

Центральноукраїнський національний технічний університет
Механіко-технологічний факультет
Кафедра кібербезпеки та програмного забезпечення

”Допущено до захисту”
Завідувач кафедри кібербезпеки
та програмного забезпечення
д.т.н., професор
_____ Олексій СМІРНОВ
“___” _____ 2026 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за першим (бакалаврським) рівнем вищої освіти
на тему

***“Програмне забезпечення системи відмовостійкого алгоритму
передачі інформаційних пакетів даних з моделюванням фізичних
завад”***

Виконав здобувач вищої освіти
IV курсу, групи КН-22
ОПП «Комп’ютерні науки»
спеціальності 122 «Комп’ютерні науки»
_____ Бездольний К.О.
«___» _____ 2026 р.

Керівник проекту
кандидат технічних наук, доцент
_____ Дрєєв О.М.
«___» _____ 2026 р.
Рецензент _____

Центральноукраїнський національний технічний університет
Факультет Механіко-технологічний
Кафедра Кібербезпеки та програмного забезпечення
Освітній ступінь бакалавр
Галузь знань ._12 _____“Інформаційні технології”
Спеціальність 122 “Комп’ютерні науки”
Освітньо-професійна (освітньо-наукова) програма “Комп’ютерні науки”

ЗАТВЕРДЖУЮ

Завідувач кафедри

д.т.н., проф.

_____ Олексій СМІРНОВ

« 06 » лютого 2026 року

ЗАВДАННЯ НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ ЗА ПЕРШИМ (БАКАЛАВРСЬКИМ) РІВНЕМ ВИЩОЇ ОСВІТИ ЗДОБУВАЧА ВИЩОЇ ОСВІТИ

Бездольному Кирилу Олександровичу

(прізвище, ім’я, по батькові)

1. Тема роботи *Програмне забезпечення системи відмовостійкого алгоритму передачі інформаційних пакетів даних з моделюванням фізичних завод*

2. Керівник роботи *Дресєв Олександр Миколайович, канд. техн. наук, доцент*

(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 116-02 від 06.02.2026 року

3. Строк подання студентом роботи до захисту 25.05.2026 р.

4. Мета та завдання випускної кваліфікаційної роботи: *Метою роботи є розробка програмного забезпечення системи відмовостійкого алгоритму передачі інформаційних пакетів даних з моделюванням фізичних завод*

5. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Призначення та область використання.

2.Перегляд аналогічних існуючих систем.

3. Опис і обґрунтування проектних рішень.

4. Етапи програмування системи.

5. Впровадження системи в промислову експлуатацію.

6. Висновки

6. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)

Структурна схема системи 1 аркуш

Функціональна схема системи 1 аркуш

Діаграма процесів 1 аркуш

Блок-схема алгоритму роботи додатку 2 аркуша

7. Дата видачі завдання « 06 » лютого 2026 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Строк виконання етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Примітка
1.	Аналіз існуючих систем	10.03.2026 р.	
2.	Постановка задачі, оформлення ТЗ	15.03.2026 р.	
3.	Розробка моделі компонента	20.03.2026 р.	
4.	Розробка структур даних	25.03.2026 р.	
5.	Розробка алгоритмів зв'язку та відображення	30.03.2026 р.	
6.	Програмування алгоритмів	10.04.2026 р.	
7.	Оформлення ПЗ	17.04.2026 р.	
8.	Попередній захист роботи	25.05.2026 р.	

Дата видачі завдання
« 06 » лютого 2026 р.

Підпис керівника

Дреєв О.М.
(прізвище та ініціали)

Завдання прийнято до виконання
« 06 » лютого 2026 р.

Підпис здобувача

Бездольний К.О.
(прізвище та ініціали)

АНОТАЦІЯ

Бездольний К.О. Програмне забезпечення системи відмовостійкого алгоритму передачі інформаційних пакетів даних з моделюванням фізичних завад. 122 Комп'ютерні науки. Центральнотукаїнський національний технічний університет. Кропивницький. 2026.

В даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи відмовостійкого алгоритму передачі інформаційних пакетів даних з моделюванням фізичних завад.

Метою розробки є програмне забезпечення системи відмовостійкого алгоритму передачі інформаційних пакетів даних з моделюванням фізичних завад.

Результат роботи – програмна реалізація системи відмовостійкого алгоритму передачі інформаційних пакетів даних з моделюванням фізичних завад.

В процесі роботи над програмною моделлю виконано аналіз існуючих апаратних та програмних засобів. В повній мірі описані всі компоненти розробленого програмного забезпечення.

Розроблено зручний інтерфейс користувача. Наведені інструкції по роботі з програмними засобами.

Програма системи може використовуватися на ПЕОМ з ОС Windows 10/11.

Програму системи розроблено за використанням Python та C++.

Ключові слова: комп'ютерні науки, лазерні мережі, передача інформації, відмовостійка передача, лазерний зв'язок, відмовостійкий алгоритм

ABSTRACT

Bezdolnyi K.O. Software for a fault-tolerant algorithm for transmitting data packets with physical interference modeling. 122 Computer Science. Central Ukrainian National Technical University. Kropyvnytskyi. 2026.

In this final qualification work for the first (bachelor's) level of higher education, software was developed for a system implementing a fault-tolerant algorithm for transmitting information data packets with modeling of physical interferences.

The purpose of the development is to create software for a system implementing a fault-tolerant algorithm for transmitting information data packets with modeling of physical interferences.

The result of the work is a software implementation of a system with a fault-tolerant algorithm for transmitting information data packets with modeling of physical interferences.

During the development of the software model, an analysis of existing hardware and software tools was carried out. All components of the developed software are described in detail.

A user-friendly interface was developed. Instructions for working with the software tools are provided.

The system software can be used on personal computers running Windows 10/11 operating systems.

The system software was developed using Python and C++.

Keywords: computer science, laser networks, information transmission, fault-tolerant transmission, laser communication, fault-tolerant algorithm

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ	2
ВСТУП.....	3
1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ	6
1.1 Призначення системи.....	6
1.2 Область застосування.....	7
2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ	8
2.1 Огляд існуючих систем, технологій, архітектур, програмних рішень за профілем теми випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти.....	8
2.2 Обґрунтування вибору засобів для побудови системи та мови програмування	13
3 ОПИС І ОБГРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ	24
3.1 Опис функціонування системи	24
3.2 Розробка структурної схеми.....	27
3.3 Розробка функціональної схеми	29
3.4 Розробка діаграми взаємодії процесів.....	35
4.1 Блок-схеми та опис алгоритмів функціонування системи.....	38
4.2 Захист розробленого програмного забезпечення.....	57
5 ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ	59
6 ОСНОВНІ ВИСНОВКИ.....	62
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	64

					ВКРБ-122.26.0003.00.00.ПЗ			
Вим.	Арк.	№ докум.	Підп.	Дата				
Розроб.	Бездольний К.О.				<i>Програмне забезпечення системи відмовостійкого алгоритму передачі інформаційних пакетів даних з моделюванням фізичних завад</i>	Літ.	Аркуш	Аркушів
Перев.	Дресв О.М.					Б	1	69
Н.контр.	Коваленко А.С.					ЦНТУ КН-22		
Затв.	Смірнов О.А.							

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

АЦП	–	Аналого-цифровий перетворювач
бод	–	Одиниця швидкості передачі за стандартом (символів за секунду в послідовному каналі)
ШИМ	–	Широтно-імпульсна модуляція
ACK	–	Acknowledgement – підтвердження успішного прийому кадру
COM	–	Назва логічного послідовного порту
CRC	–	Cyclic Redundancy Check – циклічний надлишковий контроль, контрольна сума
LEDC	–	Підсистема керування ШИМ у середовищі Arduino для ESP32
PyQt5	–	Бібліотека прив'язок мови Python до фреймворку Qt5 для побудови ГІ
FreeRTOS	–	Real-Time Operating System – операційна система реального часу
UART	–	Universal Asynchronous Receiver-Transmitter – універсальний асинхронний приймач-передавач
Кадр	–	Структурований пакет оптичного рівня: тип, номер, довжина, корисне навантаження, контрольна сума
Оптичний канал	–	Фізичний шлях передачі між випромінювачем і фотоприймачем
Тайм-аут	–	Максимально допустимий час очікування події (біта, квитанції тощо)

ВСТУП

У сучасних локальних мережах бездротова передача даних здебільшого базується на радіоканалах, які, попри свою універсальність, мають ряд обмежень. До них належать зашумленість ефіру, обмежена пропускна здатність у перевантажених середовищах, а також потенційні проблеми з безпекою через широке розповсюдження сигналу. У зв'язку з цим виникає потреба в альтернативних підходах до організації передачі даних на невеликі відстані, які могли б забезпечити більшу стабільність, швидкість або контрольованість каналу.

Одним із таких підходів є використання оптичного випромінювання, зокрема лазерів. Подібні технології вже широко застосовуються в різних сферах: від оптоволоконного інтернету, де світловий сигнал передається через кабель, до космічних систем зв'язку, де лазер використовується для високошвидкісної передачі даних між об'єктами. Навіть у менш критичних застосуваннях, таких як 2D-лідари, лазерні системи демонструють ефективність у вимірюванні та передачі інформації.

У межах цього проєкту розглядається система LaserLink, яка демонструє альтернативний канал зв'язку на основі лазерного випромінювання в локальних умовах. Система складається з двох основних частин: мікроконтролерного пристрою на базі ESP32-C3, що включає лазерний випромінювач видимого спектру (зелений колір для спостерігання за процесом передачі) для передачі даних і фотосенсор для прийому сигналу, а також настільного задія стосунку, розробленого на Python, який забезпечує керування процесом передачі даних і аналіз роботи системи.

Актуальність теми. Актуальність теми зумовлена зростаючою потребою у простих, доступних і наочних способах передавання інформації між пристроями без використання радіоканалу. У багатьох випадках застосування радіочастот є або надлишковим, або небажаним через завади, енергоспоживання

					ВКРБ-122.26.0003.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		3

чи вимоги до локалізації сигналу. Натомість оптичний канал зв'язку дозволяє забезпечити високу спрямованість передавання, що робить його зручним для експериментальних установок, навчальних стендів та досліджень альтернативних методів комунікації.

