., Jasuja A. Health monitoring based on IoT using Raspberry PI. *2017 International Conference on Computing, Communication and Automation (ICCCA)*. 2017. pp. 1335–1340.

34. Rohit S.L., Tank B.V. IoT based health monitoring system using Raspberry Pi - review. *2018 Second International Conference on Inventive Communication and Computational Technologies (ICICCT)*. 2018. pp. 997–1002.

35. Koshti M., Ganorkar S., Chiari L. IoT-based health monitoring system by using raspberry Pi and ECG signal. *International Journal of Innovative Research in Science, Engineering and Technology*. 2016. Vol. 5. pp. 8977–8985.

36. Tetila E.C., et al. Automatic recognition of soybean leaf diseases using UAV images and deep convolutional neural networks. *IEEE Geoscience and Remote Sensing Letters*. 2020. Vol. 17. pp. 903–907.

37. Junfitrhana A.P., Langlangbuana M.L., Fatah W.A., Susilawati. Developing potential agriculture land detector for determine suitable plant using Raspberry-Pi. *2017 International Conference on Computing, Engineering, and Design (ICCED)*. 2017. pp. 1–4.

38. Krishna K.L., Silver O., Malende W.F., Anuradha K. Internet of things application for implementation of smart agriculture system. *2017 International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)*. 2017. pp. 54–59.

39. Venkatesan R., Tamilvanan A. A sustainable agricultural system using IoT. *2017 International Conference on Communication and Signal Processing (ICCSP)*. 2017. pp. 763–767.

40. Sekaran K., Meqdad M.N., Kumar P., Rajan S., Kadry S. Smart agriculture management system using internet of things. *TELKOMNIKA (Telecommunication Computing Electronics and Control)*. 2020. Vol. 18. Art. no. 1275.

41. Lova Raju K., Vijayaraghavan V. IoT technologies in agricultural environment: A survey. *Wireless Personal Communications*. 2020. Vol. 113. pp. 2415–2446.

42. Stoyanova M., Nikoloudakis Y., Panagiotakis S., Pallis E., Markakis E.K. A survey on the internet of things (IoT) forensics: challenges, approaches, and open issues. *IEEE Communications Surveys & Tutorials*. 2020. Vol. 22. pp. 1191–1221.

43. Shah S.H., Yaqoob I. A survey: Internet of things (IoT) technologies, applications and challenges. *2016 IEEE Smart Energy Grid Engineering (SEGE)*. 2016. pp. 381–385.

44. Burhan M., Rehman R., Khan B., Kim B.-S. IoT elements, layered architectures and security issues: A comprehensive survey. *Sensors*. 2018. Vol. 18.

45. Ibrahim S.N., Basri A.H.H., Asnawi A.L. Development of web-based surveillance system for internet of things (IoT) application. *Bulletin of Electrical Engineering and Informatics (BEEI)*. 2019. Vol. 8. pp. 1108–1116.

46. Kumar V.S., Ashish S.N., Gowtham I.V., Balaji S.P.A., Prabhu E. Smart driver assistance system using Raspberry Pi and sensor networks. *Microprocessors and Microsystems*. 2020. Vol. 79.

47. Jabbar W.A., Wei C.W., Azmi N.A.A.M., Haironnazli N.A. An IoT Raspberry Pi-based parking management system for smart campus. *Internet of Things*. 2021. Vol. 14.

48. Jaiswal S., Sharma D.K., Jaiswal T., Basumatary B., Tiwari M., Tiwari T. Real time analysis of intelligent placing system for vehicles using IoT with deep learning. *Materials Today: Proceedings*. 2022. Vol. 51. pp. 339–343.

49. Herrera-Quintero L.F., Vega-Alfonso J.C., Banse K.B.A., Zambrano E.C. Smart ITS sensor for the transportation planning based on IoT approaches using serverless and microservices architecture. *IEEE Intelligent Transportation Systems Magazine*. 2018. Vol. 10. pp. 17–27.

50. Prabu A.V, Tolada A, Mishra J, Rajasoundaran S, Deepak T. Automatic vehicle parking space booking system using IoT. *Materials Today: Proceedings*. 2021.

51. Patil P. Smart IoT based system for vehicle noise and pollution monitoring. *2017 International Conference on Trends in Electronics and Informatics (ICEI)*. 2017. pp. 322–326.

52. Bansal K, Mittal K, Ahuja G, Singh A, Gill S.S. DeepBus: Machine learning based real time pothole detection system for smart transportation using IoT. *Internet Technology Letters*. 2020. Vol. 3. pp. 77-84.

53. Godavarthi B, Nalajala P, Ganapuram V. Design and implementation of vehicle navigation system in urban environments using internet of things (IoT). *IOP Conference Series: Materials Science and Engineering*. 2017. Vol. 225. pp. 41-52.

54. Husni E. Driving and fuel consumption monitoring with internet of things. *International Journal of Interactive Mobile Technologies (iJIM)*. 2017. Vol. 11. pp. 78–97.

55. Alluhaidan A.S, Alluhaidan M.S, Basheer S. Internet of things based intelligent transportation of food products during COVID. *Wireless Personal Communications*. 2021. Vol. 3. pp. 12-31.

56. Ishak A.H, Hajjaj S.S.H, Gsangaya K.R, Sultan M.T.H, Mail M.F, Hua L.S. Autonomous fertilizer mixer through the internet of things (IoT). *Materials Today: Proceedings*. 2021.

57. Arshad J. Intelligent greenhouse monitoring and control scheme: An arrangement of Sensors, Raspberry Pi based embedded system and IoT platform. *Indian Journal of Science and Technology*. 2020. Vol. 13. pp. 2811–2822.

58. Benyezza H., Bouhedda M., Rebouh S. Zoning irrigation smart system based on fuzzy control technology and IoT for water and energy saving. *Journal of Cleaner Production*. 2021. Vol. 302. pp. 127001.

59. Lavanya G., Rani C., Ganeshkumar P. An automated low cost IoT based fertilizer intimation system for smart agriculture. *Sustainable Computing: Informatics and Systems*. 2020. Vol. 28. pp. 01.002.

60. Abioye E.A., et al. IoT-based monitoring and data-driven modelling of drip irrigation system for mustard leaf cultivation experiment. *Information Processing in Agriculture*. 2021. Vol. 8. No. 2. pp. 270–283.

61. Goap A., Sharma D., Shukla A.K., Krishna C.R. An IoT based smart irrigation management system using machine learning and open source technologies. *Computers and Electronics in Agriculture*. 2018. Vol. 155. pp. 41–49.

62. Ramli M.R., Daely P.T., Kim D.-S., Lee J.M. IoT-based adaptive network mechanism for reliable smart farm system. *Computers and Electronics in Agriculture*. 2020. Vol. 170. pp. 105287.

63. Jesudoss A., Jacob Daniel M., Jerom Richard J. Intelligent medicine management system and surveillance in IoT environment. *IOP Conference Series: Materials Science and Engineering*. 2019. Vol. 590. pp. 012005.

64. Alarcón-Paredes A., Francisco-García V., Guzmán-Guzmán I.P., Cantillo-Negrete J., Cuevas-Valencia R.E., Alonso-Silverio G.A. An IoT-based non-invasive glucose level monitoring system using Raspberry Pi. *Applied Sciences*. 2019. Vol. 9. No. 15. pp. 3046.

65. Moghadas E., Rezazadeh J., Farahbakhsh R. An IoT patient monitoring based on fog computing and data mining: Cardiac arrhythmia usecase. *Internet of Things*. 2020. Vol. 11. pp. 100251.

66. Bhatia M., Kaur S., Sood S.K., Behal V. Internet of things-inspired healthcare system for urine-based diabetes prediction. *Artificial Intelligence in Medicine*. 2020. Vol. 107. pp. 101913.

67. Yacchirema D., Sarabia-Jácome D., Palau C.E., Esteve M. System for monitoring and supporting the treatment of sleep apnea using IoT and big data. *Pervasive and Mobile Computing*. 2018. Vol. 50. pp. 25–40.

68. Lomotey R.K., Pry J., Sriramoju S. Wearable IoT data stream traceability in a distributed health information system. *Pervasive and Mobile Computing*. 2017. Vol. 40. pp. 692–707.

69. Mano L.Y., et al. Exploiting IoT technologies for enhancing health smart homes through patient identification and emotion recognition. *Computer Communications*. 2016. Vol. 89–90. pp. 178–190.

Додаток А

(обов'язковий)

СИСТМНЕ ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

Збірка та інсталяція плагіну OLA

```
sudo apt install git libcppunit-dev libcppunit-1.13-0 uuid-dev
pkg-config  -
libncurses5-dev libtool autoconf automake g++ libmicrohttpd-dev
-
libmicrohttpd10 protobuf-compiler libprotobuf-lite9 python-
protobuf  -
libprotobuf-dev libprotoc-dev zlib1g-dev bison flex make
libftdi-dev  -
libftdi1 libusb-1.0-0-dev liblo-dev libavahi-client-dev
git clone https://github.com/OpenLightingProject/ola.git
cd ola
autoreconf -i
./configure
make
sudo make install
sudo ldconfig
```

Виконуваний скрипт

```
sudo mv init-olad.sh /etc/init.d/olad
sudo chmod a+x /etc/init.d/olad
sudo update-rc.d olad defaults
```

Фрагмент init-olad.sh

```
/sbin/start-stop-daemon --start --background --make-pidfile --
pidfile $PIDFILE  -
--umask 0002 --chuid $USER --exec $DAEMON -- $DAEMON_ARGS
 # set GPIO24 high (drive enable of IC1) and GPIO16 low (drive
enable of IC2)
echo "24" > /sys/class/gpio/export
 echo "16" > /sys/class/gpio/export
 sleep 1
```

```
 echo "out" > /sys/class/gpio/gpio24/direction
 echo "out" > /sys/class/gpio/gpio16/direction
 sleep 1
 echo "1" > /sys/class/gpio/gpio24/value
 echo "0" > /sys/class/gpio/gpio16/value
```