Крім того, система надає можливість практично досліджувати вплив різних параметрів на якість передачі, зокрема потужності лазера, чутливості фотосенсора, тривалості імпульсів і таймаутів. Наявність графічного інтерфейсу користувача забезпечує наочне спостереження за процесом передачі в реальному часі, що суттєво підвищує цінність проєкту як інструменту для навчання та експериментів.

Мета й завдання дослідження. Метою даної роботи є розробка апаратно-програмної системи для передавання текстових даних між двома мікроконтролерними пристроями через лазерний оптичний канал із можливістю гнучкого керування параметрами сигналу, контролю помилок та візуалізації результатів передачі. Така система має не лише забезпечувати базову функціональність обміну даними, але й надавати інструменти для аналізу якості зв'язку та впливу різних факторів на його надійність.

Важливим етапом є розробка схеми пристрою на базі ESP32-C3 із використанням лазерного випромінювача видимого спектру та фотосенсора, а також реалізація методу кодування бітів через часові інтервали сигналу (HIGH/LOW). На рівні обміну даними розробляється кадровий протокол, який включає службові поля (тип, номер послідовності, довжина), корисне навантаження та контрольну суму CRC для виявлення помилок.

Завершальним етапом є експериментальна перевірка працездатності системи на різних швидкостях і параметрах сигналу, що дозволяє оцінити ефективність запропонованих рішень і визначити межі стабільної роботи каналу.

Практична цінність отриманих результатів. Практична цінність роботи полягає у створенні повноцінної демонстраційної системи оптичного зв'язку, яка може бути використана як навчальний інструмент для вивчення принципів

					ВКРБ-122.26.0003.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		4

цифрової передачі даних, побудови протоколів обміну та аналізу впливу фізичних параметрів сигналу на якість зв'язку. Розроблена система є доступною для повторення завдяки використанню недорогих компонентів і може бути застосована в лабораторних роботах, дослідницьких проєктах або як база для подальших удосконалень у сфері альтернативних каналів передачі даних.

К6ПЗ_2026

					ВКРБ-122.26.0003.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		5

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Система LaserLink призначена для передавання текстових повідомлень між двома однаковими мікроконтролерними пристроями на базі ESP32 через спрямований лазерний оптичний канал. Вона реалізує повний цикл обміну даними – від введення повідомлення користувачем до його прийому, перевірки та відображення результату.

Принцип роботи системи побудований таким чином: користувач вводить текст у настільному застосунку, після чого програма передає ці дані через інтерфейс USB Serial на перший пристрій. Мікроконтролер виконує кодування інформації у послідовність світлових імпульсів (чергування станів HIGH/LOW) і передає їх через лазерний випромінювач. Другий пристрій, оснащений фотосенсором, приймає оптичний сигнал, перетворює його назад у цифрову форму, виконує перевірку цілісності кадру та передає отриманий текст у застосунок для відображення користувачу.

Функціональні можливості системи охоплюють як базову передачу даних, так і інструменти для налаштування та аналізу роботи каналу. Зокрема, передбачено підключення двох пристроїв через COM-порти з можливістю незалежного керування кожним із них. Користувач може надсилати текстові дані та отримувати результати прийому в реальному часі.

Система дозволяє гнучко налаштовувати параметри передачі, включаючи швидкість обміну, тривалість світлових імпульсів, поріг спрацювання фотосенсора, потужність лазера, розмір кадру та кількість повторів у разі помилок. Для керування процесом передбачено можливість примусової зупинки активної передачі за допомогою команди @STOP.

					ВКРБ-122.26.0003.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		6

Окрему роль відіграють засоби діагностики: застосунок веде лог подій та відображає графіки, що дозволяє аналізувати поведінку системи, виявляти помилки та оцінювати вплив параметрів на якість зв'язку.

1.2 Область застосування

Розроблена система LaserLink орієнтована передусім на навчальні, лабораторні та експериментальні задачі, де важлива не лише передача даних, а й розуміння процесів, що відбуваються на різних рівнях зв'язку. Завдяки своїй наочності та простоті реалізації вона дозволяє демонструвати принципи роботи фізичного рівня передачі даних, включаючи формування сигналу, його поширення та прийом.

Система може ефективно використовуватись для вивчення побудови простих протоколів обміну, аналізу помилок та механізмів їх обробки. Окрему цінність становить можливість дослідження впливу зовнішніх і внутрішніх факторів на якість зв'язку, зокрема шуму, відстані між пристроями, умов освітлення, а також параметрів фотосенсора та лазерного випромінювача.

У практичному плані LaserLink може бути застосована як основа для навчальних стендів з дисциплін, пов'язаних із комп'ютерними мережами, мікроконтролерами та вбудованими системами. Вона також підходить для проведення лабораторних робіт із цифрового кодування сигналів і передачі даних. Крім того, систему можна використовувати для прототипування короткодістанційного спрямованого зв'язку, де важлива локалізація сигналу.

Ще одним напрямком застосування є дослідження надійності передачі даних, зокрема ефективності використання механізмів підтвердження прийому (ACK), контролю цілісності (CRC) та повторної передачі. Це дозволяє оцінити поведінку системи в умовах перешкод і сформулювати уявлення про принципи побудови стійких каналів зв'язку.

					ВКРБ-122.26.0003.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		7

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

2.1 Огляд існуючих систем, технологій, архітектур, програмних рішень за профілем теми випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти

На сьогоднішній день не існує універсального готового програмного продукту, який би повністю вирішував задачу відмовостійкої передачі інформаційних пакетів у лазерних мережах зв'язку з моделюванням фізичних завад. Проте, для дослідження окремих аспектів цієї проблематики активно використовуються спеціалізовані симулятори та наукові розробки. Система LaserLink належить до класу оптичних каналів зв'язку, однак за своєю реалізацією та цілями суттєво відрізняється від більшості існуючих рішень. Для кращого розуміння її місця доцільно розглянути кілька поширених підходів до передачі даних.

1) Інфрачервоний зв'язок, зокрема стандарти IrDA, є одним із найближчих аналогів. У таких системах передача даних здійснюється через інфрачервоне випромінювання за принципом прямої видимості, що забезпечує відносно високу швидкість і низький рівень помилок. Водночас IrDA включає повноцінний стек протоколів – від фізичного до прикладного рівня – і є стандартизованим рішенням, орієнтованим на сумісність між пристроями. На Рисунку 2.1 зображено приклад одних з таких модулів. У порівнянні з ним LaserLink не використовує готові стандарти, а навпаки – реалізує власний спрощений протокол, що робить його більш придатним для навчальних цілей і дослідження принципів передачі даних.

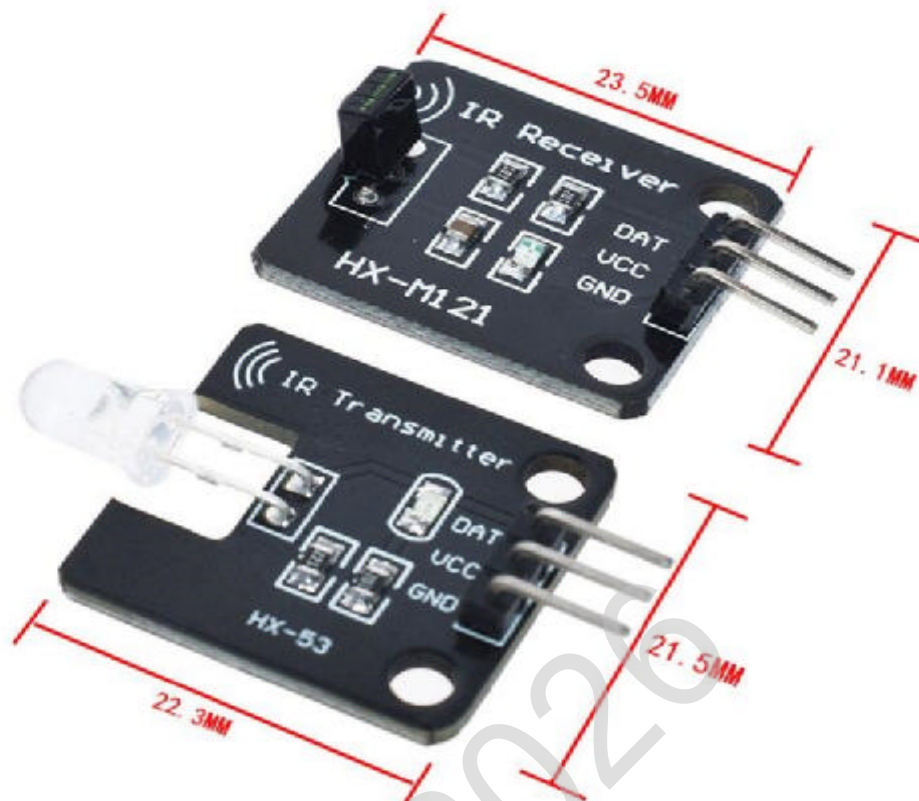


Рисунок 2.1 – Модулі інфрачервоного зв'язку

2) Технології типу Li-Fi та VLC (Visible Light Communication) представляють більш сучасний напрям оптичного зв'язку. Вони також базуються на модуляції світлового сигналу, але використовують складніші апаратні рішення, такі як світлодіоди високої частоти та фотоприймачі, і здатні досягати дуже високих швидкостей – від сотень мегабіт до десятків гігабіт за секунду. Крім того, такі системи часто мають стандартизовані протоколи та підтримують різні топології мереж. У цьому контексті LaserLink є значно простішим і повільнішим, проте його перевага полягає у прозорості реалізації та можливості безпосередньо спостерігати процес передачі.

3) Радіоканальні технології, такі як Bluetooth і Wi-Fi, є найбільш поширеними засобами бездротового зв'язку. Вони забезпечують зручність використання, високу швидкість і відсутність вимог до прямої видимості, на

Рисунку 2.2 зображено сучасний Wi-Fi роутер. Однак їхня складна внутрішня реалізація приховує деталі фізичного рівня, що робить їх менш придатними для демонстрації базових принципів передачі сигналу. На відміну від них, LaserLink дозволяє явно спостерігати формування та прийом сигналу, що є важливим для навчальних і експериментальних задач.