Скрипт плагін UART OLA

```
enabled = true
device = /dev/ttyAMA0
/dev/ttyAMA0-break = 100
/dev/ttyAMA0-malf = 100
```

Скрипт налаштування USB

```
sudo wget -O /etc/udev/rules.d/10-ola.rules
https://raw.githubusercontent.com/ -
OpenLightingProject/raspberrypi/master/etc/udev/rules.d/10-
local.rules
sudo udevadm control --reload-rules
```

Скрипт мережевихналаштувань

```
static ip
interface eth0
static ip_address=192.168.0.10/24
static routers=192.168.0.1
static domain_name_servers=192.168.0.1
```

Скрипт використання порту 80 для веб-сервера

```
sudo sysctl -w net.ipv4.ip_forward=1
sudo sysctl -w net.ipv4.conf.all.route_localnet=1
sudo iptables -A PREROUTING -t nat -i eth0 -p tcp --dport 80 -j
DNAT --to  -
127.0.0.1:9090
sudo mkdir /etc/iptables
sudo sh -c "iptables-save > /etc/iptables/rules.v4"
```

Розширення плагіна OLA's USB DMX Plugin

```
const uint16_t DMXCreator512BasicFactory::VENDOR_ID = 0x0a30;
const uint16_t DMXCreator512BasicFactory::PRODUCT_ID = 0x0002;
bool DMXCreator512BasicFactory::DeviceAdded(
WidgetObserver *observer,
libusb_device *usb_device,
const struct libusb_device_descriptor &descriptor) {
if (descriptor.idVendor != VENDOR_ID || descriptor.idProduct !=
PRODUCT_ID) {
return false;
 }
LibUsbAdaptor::DeviceInformation info;
 if (!m_adaptor->GetDeviceInfo(usb_device, descriptor, &info)) {
return false;
 }
 OLA_INFO << "Found a new DMXCreator 512 Basic device";

if (info.serial.empty()) {
 if (m_missing_serial_number) {
 OLA_WARN << "We can only support one device without a serial
number.";
 return false;
 } else {
 m_missing_serial_number = true;
 }
 }
```

Асинхронна реалізації DMXCreator512Basic

```
bool SynchronousDMXCreator512Basic::Init() {
 libusb_device_handle *usb_handle;
 bool ok = m_adaptor->OpenDeviceAndClaimInterface(
 m_usb_device, 0, &usb_handle);
 if (!ok) {
 return false;
 }
 std::auto_ptr<DMXCreator512BasicThreadedSender> sender(
 new DMXCreator512BasicThreadedSender(m_adaptor, m_usb_device,
 usb_handle));
 if (!sender->Start()) {
 return false;
 }
 m_sender.reset(sender.release());
 return true;
```

```
 }
unsigned int length = CHANNELS_PER_PACKET;
 m_dmx_buffer.Get(m_universe_data_lower, &length);
 memset(m_universe_data_lower + length, 0, CHANNELS_PER_PACKET -
length);
 length = CHANNELS_PER_PACKET;
 m_dmx_buffer.GetRange(CHANNELS_PER_PACKET,
m_universe_data_upper, &length);
 memset(m_universe_data_upper + length, 0, CHANNELS_PER_PACKET -
length);
bool SynchronousDMXCreator512Basic::SendDMX(const DmxBuffer
&buffer) {
return m_sender.get() ? m_sender->SendDMX(buffer) : false;
}
bool DMXCreator512BasicThreadedSender::TransmitBuffer(
libusb_device_handle *handle, const DmxBuffer &buffer) {
if (m_dmx_buffer == buffer) {
// no need to update -> sleep 50µs to avoid timeout errors
usleep(50);
return true;
}
m_dmx_buffer = buffer;
```

Лістинг системного програмного забезпечення реалізації OLA Native SPI DMX плагіна

```
BRKINT If IGNBRK is set, a BREAK is ignored. If it is not set
but
BRKINT is set, then a BREAK causes the input and output queues
to be flushed, and if the terminal is the controlling terminal
of a foreground process group, it will cause a SIGINT to be
sent to this foreground process group. When neither IGNBRK
nor BRKINT are set, a BREAK reads as a null byte ('\0'),
except when PARMRK is set, in which case it reads as the
sequence \377 \0 \0.
IGNPAR Ignore framing errors and parity errors.
PARMRK If this bit is set, input bytes with parity or framing
errors
are marked when passed to the program. This bit is meaningful
only when INPCK is set and IGNPAR is not set. The way erro-
neous bytes are marked is with two preceding bytes, \377 and
\0. Thus, the program actually reads three bytes for one
erroneous byte received from the terminal. If a valid byte
```

```
has the value \377, and ISTRIP (see below) is not set, the
program might confuse it with the prefix that marks a parity
error. Therefore, a valid byte \377 is passed to the program
as two bytes, \377 \377, in this case.
If neither IGNPAR nor PARMRK is set, read a character with a
parity error or framing error as \0.
INPCK Enable input parity checking.
ISTRIP Strip off eighth bit.
```