Рисунок 2.2 – Wi-Fi роутер

4) Провідні інтерфейси, такі як UART/Serial між мікроконтролерами, забезпечують простий і надійний обмін даними, але не включають задач, пов'язаних із передачею сигналу через середовище. У таких системах відсутні проблеми оптичного детектування, впливу зовнішніх завад чи необхідності калібрування параметрів сигналу. LaserLink, навпаки, додає ці фактори, що дозволяє досліджувати реальні умови передачі даних.

5) Окрему категорію становлять готові оптичні модулі зв'язку, які інтегрують передавач, приймач і протокол обміну в одному пристрої. Вони забезпечують високу ефективність і простоту використання, але практично не залишають можливостей для втручання в низькорівневу логіку передачі. У цьому сенсі LaserLink має перевагу як дослідницька платформа, оскільки дозволяє самостійно реалізувати всі рівні обміну – від фізичного сигналу до протоколу передачі даних.

Якщо розглядати існуючі рішення, то можна виділити кілька найбільш поширених та ефективних інструментів для моделювання та аналізу оптичних і лазерних мереж зв'язку. Ці інструменти забезпечують гнучкість у налаштуванні параметрів мережі, дозволяють враховувати фізичні завади та відмови елементів, а також підтримують розробку і тестування власних алгоритмів маршрутизації та відновлення працездатності мережі. Нижче приведено список декількох з них:

- Одним із найпоширеніших інструментів для моделювання та аналізу оптичних мереж є Optical Network Simulator (ONS). Цей симулятор дозволяє дослідникам розробляти та тестувати алгоритми маршрутизації, оцінювати відмовостійкість мережі, а також моделювати різні сценарії передачі даних у WDM та еластичних оптичних мережах. ONS підтримує розширення функціоналу, що дає змогу реалізовувати власні модулі для моделювання специфічних фізичних завад, таких як атмосферні явища, вібрації чи перешкоди на шляху лазерного променя.

- Ще одним потужним інструментом є OMNeT++ (див. Рисунок 2.3) – універсальна платформа для симуляції комп'ютерних мереж, яка має модулі для моделювання оптичних мереж, зокрема еластичних оптичних мереж (наприклад, eonsim). OMNeT++ дозволяє створювати складні сценарії передачі даних, аналізувати вплив відмов окремих елементів мережі та досліджувати ефективність різних алгоритмів маршрутизації в умовах фізичних завад.

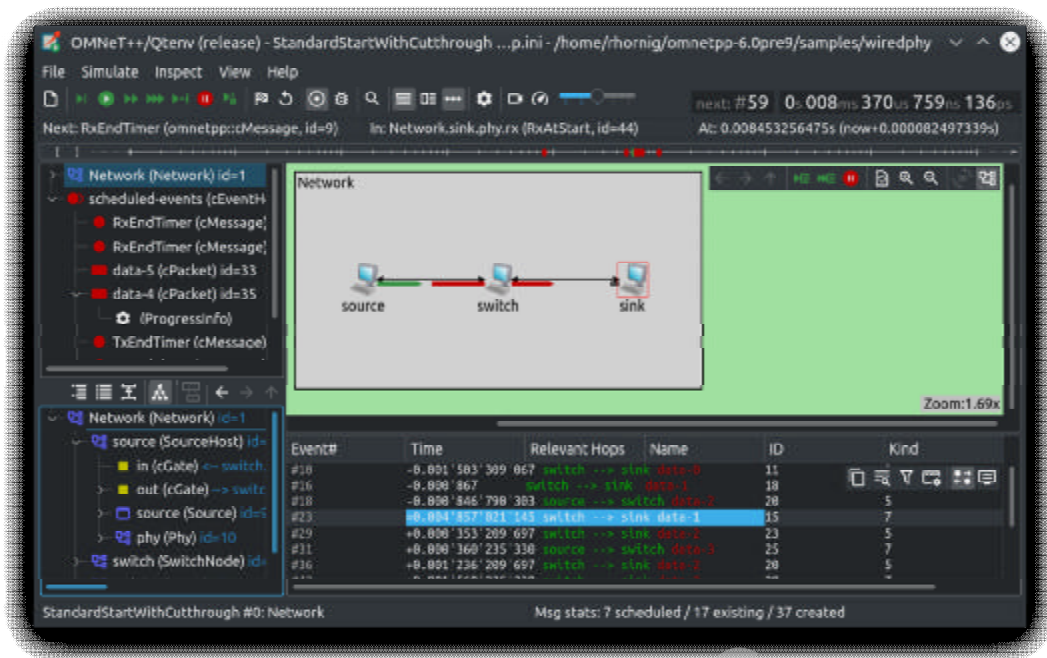


Рисунок 2.3 – Інтерфейс OMNeT++ системи

- Окремо варто згадати симулятор пасивних оптичних мереж (PON) із фактором поділу (див. Рисунок 2.4). Цей симулятор дозволяє моделювати структуру та роботу пасивних оптичних мереж із різним ступенем поділу сигналу, що є актуальним для аналізу відмовостійкості та ефективності передачі даних у багатокористувацьких середовищах.



Рисунок 2.4 – Фізичний симулятор пасивних оптичних мереж (PON)

Таким чином, LaserLink займає проміжне положення між простими навчальними моделями та складними промисловими рішеннями, поєднуючи доступність реалізації з можливістю глибокого дослідження процесів цифрового оптичного зв'язку.

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

Вибір апаратної та програмної платформи для системи LaserLink визначався вимогами до доступності, гнучкості налаштування та можливості роботи з низькорівневими параметрами сигналу.

Вибір мікроконтролера

Для реалізації системи LaserLink необхідно було обрати мікроконтролерну платформу, яка одночасно забезпечує роботу з цифровими та аналоговими сигналами, підтримує точне керування часовими інтервалами та має зручні засоби взаємодії з настільним застосунком. З урахуванням цих вимог основою системи було обрано мікроконтролер ESP32-C3 (див. Рисунок 2.5).

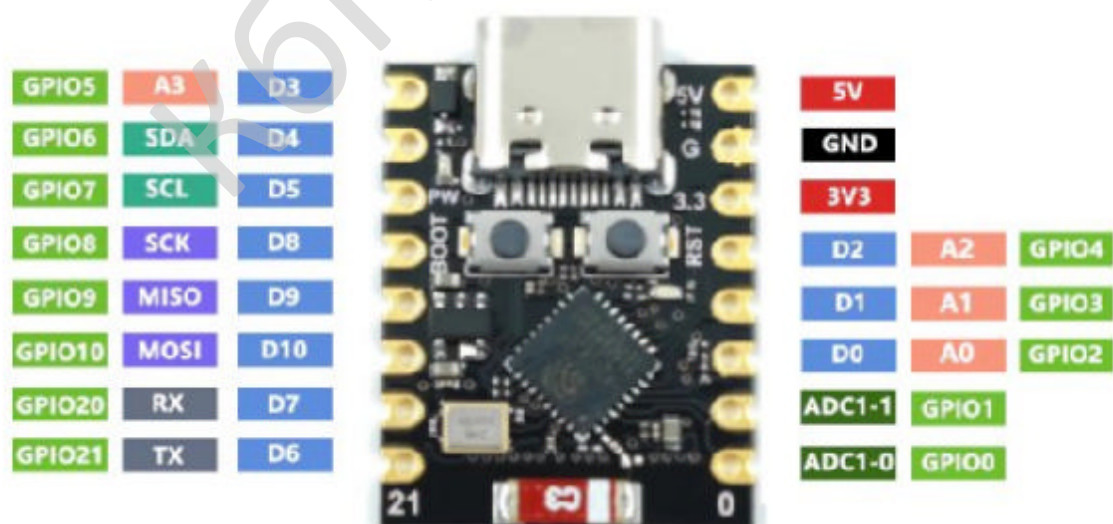


Рисунок 2.5 – Мікроконтролер ESP32-C3 supermini

Однією з ключових причин вибору є достатня обчислювальна продуктивність ESP32-C3 для реалізації логіки протоколу передачі даних. У межах проєкту мікроконтролер виконує розбиття повідомлень на кадри, обчислення CRC, обробку підтверджень (ACK), а також керує механізмами повторної передачі. Ці задачі потребують швидкої обробки даних і точного контролю послідовностей, що ESP32-C3 забезпечує без перевантаження ресурсів. У межах проєкту мікроконтролер виконує низку взаємопов'язаних задач:

- Розбиття повідомлень на кадри. Текстове повідомлення з боку оператора перетворюється на послідовність кадрів фіксованої або керованої структури; тут важливі буферизація, контроль довжини та коректна поведінка на межі «рваного» вводу з послідовного порту.

- Обчислення CRC. Контроль цілісності на рівні кадру знижує ймовірність тихих помилок, коли приймач приймає спотворені дані за валідні. Для навчального проєкту це також дисциплінує інженерний підхід: наявність формалізованого критерію коректності полегшує відтворення сценаріїв збоїв.

- Обробка підтверджень (ACK). Надійний режим передачі передбачає узгодження між вузлами: отримання квитанції, очікування, повтор у разі тайм-ауту. Це перетворює одноразову відправку імпульсів на протокол із станами, який повинен коректно працювати при завантаженні процесора діагностикою.

- Керування механізмами повторної передачі. Повтори повинні бути обмежені розумними межами, щоб не «забити» канал і не створити лавину повторів у поганих умовах; паралельно слід уникати взаємного блокування логіки прийому й передачі.

Ці задачі потребують швидкої обробки даних і точного контролю послідовностей, що ESP32-C3 забезпечує без перевантаження ресурсів у типовій конфігурації проєкту: обсяг оперативної пам'яті та пропускна здатність периферії дозволяють тримати буфери кадрів, таблиці для CRC і службові структури конфігурації, не вдаючись до надмірно складних оптимізацій на рівні асемблера.