Перелік виконаних команд для увімкнення SPI.

```
wget https://raw.githubusercontent.com/raspberrypi/linux/rpi-
3.10.y/ -
Documentation/spi/spidev_test.c
gcc -o spidev_test spidev_test.c
./spidev_test --device /dev/spidev0.0 --speed 2000000
```

Результат:

```
spi mode: 0
bits per word: 8
max speed: 2000000 Hz (2000 KHz)
FF FF FF FF FF FF
40 00 00 00 00 95
FF FF FF FF FF FF
FF FF FF FF FF FF
FF FF FF FF FF FF
DE AD BE EF BA AD
F0 0D
```

Лістинг розробленого системного програмного забезпечення для отримання даних DMX

```
struct spi_ioc_transfer tr = {
// don't transmit anything
.tx_buf = 0,
// save received bytes into `rx` buffer (at appropriate offset)
.rx_buf = (unsigned long)(rx + BYTES_PER_TRANSFER*offset),
// bytes to send/receive in this transfer operation
.len = BYTES_PER_TRANSFER,
 // don't delay after data bytes are sent
54 .delay_usecs = delay,
// overwrite speed temporarily to 2MHz
```

```
.speed_hz = speed,
// overwrite bits per word temporarily to 8
.bits_per_word = bits_per_word,
 };
```

Деконструкція байтів у 8 одиниць та нулів

```
#define BYTE_TO_BINARY_PATTERN "%c %c %c %c %c %c %c %c "
#define BYTE_TO_BINARY(byte) \
(byte & 0x80 ? '1' : '0'), \
(byte & 0x40 ? '1' : '0'), \
(byte & 0x20 ? '1' : '0'), \
(byte & 0x10 ? '1' : '0'), \
 (byte & 0x08 ? '1' : '0'), \
 (byte & 0x04 ? '1' : '0'), \
 (byte & 0x02 ? '1' : '0'), \
(byte & 0x01 ? '1' : '0')
```

Функція WaitForMab

```
void SPIDMXParser::WaitForMab() {
uint8_t byte = chunk[chunk_bitcount];
if (byte != 0) {
int8_t ones = DetectRisingEdge(byte);
if (ones > 0) {
ChangeState(IN_MAB);
state_bitcount = ones;
} else {
ChangeState(WAIT_FOR_BREAK);
}
}
chunk_bitcount++;
}
```

Функція InDataStartbit()

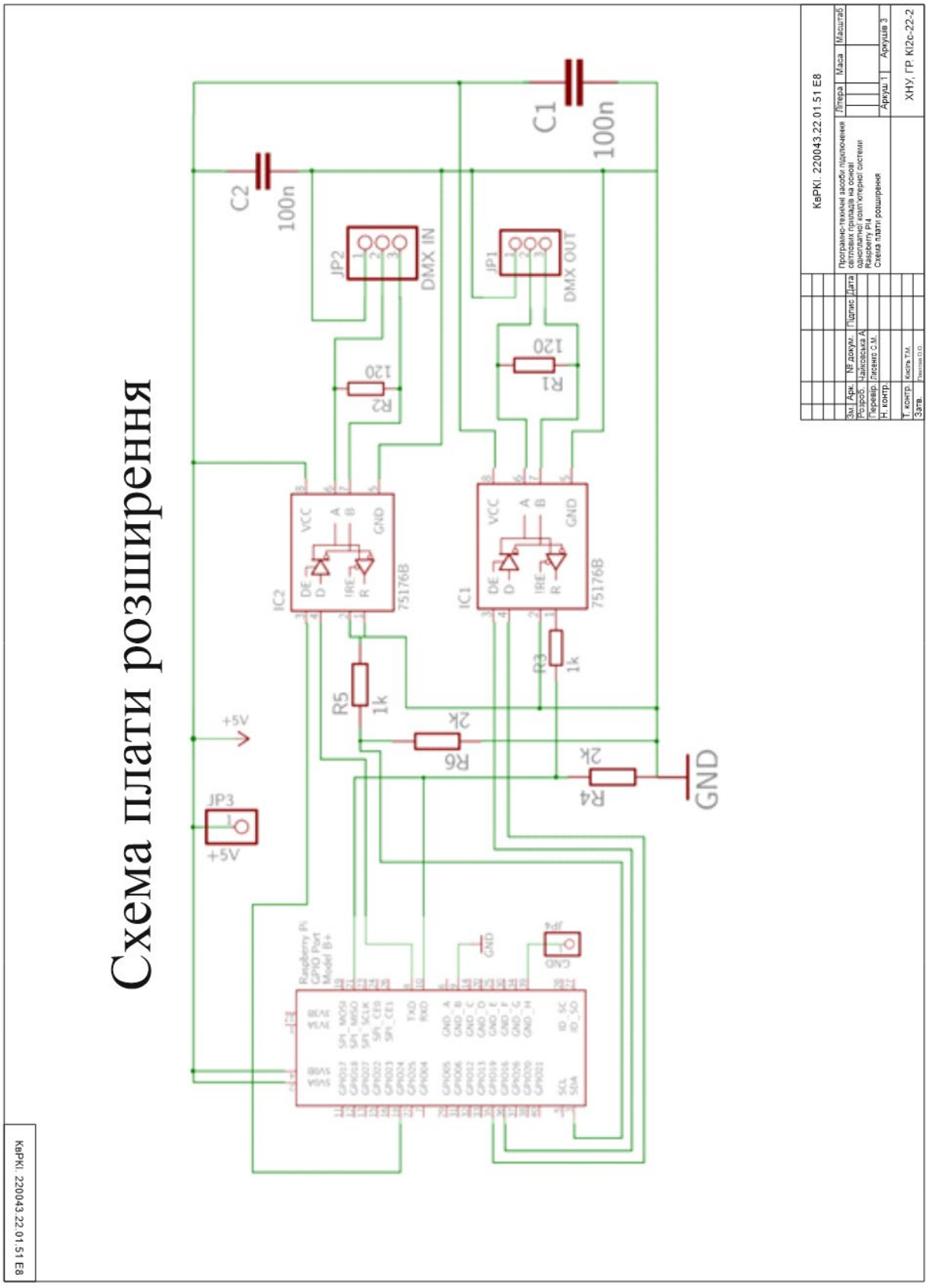
```
void SPIDMXParser::InDataStartbit() {
uint8_t byte = chunk[chunk_bitcount];
if (state_bitcount >= 4) {
// look at the last byte again and don't increase chunk_bitcount
byte = chunk[chunk_bitcount - 1];
sampling_position = state_bitcount - 4;
} else {
// next byte will be handled in next step as usual
chunk_bitcount++;
```

```
sampling_position = state_bitcount + 8 - 4;
}
// start bit must be zero
if ((byte & (1 << sampling_position))) {
ChangeState(WAIT_FOR_BREAK);
} else {
current_dmx_value = 0x00;
ChangeState(IN_DATA_BITS);
}
}
```