Варто також відокремити апаратну підтримку від програмної політики реального часу. Навіть потужний процесор не гарантує стабільних таймінгів, якщо критичні ділянки змішані з блокуючими операціями високого рівня. Тому в обґрунтуванні вибору ESP32-C3 важлива не лише сира частота, а й можливість будувати прошивку так, щоб керування імпульсами та виміри АЦП відбувалися передбачувано (через таймерну логіку, апаратні модулі PWM, дисципліну в loop та обмеження важких операцій у критичних фазах).

Наявність універсальних GPIO-виводів дозволяє безпосередньо керувати лазерним модулем, формуючи цифрові імпульси для передачі даних. Водночас вбудований аналогово-цифровий перетворювач (ADC) використовується для зчитування сигналу з фотосенсора, що є критично важливим для приймальної частини системи. Це дозволяє реалізувати повний цикл обробки сигналу – від генерації до аналізу – на одному мікроконтролері.

Додатковою перевагою є наявність апаратного PWM (LEDC), який використовується для керування потужністю лазерного випромінювача. Це дає змогу гнучко змінювати інтенсивність сигналу без зміни апаратної частини, що особливо корисно під час експериментів із різними умовами передачі.

Інтерфейс USB/Serial значно спрощує інтеграцію з настільним застосунком: він використовується для передавання текстових даних, конфігураційних команд та діагностичної інформації. Це дозволяє реалізувати керування системою без додаткових інтерфейсних модулів.

ESP32-C3 також вирізняється доступністю, компактністю та широкою підтримкою інструментів розробки. У проєкті використовується середовище PlatformIO з платформною espressif32 та платою esp32-c3-devkitm-1, а також Arduino framework, що забезпечує швидкий старт, зручну конфігурацію та велику кількість готових бібліотек. Це особливо важливо для навчального та експериментального проєкту, де швидкість розробки і простота налаштування відіграють значну роль.

					ВКРБ-122.26.0003.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		15

У порівнянні з альтернативами, ESP32-C3 займає збалансовану позицію. Наприклад, Arduino Uno є простішим у використанні, однак має значно меншу продуктивність і обмежені можливості роботи з сучасними інструментами та бібліотеками. STM32, навпаки, пропонує високу продуктивність і гнучкість, але потребує більш складного налаштування і є менш зручним для швидкого прототипування. Одноплатні комп'ютери, такі як Raspberry Pi, є надлишковими для задачі керування лазером і фотосенсором у реальному часі, оскільки їх ресурси значно перевищують вимоги проєкту, а затримки операційної системи можуть ускладнювати точне керування таймінгами. Порівняння мікроконтролерів можна побачити у Таблиці 2.1:

Таблиця 2.1 – Порівняння оглянутих мікроконтролерів

Характеристика	ESP32-C3	Arduino Uno	STM32 Nucleo	Raspberry Pi 5
Тип пристрою	Мікроконтролер	Мікроконтролер	Мікроконтролер	Одноплатний комп'ютер
Архітектура	RISC-V	AVR	ARM Cortex-M	ARM Cortex-A
Тактова частота	до 160 МГц	16 МГц	72–480 МГц	до 2.4 ГГц
Бездротовий зв'язок	Wi-Fi, Bluetooth LE	Відсутній	Залежить від моделі	Wi-Fi, Bluetooth
Операційна система	Не потрібна	Не потрібна	Не потрібна	Linux
Енергоспоживання	Низьке	Низьке	Низьке / середнє	Високе
Орієнтовна вартість	~200–350 грн	~900–1300 грн	~800–2000 грн	~3500–6000 грн

Вибір фотосенсора

Приймальна частина системи LaserLink виконує критично важливу функцію – перетворення світлових імпульсів лазера на електричний сигнал, придатний для подальшої обробки мікроконтролером. Для реалізації цього вузла

обрано фотодіод BPW34 (див. Рисунок 2.6) у поєднанні з операційним підсилювачем MCP602 у корпусі DIP-8.

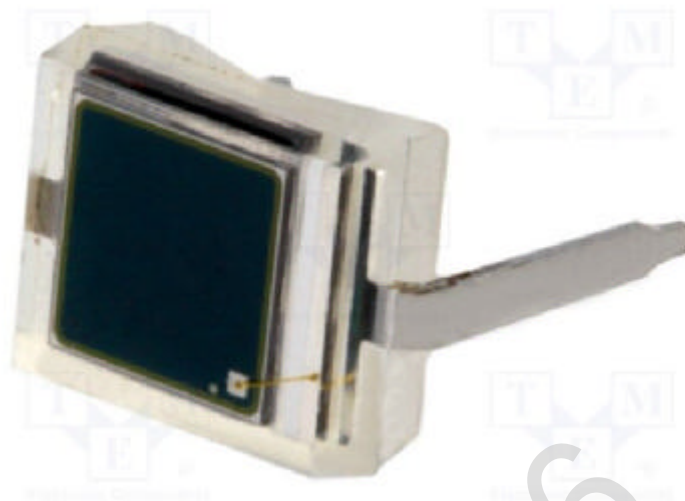


Рисунок 2.6 – Фотодіод BPW34

Фотодіод BPW34 є поширеним і добре дослідженим компонентом, який використовується для реєстрації світла у видимому та ближньому інфрачервоному діапазоні. Це робить його придатним для роботи з лазерними модулями, що застосовуються у проєкті. Однією з важливих характеристик BPW34 є достатня швидкодія, яка дозволяє фіксувати короткі світлові імпульси, що використовуються для кодування бітів. У порівнянні з альтернативами, такими як фоторезистори, фотодіод має значно швидшу реакцію на зміну освітлення, що є критично важливим для передачі цифрових сигналів. Крім того, BPW34 є компактным, доступним і широко використовується в експериментальних схемах оптичного зв'язку, що робить його зручним вибором для навчального прототипу.

Водночас фотодіод генерує дуже слабкий струм, рівень якого часто є близьким до шуму, особливо при невеликій інтенсивності світла або значній відстані між передавачем і приймачем. Тому безпосереднє підключення до входу мікроконтролера є недостатнім для стабільного зчитування сигналу. Для

					ВКРБ-122.26.0003.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		17

вирішення цієї проблеми використано операційний підсилювач MCP602 (див. Рисунок 2.7), який виконує функцію підсилення сигналу фотодіода.



Рисунок 2.7 – Операційний підсилювач MCP602 у корпусі DIP-8

MCP602 дозволяє перетворити слабкий фотострум у напругу з достатнім рівнем для коректного зчитування через ADC мікроконтролера. Це забезпечує більш стабільний і надійний прийом сигналу, зменшуючи вплив шумів і випадкових коливань. Додатковою перевагою є корпус DIP-8, який значно спрощує макетування, пайку та використання в лабораторних умовах, що відповідає характеру проєкту як експериментального стенду.

Використання підсилювального каскаду також підвищує стійкість системи до змін зовнішніх умов, таких як освітлення, відстань між пристроями або точність наведення лазера. Це дозволяє отримати більш передбачувану форму сигналу на вході ADC і спростити його подальшу цифрову обробку.

На програмному рівні обробка сигналу реалізується через використання порогового значення `sensorThreshold`. Мікроконтролер постійно зчитує значення з ADC і порівнює його з цим порогом, визначаючи наявність або відсутність світлового імпульсу. Такий підхід дозволяє перетворити аналоговий сигнал у цифрову послідовність бітів, придатну для подальшого декодування.

					ВКРБ-122.26.0003.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		18

Вибір мови та засобів програмування пристрою

Для реалізації прошивки мікроконтролерної частини системи LaserLink обрано мову програмування C++ у поєднанні з Arduino framework та середовищем розробки PlatformIO. Такий стек дозволяє ефективно поєднати низькорівневий контроль апаратури з відносною простотою розробки та підтримкою модульної архітектури.

Вибір мови C++ обумовлений тим, що вона є стандартом де-факто для розробки вбудованих систем. Вона забезпечує прямий доступ до апаратних ресурсів мікроконтролера, зокрема GPIO для керування лазером, ADC для зчитування сигналу з фотосенсора, PWM/LEDC для регулювання потужності випромінювання та Serial для обміну даними з ПК. Окрім цього, C++ дозволяє точно контролювати часові інтервали, що є критично важливим для коректного формування та декодування світлових імпульсів. Завдяки підтримці структурованого програмування та об'єктно-орієнтованих підходів, код може бути організований у вигляді окремих модулів і компонентів, що підвищує його читабельність і спрощує супровід. Продуктивність C++ є достатньою для виконання обчислень CRC, обробки протоколу, аналізу сигналу та реалізації механізмів повторної передачі в режимі реального часу.

Arduino framework обрано як програмну основу для взаємодії з апаратною частиною. Він значно спрощує роботу з мікроконтролером, надаючи зрозумілі та зручні функції, такі як `setup()`, `loop()`, `millis()`, `delay()` і `Serial`. Це дозволяє швидко реалізувати основну логіку пристрою без необхідності глибокого занурення в низькорівневу конфігурацію, характерну для середовища ESP-IDF. Таким чином, Arduino framework забезпечує баланс між простотою використання та доступом до необхідних апаратних можливостей.

Середовище PlatformIO використовується для організації процесу розробки, збірки та прошивання. Воно дозволяє чітко визначити конфігурацію проєкту, включаючи платформу (`espressif32`), конкретну плату (`esp32-c3-devkitm-1`), параметри компіляції та швидкість роботи Serial-монітора. PlatformIO також

					ВКРБ-122.26.0003.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		19

надає зручні інструменти для керування залежностями, структурування проєкту та інтеграції з редакторами коду. Це робить його особливо придатним для реалізації більш складних embedded-проєктів, ніж стандартне Arduino IDE.

Прошивка організована за модульним принципом, що дозволяє чітко розділити відповідальність між різними частинами системи. Основний файл `src/device/src/main.cpp` виконує координацію роботи пристрою та ініціалізацію всіх компонентів. Модулі `laser.h` і `sensor.h` відповідають за взаємодію з апаратними елементами — лазерним випромінювачем і фотосенсором відповідно. Файл `sender.h` реалізує логіку передачі сигналів, тоді як `receiver.h` відповідає за приймання та декодування вхідного потоку.

Окрему роль відіграє модуль `protocol.h`, у якому визначено структуру кадру та алгоритм обчислення CRC. На його основі працює `protocol_handler.h`, що реалізує механізми підтвердження прийому (ACK), повторної передачі кадрів і контроль успішності обміну. Така структура дозволяє ізолювати логіку протоколу від фізичного рівня передачі, що спрощує тестування та можливе розширення системи.

Вибір засобів для настільного застосунку системи

Настільна частина системи LaserLink виконує роль центрального інтерфейсу керування, через який користувач взаємодіє з пристроями, налаштовує параметри передачі, надсилає повідомлення та спостерігає за роботою системи в реальному часі. Для її реалізації обрано стек технологій: Python, PyQt5, pyserial, pyqtgraph і numpy, що забезпечує баланс між швидкістю розробки, функціональністю та зручністю використання.

Вибір мови програмування Python обумовлений її високою продуктивністю у створенні прикладного програмного забезпечення та широкою екосистемою бібліотек. Python дозволяє швидко реалізувати логіку взаємодії з пристроями, обробку отриманих даних і побудову інтерфейсу користувача. Це особливо важливо для лабораторних і експериментальних систем, де потрібна гнучкість і можливість швидкого внесення змін.

					ВКРБ-122.26.0003.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		20

Для створення графічного інтерфейсу використано бібліотеку PyQt5. Вона надає повноцінний набір інструментів для розробки desktop-застосунків, включаючи кнопки, вкладки, поля введення, списки COM-портів та області для відображення логів. Завдяки цьому забезпечується зручний і зрозумілий інтерфейс для користувача, який дозволяє легко керувати всіма аспектами системи.

Взаємодія з мікроконтролерами реалізується за допомогою бібліотеки pyserial, яка забезпечує роботу з COM-портами. Вона дозволяє надсилати текстові повідомлення, конфігураційні команди типу @CFG, а також службові команди, зокрема @STOP для зупинки передачі. Окрім цього, pyserial використовується для прийому логів і даних від ESP32 у реальному часі, що є основою для подальшої обробки та відображення.

Для візуалізації роботи системи використовується бібліотека pyqtgraph, яка оптимізована для швидкого відображення графіків у реальному часі. Вона дозволяє ефективно оновлювати графіки сигналу без значних затримок, що є важливим при аналізі оптичного каналу. У поєднанні з бібліотекою numpy, яка забезпечує зручну та швидку обробку числових масивів, це дозволяє реалізувати інструменти для аналізу значень сенсора, результатів передачі та часових характеристик сигналу.

Структура настільного застосунку організована таким чином, щоб розділити логіку інтерфейсу, обробки даних і взаємодії з пристроями. Файл src/app/main.py відповідає за запуск програми та ініціалізацію основних компонентів. Основне вікно реалізоване у src/app/ui/main_window.py, де зосереджено елементи керування, налаштування та логіка взаємодії з ESP32. Модуль src/app/services/serial_reader.py забезпечує обмін даними через Serial-інтерфейс, тоді як src/app/services/log_parser.py виконує розбір службових повідомлень, що надходять від пристроїв. За відображення графіків відповідає модуль src/app/ui/charts_tab.py, який реалізує вкладку діагностики.

Узагальнення вибору засобів

У результаті проведеного обґрунтування можна зробити висновок, що обрані апаратні та програмні засоби повністю відповідають задачі створення недорогого, доступного й наочного прототипу системи лазерної передачі даних. Вони забезпечують необхідний баланс між функціональністю, простотою реалізації та можливістю подальшого розширення.

Мікроконтролер ESP32-C3 виконує ключову роль у системі, забезпечуючи керування апаратною частиною, обробку протоколу передачі та точне дотримання часових параметрів сигналу. Приймальний оптичний вузол, реалізований на базі фотодіода BPW34 і операційного підсилювача MCP602, дозволяє стабільно перетворювати світлові імпульси у сигнал, придатний для аналізу.

Використання C++ у поєднанні з Arduino framework та PlatformIO забезпечує необхідний рівень контролю над апаратними ресурсами й часовими процесами, що є критично важливим для реалізації оптичного каналу зв'язку. Водночас настільний застосунок, створений на Python із використанням PyQt, забезпечує зручний інтерфейс користувача, керування параметрами системи та засоби для візуалізації й аналізу її роботи.

Таким чином, обраний стек технологій дозволяє ефективно реалізувати систему LaserLink як експериментальну платформу для дослідження принципів оптичної передачі даних.

2.3 Розгорнута постановка завдання

У межах даної роботи необхідно розробити повноцінну апаратно-програмну систему LaserLink, яка забезпечує передачу текстових даних між двома мікроконтролерними пристроями через оптичний (лазерний) канал зв'язку та дозволяє досліджувати параметри такого каналу в реальному часі.

					ВКРБ-122.26.0003.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		22

Система повинна приймати від користувача текстове повідомлення через настільний застосунок і передавати його на мікроконтролер ESP32-C3 за допомогою інтерфейсу Serial (USB). Після отримання даних пристрій-ініціатор повинен виконувати їх попередню обробку та розбиття на окремі кадри фіксованого або обмеженого розміру payload, щоб забезпечити коректну передачу навіть довгих повідомлень. Кожен кадр має бути сформований за визначеним структурним протоколом і містити службову інформацію, а також корисне навантаження. Далі дані повинні кодуватися на рівні окремих байтів у послідовність світлових імпульсів шляхом модуляції лазерного випромінювання (чергування логічних станів HIGH/LOW у часі). Таким чином реалізується фізичний рівень передачі даних через оптичний канал. На приймальній стороні система повинна зчитувати світлові імпульси за допомогою фотодіода та підсилювального каскаду, оцифровувати сигнал через ADC та відновлювати вихідні байти відповідно до заданого методу кодування. Після відновлення кадру необхідно виконати перевірку цілісності даних за допомогою CRC-контролю, що дозволяє виявляти помилки, спричинені шумами, нестабільністю сигналу або зовнішніми впливами. У випадку успішного прийому кадру приймальний пристрій повинен сформулювати та надіслати підтвердження (ACK) назад передавачу. Якщо підтвердження не було отримано у встановлений таймаут або виявлено помилку CRC, система повинна ініціювати повторну передачу відповідного кадру, що забезпечує базову надійність каналу зв'язку.

Окремою вимогою є можливість динамічного налаштування параметрів системи без необхідності перепрошивання пристрою. Користувач повинен мати змогу змінювати такі параметри, як швидкість передачі, тривалість імпульсів, поріг спрацювання фотосенсора, потужність лазера, розмір кадру та кількість повторних передач.

					ВКРБ-122.26.0003.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		23

3 ОПИС І ОБГРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

Робота системи LaserLink починається з ініціалізації настільної частини, яка виконує роль центрального вузла керування та моніторингу всього процесу передачі даних:

1) Користувач запускає Python-застосунок `src/app/main.py`. Після запуску ініціалізується головне вікно графічного інтерфейсу, створюються всі необхідні модулі для роботи з послідовними портами, логуванням та візуалізацією даних. На цьому етапі також завантажуються бібліотеки для роботи з COM-з'єднаннями (`pyserial`) та графіками реального часу (`pyqtgraph`). Застосунок переходить у стан очікування підключення пристроїв і готовності користувача до конфігурації системи.

2) У вкладці керування він підключає два ESP32-пристрої через COM-порти. Користувач вручну вибирає два доступні послідовні порти, що відповідають передавальному та приймальному ESP32-пристрою. Після встановлення з'єднання відкриваються двонаправлені Serial-канали зв'язку. На цьому етапі відбувається перевірка доступності пристроїв, ініціалізація обміну та підтвердження коректного підключення. Таким чином формується базова комунікаційна інфраструктура між ПК і мікроконтролерами.

3) Після встановлення з'єднання настільний застосунок формує конфігураційний пакет, який містить параметри роботи системи: швидкість передачі, тривалість імпульсів, поріг спрацювання фотосенсора, потужність лазера, розмір кадру та кількість повторних передач. Ці дані передаються на ESP32 у вигляді службової команди `@CFG`, після чого мікроконтролер парсить отриманий рядок і оновлює свої внутрішні параметри протоколу без необхідності перепрошивання.

					ВКРБ-122.26.0003.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		24

4) У графічному інтерфейсі користувач вводить текстове повідомлення, яке має бути передано через оптичний канал. Після натискання кнопки Send застосунок виконує попередню обробку даних: формує рядок для передачі, може додавати службову інформацію (за потреби) і передає його через Serial на передавальний ESP32. На цьому етапі запускається логіка ініціації передачі даних на стороні мікроконтролера.

5) Передавальний мікроконтролер приймає текстове повідомлення через UART-інтерфейс і передає його у модуль протоколу. Там дані розбиваються на кадри відповідно до заданого розміру payload. Для кожного кадру запускається функція `sendFrameReliable`, яка відповідає за надійну передачу: формування структури кадру, додавання службових полів (номер, довжина, CRC), кодування бітів у часові інтервали та ініціалізацію передачі через лазерний випромінювач. Одночасно активується механізм очікування підтвердження (ACK) від приймальної сторони, що є основою надійності всього каналу зв'язку.

6) Після отримання рядка та запуску механізму `sendFrameReliable` передавальний ESP32 виконує розбиття вхідного тексту на окремі кадри відповідно до заданого максимального розміру payload. Це необхідно для того, щоб забезпечити стабільну передачу навіть довгих повідомлень і спростити обробку на приймальній стороні. Для кожного сформованого кадру додатково обчислюється контрольна сума CRC, яка базується на всіх байтах службових полів і корисного навантаження. Це дозволяє приймальному пристрою перевіряти цілісність отриманих даних і виявляти помилки, що могли виникнути під час оптичної передачі через шум або втрати імпульсів.