Додаток Б

(обов’язковий)

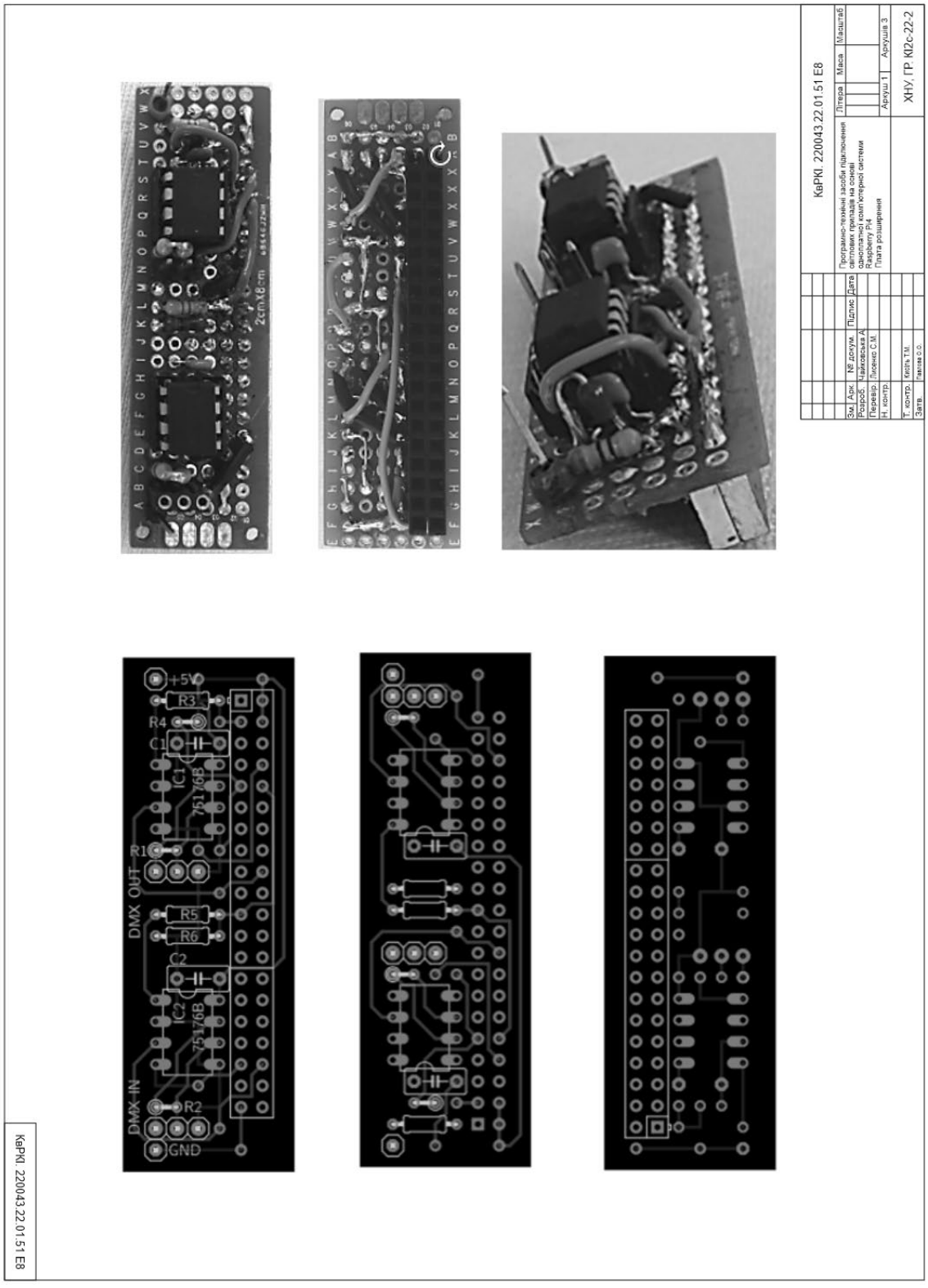
КОПІЯ КРЕСЛЕННЯ «СХЕМА ПЛАТИ РОЗШИРЕННЯ»