7) Перед передачею кожного кадру система генерує спеціальну структуру сигналу, яка складається з трьох логічних частин: стартового ключового сигналу, основного потоку даних і кінцевого маркера. Стартовий сигнал використовується для синхронізації приймача та дозволяє йому визначити початок кадру, навіть якщо попередній стан каналу був невизначеним. Далі передаються біти даних, які кодуються через керування лазером у вигляді послідовності світлових імпульсів

					ВКРБ-122.26.0003.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		25

із заданою тривалістю HIGH/LOW. Після завершення передачі формується кінцевий сигнал, який дозволяє приймачу чітко визначити межу кадру та перейти до етапу обробки отриманих даних.

8) Приймальний ESP32 постійно моніторить вхідний сигнал із фотодіода через ADC і виконує аналіз рівня сигналу для виявлення стартового ключового патерну. Після його розпізнавання пристрій переходить у режим активного прийому і починає дискретизацію імпульсів, відновлюючи послідовність бітів. Отримані біти групуються у байти та формують повний кадр згідно з протоколом. Після завершення прийому виконується перевірка CRC, яка дозволяє визначити, чи не було спотворень даних під час передачі.

9) У випадку, коли перевірка CRC проходить успішно, приймальний ESP32 вважає кадр коректно отриманим. Він формує службове повідомлення у форматі [DATA]..., яке передається далі в настільний застосунок для відображення користувачу. Паралельно формується сигнал підтвердження АСК, який передається назад до передавача також через лазерний канал у вигляді короткого спеціального імпульсу або послідовності імпульсів, що однозначно ідентифікуються як підтвердження прийому.

10) Передавальний ESP32 після відправки кадру переходить у режим очікування підтвердження протягом заданого таймауту. Якщо АСК отримано вчасно, кадр вважається успішно переданим, і система переходить до обробки наступного кадру повідомлення. Якщо ж підтвердження не надходить або воно не розпізнане, ініціюється повторна передача поточного кадру відповідно до заданого ліміту повторів. Це забезпечує базову надійність каналу навіть при нестабільних умовах оптичного зв'язку.

11) Настільний застосунок постійно отримує службові повідомлення від обох ESP32 через Serial-інтерфейс і виконує їх парсинг. На основі отриманих даних оновлюється інтерфейс користувача: відображаються прийняті повідомлення, статуси передачі (АСК, помилки, повтори), а також оновлюються графіки діагностики сигналу. Це дозволяє користувачу в реальному часі

спостерігати за роботою системи, аналізувати якість передачі та оцінювати вплив налаштувань на стабільність каналу.

Таким чином, функціонування системи LaserLink побудоване як послідовний і замкнений цикл передачі даних між настільним застосунком та двома мікроконтролерами ESP32. Весь процес – від введення тексту користувачем до його відновлення на приймальній стороні – реалізується через багаторівневу взаємодію програмного забезпечення, протоколу обміну та фізичного оптичного каналу.

3.2 Розробка структурної схеми

На структурній схемі (див. Рисунок 3.1) показано загальну архітектуру системи LaserLink та взаємодію між її основними апаратними й програмними компонентами. Система складається з настільної частини (ПК із PyQt5-застосунком), двох однакових мікроконтролерних вузлів на базі ESP32-C3, а також оптичного каналу зв'язку між ними.

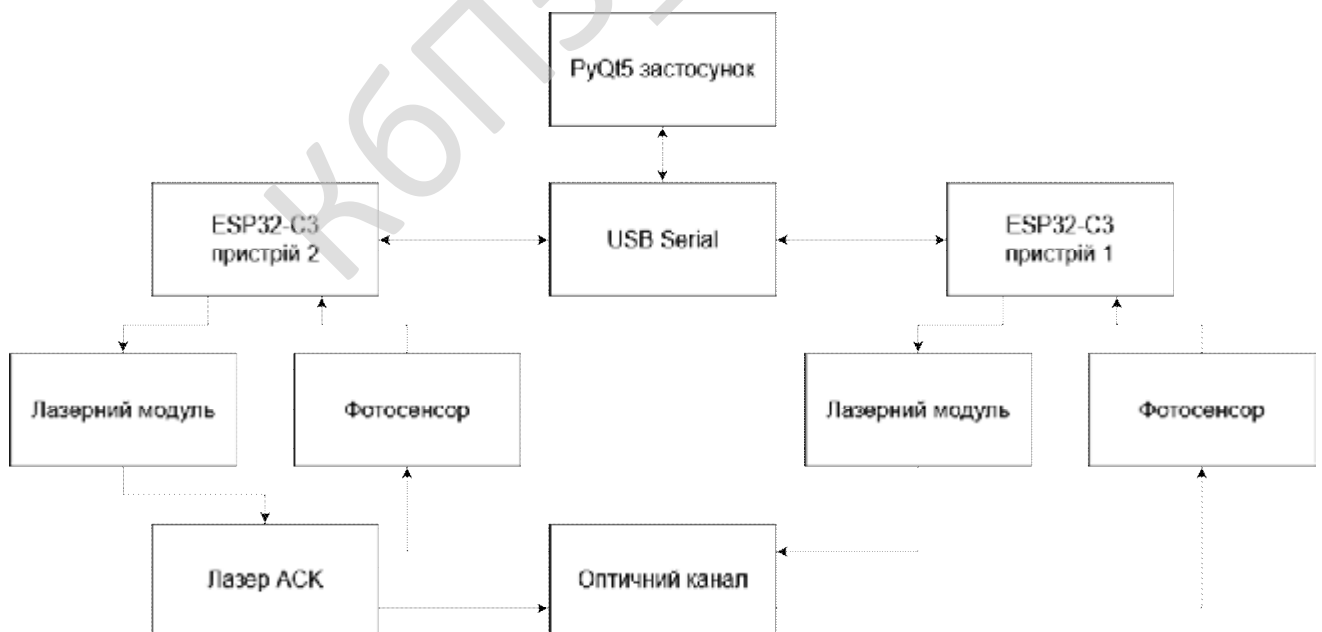


Рисунок 3.1 – Структурна схема

- Роль PyQt5-застосунка полягає у забезпеченні користувацького інтерфейсу та керування всією системою. Саме через нього користувач вводить текстові дані, налаштовує параметри передачі (швидкість, довжина імпульсів, пороги детекції тощо) та ініціює обмін даними. Також застосунок відповідає за відображення результатів передачі, логів та діагностичних графіків. Обмін з мікроконтролерами здійснюється через інтерфейс USB Serial, який виступає каналом керування та передачі даних між ПК і ESP32.

- Блоки ESP32-C3 (пристрій 1 і пристрій 2) є центральними елементами фізичної реалізації системи. Вони відповідають за реалізацію протоколу передачі даних, формування та обробку кадрів, а також генерацію та декодування сигналів. Один з пристроїв виконує роль передавача, інший – приймача, при цьому їх функції можуть бути симетричними залежно від режиму роботи системи. Саме на рівні ESP32 реалізуються таймінги, контроль послідовностей, CRC-перевірка та логіка ACK/повторів.

- Блок лазерного модуля використовується як фізичний передавач інформації. Він перетворює цифрові сигнали з ESP32 у світлові імпульси, формуючи спрямований оптичний канал. Саме за допомогою модуля реалізується кодування бітів у часові інтервали HIGH/LOW, що дозволяє передавати інформацію через простір без використання радіоканалу.

- На приймальній стороні використовується блок фотосенсора, який виконує функцію перетворення світлового сигналу назад у електричний. У реальній реалізації це фотодіод із підсилювальним каскадом, який дозволяє зчитувати слабкі зміни світлового потоку та перетворювати їх у придатний для обробки аналоговий сигнал. Далі цей сигнал оцифровується ESP32 через ADC і використовується для відновлення переданих бітів.

- Окремо на схемі виділено блок «Оптичний канал», який відображає середовище передачі між лазерним модулем і фотосенсором. Він є критичним елементом системи, оскільки саме в цьому середовищі відбувається вплив зовнішніх факторів – шуму, відстані, вирівнювання променя та освітлення.

- Також на схемі присутній блок «Лазер АСК», який відповідає за передачу підтверджень прийому. Він реалізує зворотний канал зв'язку, у якому короткі світлові імпульси використовуються для сигналізації успішного прийому кадру. Це дозволяє замкнути цикл передачі даних і реалізувати механізм надійної доставки з повторними передачами.

У цілому структурна схема відображає поділ системи на три логічні рівні: програмний (ПК), керуючий/обчислювальний (ESP32) та фізичний (оптичний канал), що забезпечує повну реалізацію концепції LaserLink як експериментальної системи оптичного зв'язку.

3.3 Розробка функціональної схеми

Функціональна схема системи LaserLink (див. Рисунок 3.2) є одним із ключових проектних документів, який зв'язує між собою постановку задачі, обрану архітектуру та фактичну реалізацію апаратно-програмного комплексу. На відміну від структурної схеми, що ілюструє склад фізичних компонентів і шин зв'язку між ними, функціональна схема зосереджена на діях, які виконує кожен блок, та на потоках даних, що пов'язують ці дії в єдиний обчислювальний процес. Такий зсув акценту дозволяє читачеві простежити, як саме формується повідомлення на стороні оператора, які перетворення воно проходить у настільному застосунку та в прошивці передавального вузла, як інформація переноситься оптичним каналом, відновлюється приймачем і повертається у вигляді результатів та логів на персональний комп'ютер.

Побудова функціональної схеми ґрунтується на загальноприйнятому в інженерній практиці принципі поділу за відповідальністю: кожен функціональний блок реалізує одну логічно завершену дію (введення тексту, формування кадру, обчислення контрольної суми, керування лазером тощо), а зв'язки між блоками відображають реальні шляхи передавання даних чи керуючих сигналів. Завдяки такому підходу схема залишається придатною до

					ВКРБ-122.26.0003.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		29

верифікації: за нею можна простежити, що жоден функціональний блок не висить у повітрі, а кожен потік даних має чітко визначеного відправника й одержувача. Це особливо важливо для випускної кваліфікаційної роботи, де функціональна схема використовується не лише як ілюстрація, а й як інструмент обґрунтування прийнятих проектних рішень.

Загалом схему природно поділити на чотири функціональні зони, які відповідають фізично відокремленим елементам комплексу. Верхня зона – це підсистема настільного застосунку, реалізованого на PyQt5. Вона виконує роль інтерфейсу користувача та координатора всього сеансу роботи. Середня зона – це канал Serial-обміну, що з'єднує застосунок із двома мікроконтролерами і несе на собі як корисні дані, так і службову телеметрію. Нижня права зона – це передавальний пристрій ESP32-C3 (далі пристрій 1), у якому реалізована повна ланка перетворення тексту в оптичні імпульси. Нижня ліва зона – приймальний пристрій ESP32-C3 (пристрій 2), який виконує зворотну послідовність дій і формує підтвердження ACK. Між пристроєм 1 і пристроєм 2 у функціональній схемі присутній окремий контур керування передачею, що відповідає за очікування підтвердження та повторні спроби; цей контур принципово відрізняє розроблений канал від односторонньої передачі без зворотного зв'язку.

					ВКРБ-122.26.0003.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		30

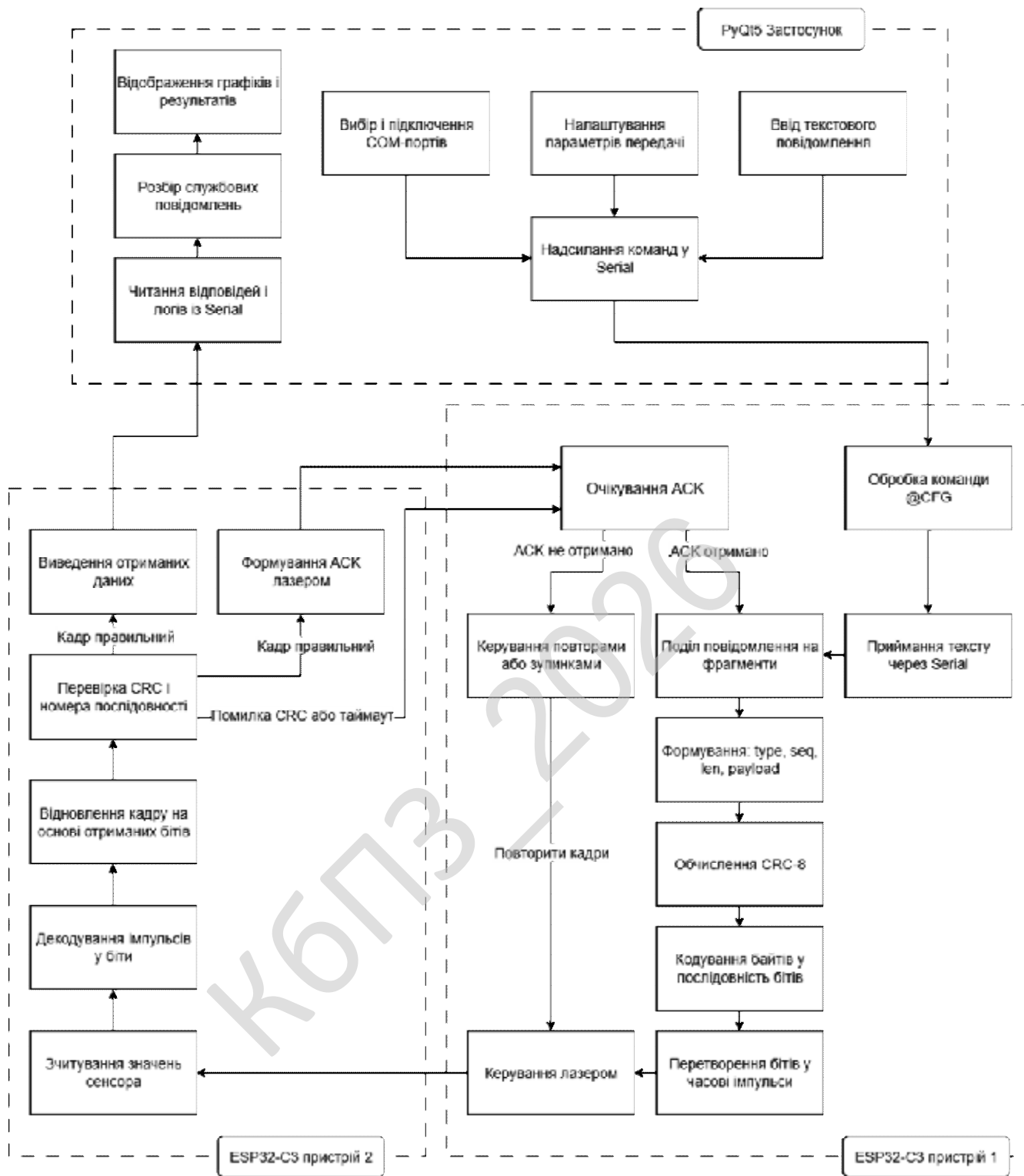


Рисунок 3.2 – Функціональна схема

Верхня частина функціональної схеми відображає настільний застосунок, побудований на бібліотеці PyQt5 та допоміжних модулях pyserial і pyqtgraph. Він

виконує дві принципово різні ролі. По-перше, це операторська консоль: оператор формує текстове повідомлення, обирає робочі СОМ-порти, задає параметри передачі та керує сеансом за допомогою екранних елементів. По-друге, це центр візуалізації та діагностики: усе, що повертається з пристроїв у вигляді службових рядків телеметрії, тут перетворюється на графіки, лічильники та елементи журналу. Така подвійна функція пояснює, чому в схемі застосунок представлений не одним блоком, а групою з шести взаємопов'язаних функціональних елементів.

Блок «Ввід текстового повідомлення» є першим за порядком дій і водночас одним із найпростіших за внутрішньою логікою. Він представлений у вкладці керування у вигляді багаторядкового поля введення, до якого оператор вводить довільний рядок – від короткого тестового повідомлення на кілька символів до фрагменту тексту, що завідомо перевищує максимальну довжину одного кадру протоколу й тим самим змусить підсистему передачі вмикати механізм фрагментації. На функціональній схемі цей блок зображено як одне з джерел вхідних даних, оскільки саме він починає ланцюг «оператор – застосунок – пристрій – оптика – пристрій – застосунок – оператор», який і є предметом дослідження. Важливо, що з точки зору функціональності блок введення не накладає змістових обмежень на повідомлення: усі обмеження (розмір корисного навантаження, поділ на фрагменти, кодування символів) виносяться у відповідні блоки прошивки, що дозволяє зосередити логіку «розуміння тексту» в одному місці і не дублювати її в інтерфейсі.

Далі блок «Розбір службових повідомлень» реалізує функцію парсера телеметрії. Він приймає на вхід завершені рядки, прочитані з порту, аналізує їх префікс (@S, @BIT, @RX_START, @RX_END, @TX_DONE, @CHUNK, @CFG) і перетворює на типізовані події (значення сенсора, таймінг біта, межі вікна прийому, підсумок передачі, статус застосування конфігурації тощо). На функціональній схемі цей блок зображено в центральній верхній частині, оскільки він стоїть на перехресті двох інформаційних потоків: сирий текст з порту та структуровані події для побудови графіків. Інженерна цінність парсера

					ВКРБ-122.26.0003.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		32

полягає в тому, що він локалізує знання про формат службових повідомлень в одному місці; будь-яке розширення протоколу телеметрії (додання нового префікса) вимагає змін лише в цьому блоці, не зачіпаючи решти інтерфейсу.

Між настільним застосунком та обома пристроями ESP32 у функціональній схемі присутні два логічно незалежні канали Serial-обміну, по одному на кожен фізичний пристрій. У апаратному плані вони реалізовані через інтерфейс USB-CDC (USB-UART) на швидкості 115200 бод, що відповідає узгодженій ініціалізації Serial.begin(115200) у прошивці. У функціональному плані Serial-канал не є «прозорим дротом» – він виконує цілий набір функцій, які варто виокремити.

По-перше, канал реалізує двонаправлений обмін: від ПК до пристрою передаються службові команди (@CFG, @STOP) і користувацький текст для оптичної передачі, тоді як у зворотному напрямку йде телеметрія, статуси та діагностичні рядки. По-друге, на Serial-канал покладено функцію граничного буферизатора: фрагменти даних, що надходять зі швидкістю, відмінною від швидкості обробки, тимчасово утримуються у внутрішніх буферах драйверів і прошивки, що згладжує короткочасні стрибки навантаження. По-третє, цей канал виконує роль сервісної шини: саме через нього оператор отримує можливість змінювати параметри пристрою під час роботи програми, без перепрошивки чи перезавантаження, що відповідає принципу гарячого конфігурування, поширеному в сучасних вбудованих системах.

Окрему функцію в межах Serial-каналу виконує розбір рядків на стороні прошивки: символи зчитуються по одному, ігнорується символ повернення каретки, а на символі нового рядка зібраний буфер передається у функціональний блок обробки команди. Така схема дозволяє реалізувати «людино-машинний» протокол, у якому повідомлення мають вигляд звичайних рядків, зручних для тестування з будь-якого монітора порту. Слід підкреслити: Serial-канал у LaserLink не є каналом передачі даних від користувача до користувача – це лише канал керування й діагностики, тоді як саму інформацію між двома вузлами

					ВКРБ-122.26.0003.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		33

переносить оптичний канал, описаний нижче.

Передавальна частина ESP32 (пристрій 1) представлена як послідовність функціональних блоків, що реалізують протокол передачі. Спочатку відбувається приймання тексту через Serial, після чого повідомлення розбивається на фрагменти. Для кожного фрагмента формується кадр із полями типу, номера послідовності, довжини та payload. Далі виконується обчислення контрольної суми CRC-8, після чого байти кодуються у послідовність бітів і перетворюються у часові імпульси. Завершальним етапом є керування лазером, який фізично передає сформований сигнал в оптичний канал.

Окремо у функціональній схемі виділено механізм контролю передачі. Після відправлення кадру передавач переходить у стан очікування підтвердження (ACK). Якщо ACK отримано, система переходить до наступного кадру; якщо ні – активується блок керування повторами або зупинкою передачі, що забезпечує надійність обміну.