Додаток В

(обов’язковий)

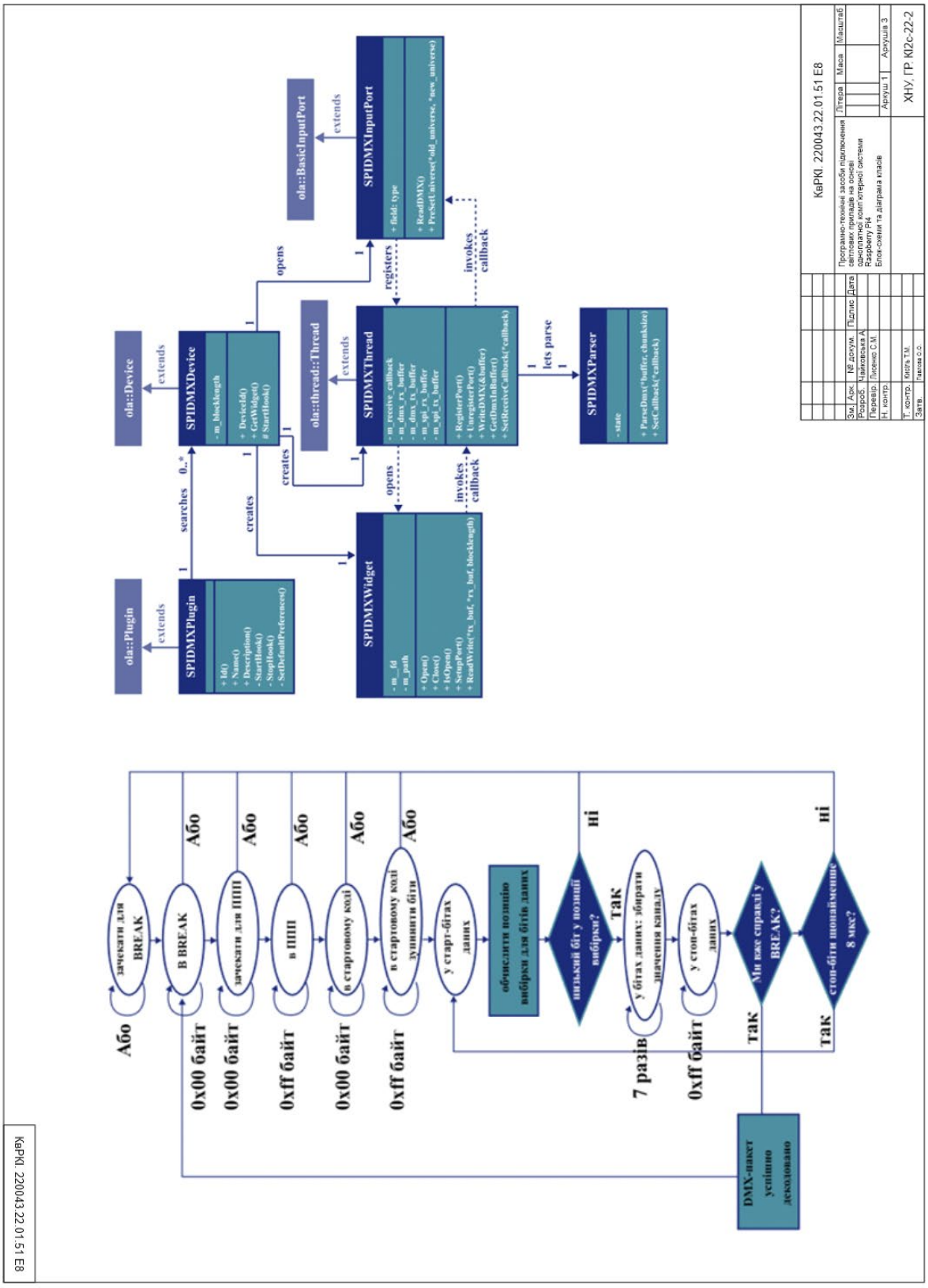
КОПІЯ КРЕСЛЕННЯ «ПЛАТА РОЗШИРЕННЯ»



Додаток Г

(обов’язковий)

КОПІЯ КРЕСЛЕННЯ «БЛОК-СХЕМИ ТА ДІАГРАМА КЛАСІВ»



Протокол аналізу звіту подібності експертом

Заявляю, що я ознайомився (-лась) з Повним звітом подібності, який був згенерований Системою виявлення і запобігання плагіату щодо роботи:

Автор: Анастасія ЧАЙКОВСЬКА

Співавтор:

Назва: Чайковська_Програмно-технічні засоби підключення світлових приладів на основі одноплатної комп'ютерної системи Raspberry Pi4

Експерт:

Підрозділ: Кафедра комп'ютерної інженерії та інформаційних систем

Коефіцієнт подібності 1:13%

Коефіцієнт подібності 2:8%

Мікропробіли: 6

Заміна букв: 9

Інтервали: 0

Білі знаки: 0

Дата створення звіту: 2025-05-31 07:58:44.0

Після аналізу Звіту подібності констатую наступне:

☒ **Запозичення, виявлені в роботі є законними і не є плагіатом. Рівень подібності не перевищує допустимої межі. Таким чином робота незалежна і приймається.**

☐ **Запозичення не є плагіатом, але перевищено граничне значення рівня подібностей. Таким чином робота повертається на доопрацювання.**

☐ **Виявлено запозичення і плагіат або навмисні текстові спотворення (маніпуляції), як передбачувані спроби укриття плагіату, які роблять роботу невідповідною вимогам законодавства (Ст. 32. ЗУ Про вищу освіту, пункт 3.1, Ст. 42. ЗУ Про освіту) та вимог НАЗЯВО (Критерій 5), а також кодексу етики і процедурам. Таким чином робота не приймається.**

Обгрунтування:

2025-05-31 Доцент Андрій Нічепорук

Дата експерт

Anti-Plagiarism (UA) v-15.281 Educational

The maximum coincidence with one document 24.0%

Dictionaries check: en_US, ru_RU, ua_UA. **Errors in the documents: 12%**

	Document		Sum coincidence on the DB	
ID: 242693 Title: БКР Програмно-технічні засоби підключення світлових приладів на основі одноплатної комп'ютерної системи Raspberry Pi4 Added in a DB: 2025-05-31 Authors: Анастасія ЧАЙКОВСЬКА Heads: Сергій ЛИСЕНКО Consultants: Opponents:	Symbols	Lexemes	Symbols	Lexemes
	82736	668	21004 (25%)	174 (26%)

Plagiarism sources

ID	Description	Plagiarism presence in the document	
		Symbols	Lexemes
240781	Title: Звіт з ПДП Програмно-технічний засоби підключення світлових приладів на основі одноплатної комп'ютерної системи Raspberry Pi4 Added in a DB: 2025-05-02 Authors: Чайковської А.В. Heads: Лисенко С.М. Consultants: Opponents:	19583 (24.0%)	156 (23.0%)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

РЕЦЕНЗІЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

Дипломник: ЧАЙКОВСЬКА АНАСТАСІЯ ВОЛОДИМИРІВНА

Тема: Програмно-технічні засоби підключення світлових приладів на основі одноплатної комп'ютерної системи Raspberry Pi4

Спеціальність: 123 «Комп'ютерна інженерія»

Обсяг кваліфікаційної роботи:

Кількість листів креслень ___3____ Кількість сторінок записки ___59___

1. Короткий зміст роботи та прийнятих рішень: Метою кваліфікаційної роботи є проектування засобів підключення світлових приладів на основі одноплатної комп'ютерної системи Raspberry Pi4.

2. Висновок про відповідність роботи дипломному завданню: Робота повністю відповідає поставленому завданню.

3. Характеристика виконання кожного розділу, ступінь використання останніх досягнень науки і техніки і передових методів роботи: В першому розділі проведено аналіз сучасних підходів до цифрового керування освітленням, зосереджений на застосуванні PC-DMX інтерфейсу. Було обґрунтовано доцільність використання протоколу DMX512 як основи для реалізації гнучкого й масштабованого керування світловими пристроями. В другому розділі проведено проведено системне проектування кіберфізичної системи адаптивного застосування моніторингових елементів. Визначено її апаратну та програмну архітектуру, описано компоненти системи та їхню взаємодію. Зокрема, проаналізовано реалізацію передачі даних через UART, USB та SPI, застосування Open Lighting Architecture (OLA) як системного ПЗ, а також адаптацію плагінів для підтримки специфічних каналів вводу/виводу. В третьому розділі реалізовано програмно-апаратну частину проєкту. Детально описано процес налаштування Raspberry Pi4, встановлення й конфігурації OLA, реалізацію електромонтажу та інтеграцію з USB- і SPI-пристроями. Розроблено механізм обробки вхідного і вихідного DMX-сигналу, зокрема,

використано бітове зчитування та реалізацію машини станів для декодування SPI-даних.

4. Позитивні сторони роботи: у роботі було розроблено і реалізовано програмно-технічні засоби реалізації PC-DMX інтерфейсу на основі одноплатної комп'ютерної системи raspberry PI.

5. Негативні сторони роботи: не в повній мірі розписано етапи проектування плати розширення.

6. Оцінка графічного оформлення та пояснювальної записки роботи: Пояснювальна записка оформлена коректно, згідно діючих стандартів оформлення документації.

7. Відгук про роботу в цілому: Робота виконана на належному науково-технічному рівні.

8. Інші зауваження: —

9. Оцінка дипломної роботи: Задовільно

Рецензент (прізвище, ім'я, по батькові, посада, місце роботи) _____

Карпуха Юрій Вікторович

доцент кафедри ПЗ

"2" червня 2025 р. _____(підпис)

Завідувачу кафедри КІІС
д-р. філософії, доц. Ользі ПАВЛОВІЙ

Анастасії ЧАЙКОВСЬКОЇ
ПІБ здобувача вищої освіти

ФІТ, 3 курсу, групи КІ2с-22-2

ЗАЯВА

З правилами чинного Положення «Про систему забезпечення академічної доброчесності у Хмельницькому національному університеті» від 01.07.2022, згідно з яким виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту та застосування заходів дисциплінарної та академічної відповідальності, ознайомлений (а). Про використання програмно-технічних засобів для перевірки кваліфікаційних робіт здобувачів вищої освіти на плагіат оповіщений(а) та надаю свою згоду на обробку та збереження університетом моєї роботи в інституційному репозитарії університету.

Також надаю університету право на передачу моєї роботи для обробки та збереження в базах даних програмно-технічних засобів (Strike-Plagiarism та Anti-Plagiarism) та використання роботи для виявлення плагіату в інших роботах, які перевіряються програмно-технічними засобами та користувачами, що мають доступ до цих програмно-технічних засобів, виключно в обмежених цілях для виявлення плагіату в текстах робіт.

Робота для перевірки університетом надається в друкованому та електронному варіанті. Електронна версія моєї роботи збігається (ідентична) з друкованою.

02 червня 2025 року

РІШЕННЯ ЕКСПЕРТНОЇ КОМІСІЇ
КАФЕДРИ КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ ТА ІНФОРМАЦІЙНИХ СИСТЕМ
ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Підтверджуємо ознайомлення з результатом звіту подібності щодо роботи, генерованого системою виявлення текстових збігів/ідентичності/схожості:

Назва: Програмно-технічні засоби підключення світлових приладів на основі одноплатної комп'ютерної системи Raspberry Pi4

Автор: Анастасія ЧАЙКОВСЬКА

Спеціальність: 123– Комп'ютерна інженерія

Освітня програма: освітньо-професійна

Науковий керівник: Сергій ЛИСЕНКО, д.т.н, професор

Після аналізу звіту подібності зроблено такий висновок:

№	Висновок	Позначка про відповідність
1	Запозичення, виявлені в роботі, є законними і не є плагіатом. Робота приймається до захисту.	відповідає
2	Виявлені запозичення не є плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи Робота приймається до захисту, але має бути відкоригована. Відкоригований варіант має бути поданий на кафедру за 2 дні до захисту, разом із заявою щодо самостійності виконання письмової роботи та ідентичності друкованої та електронної версії роботи	
3	Виявлені запозичення не є плагіатом, але частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. В зв'язку з цим мета роботи та поставлені завдання не були досягнені. Робота може бути допущена до захисту (наступного року) після того як буде відкоригована та допрацьована і успішно пройде повторну перевірку на академічний плагіат.	
4	Робота містить навмисні текстові спотворення, передбачувані спроби укриття запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	

Підтвердження:

Запозичення, виявлені в роботі, є законними і не є плагіатом, оскільки:

1) запозичення розміщені в розділах аналізу існуючих аналогів та прототипів, які не описують безпосередньо авторське дослідження і не стосуються результатів роботи;
2) усі запозичення фрагментарні, або мають належним чином оформленні посилання;
3) окремі виявлені збіги є загальновживаними фразами або виразами, про що свідчить посилання системи на збіг з 10 джерелами на один фрагмент речення;
4) всі зафіксовані системою ознаки модифікації тексту відносяться до комбінування латинських символів зі україномовними скороченнями індексів в формулах, що не є модифікацією тексту.

Сумарний обсяг всіх запозичень, визначений системою виявлення збігів/ ідентичності/схожості StrikePlagiarism, складає 13% і адресується до 38 першоджерел; та системою Anti-Plagiarism складає 24.0%, що, з урахуванням наведених обґрунтувань, відповідає характеру наукового дослідження і свідчить на користь кваліфікаційної роботи.

Керівник роботи _____ Сергій ЛИСЕНКО

Гарант ОП _____ Андрій НІЧЕПОРУК

Завідувач кафедри КІС _____ Ольга ПАВЛОВА