Приймальна частина ESP32 (пристрій 2) виконує зворотну послідовність операцій. Спочатку зчитуються значення з фотосенсора, після чого імпульси декодуються у біти. На основі отриманих бітів відновлюється структура кадру, після чого виконується перевірка CRC і номера послідовності. Якщо кадр правильний, дані передаються у настільний застосунок, а також формується сигнал підтвердження ACK за допомогою лазера. У випадку помилки CRC або таймауту відповідний кадр ігнорується, що призводить до його повторної передачі з боку передавача.

Завершальним елементом функціональної схеми є блок діагностики та візуалізації, який реалізований у настільному застосунку. Він відповідає за відображення отриманих даних, логів роботи системи та графіків сигналу, що дозволяє користувачу аналізувати якість передачі та вплив параметрів системи.

Таким чином, функціональна схема демонструє повний цикл обробки інформації в системі LaserLink – від введення даних користувачем до їх передачі,

					ВКРБ-122.26.0003.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		34

прийому, перевірки та відображення результатів із урахуванням механізмів контролю помилок і повторів.

3.4 Розробка діаграми взаємодії процесів

Діаграма взаємодії процесів відображає послідовність виконання основних операцій у системі LaserLink та взаємозв'язки між ними. На діаграмі (див. Рисунок 3.3) всі елементи представлені у вигляді овалів, що відповідають окремим процесам або етапам роботи системи. Стрілки між ними позначають переходи, які відбуваються після завершення відповідної дії або за певної умови. Таким чином, діаграма описує логіку виконання системи як послідовність станів.

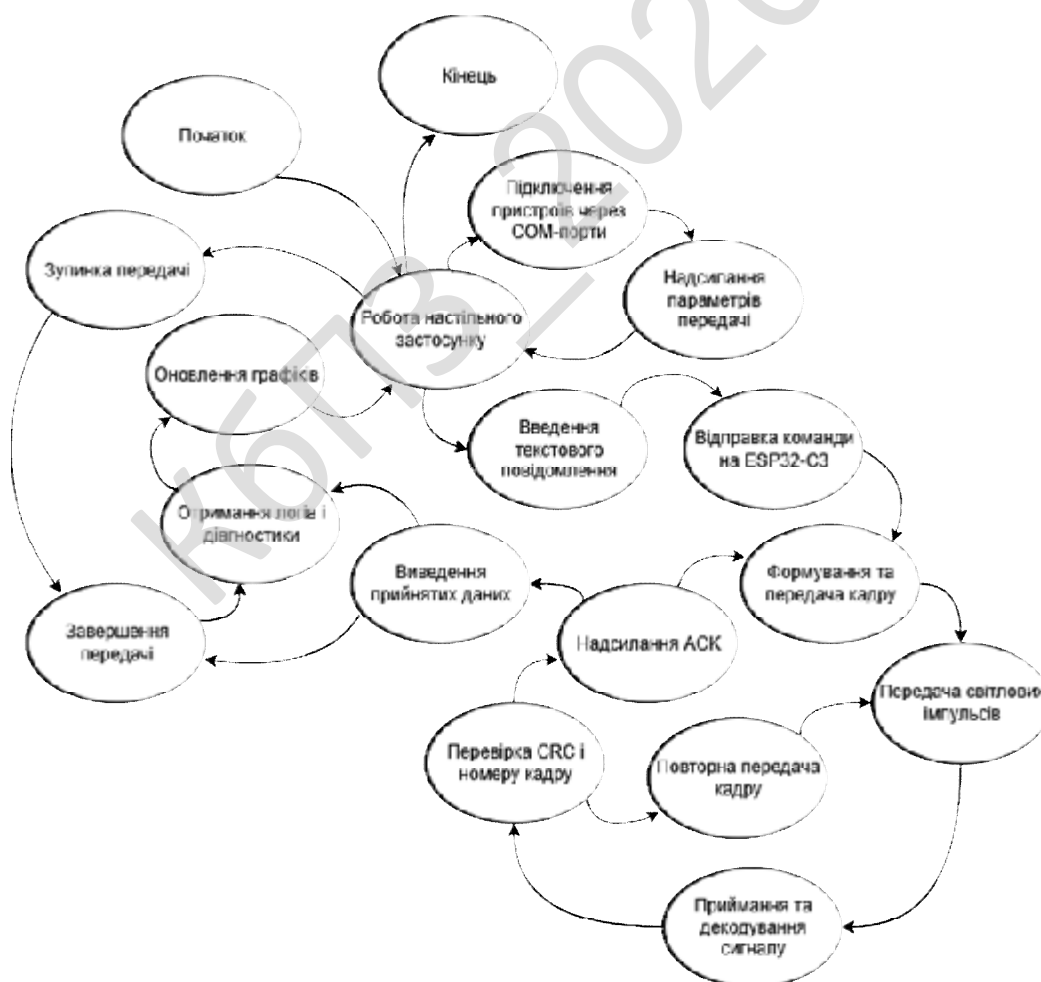


Рисунок 3.3 – Діаграма взаємодії процесів

Центральним елементом є процес роботи настільного застосунку. Саме він координує всі інші етапи: користувач через нього підключає пристрої, надсилає параметри передачі, вводить текстові повідомлення та ініціює передачу. Крім того, застосунок відповідає за отримання логів, відображення результатів і побудову графіків, що робить його ключовою точкою взаємодії з системою.

Після початку роботи відбувається підключення пристроїв через COM-порти та надсилання параметрів передачі. Далі користувач вводить повідомлення, яке передається на ESP32 і обробляється передавальною частиною: формується кадр і виконується його передача через оптичний канал. На приймальній стороні відбувається декодування сигналу, перевірка CRC і номера кадру.

Важливою частиною діаграми є розгалуження після перевірки кадру. У разі успішного прийому формується підтвердження АСК, після чого система може переходити до передачі наступного кадру та виведення отриманих даних. Якщо ж виявлено помилку або не отримано підтвердження, активується процес повторної передачі, який повертає систему до етапу формування і відправлення кадру. Таким чином реалізується цикл забезпечення надійності передачі.

Окремо виділено блоки, пов'язані з діагностикою: отримання логів, їх обробка та оновлення графіків. Після побудови графіків керування знову повертається до настільного застосунку, що утворює цикл безперервного моніторингу та взаємодії з користувачем.

Ще один важливий процес – це зупинка передачі, яка може бути ініційована користувачем у будь-який момент. Після цього система переходить у стан завершення роботи. Таким чином, діаграма чітко демонструє два ключові цикли: цикл взаємодії з користувачем через інтерфейс (з поверненням від графіків до основного застосунку) та цикл повторної передачі кадрів у разі помилок. Це підкреслює як інтерактивний характер системи, так і наявність механізмів забезпечення надійності передачі даних.

4 РЕАЛІЗАЦІЯ ПРОЕКТУ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ПРАВИЛЬНІСТЬ ПРОЕКТНИХ РІШЕНЬ

Об'єктом розробки у випускній кваліфікаційній роботі є програмно-апаратний комплекс LaserLink, призначений для організації оптичного каналу зв'язку між двома вузлами на базі мікроконтролера ESP32-C3 та супроводу цього каналу настільним застосунком на мові Python із графічним інтерфейсом PyQt5. У межах цього розділу авторське рішення не обмежується переліком файлів чи коротким описом інтерфейсу користувача: на меті стоїть показати, як саме реалізовані задекларовані у попередніх розділах функції – від розбиття повідомлення на кадри й обчислення контрольної суми до узгодженої роботи потоків у застосунку та збору експериментальних даних, які дозволяють судити про коректність і стійкість системи в реальних умовах експлуатації.

З точки зору програмної реалізації комплекс можна поділити на вбудоване та прикладне програмне забезпечення, які співіснують у єдиному експериментальному контурі, але виконують різні за важливістю ролі. Вбудований компонент зібраний у середовищі PlatformIO для плати esp32-c3-devkitm-1 (конфігурація esp32_tcp у файлі src/device/platformio.ini) і реалізує безпосередню взаємодію з лазером і фотоприймачем, тобто тим «вузлом довіри», на якому лежить відповідальність за фізичну передачу бітів, тайм-аути та повторні спроби. Швидкість обміну з персональним комп'ютером за послідовним інтерфейсом вибрана 115200 бод; це типове для діагностичних і лабораторних режимів значення, яке забезпечує достатню швидкість для команд і журналу без надмірного навантаження на обробку рядків у застосунку.

Прикладний компонент – застосунок LaserLink App у каталозі src/app – не дублює низькорівневу модуляцію оптичного сигналу, а виконує роль операторської консолі та середовища візуалізації. Він користується бібліотекою pyserial для доступу до COM-портів і pyqtgraph для швидкого оновлення графіків,

					ВКРБ-122.26.0003.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		37

що важливо, коли з пристрою надходить щільний потік діагностичних рядків. Таким чином, архітектурно виникає класична для вбудованих систем схема: основна логіка зосереджена на мікроконтролері, тоді як ПК дає зручність налаштування, спостереження та збереження результатів у межах одного сеансу роботи.

Варто відразу зафіксувати методичний момент, стан сеансу на стороні ПК тримається в оперативній пам'яті (зокрема, у структурах на кшталт deque у класі ChartDataStore), а архівом виступає текстовий журнал послідовного порту, який за бажання можна зберегти у файл уже поза програмою. На стороні пристрою конфігурація зберігається в глобальній структурі DeviceConfig, а транспортні пакети описуються структурою Frame.

4.1 Блок-схеми та опис алгоритмів функціонування системи

Якщо абстрагуватися від деталей реалізації, система виглядає як ланцюг «людина – програма на ПК – послідовний порт – мікроконтролер – оптика», де кожен елемент має власну зону відповідальності, і саме в узгодженні цих зон полягає працездатність усього комплексу. Користувач задає не лише що передати, а й умови, у яких це має відбутися: пресет швидкості, поріг спрацьовування сенсора, потужність лазера, тайм-аути квитанції тощо. Ці рішення не залишаються на рівні абстракції: вони перетворюються на конкретні текстові команди й потрапляють у прошивку, де вже перетворюються на тривалості імпульсів і пороги АЦП.

LaserLink App виступає центральним вузлом прикладного рівня. Він одночасно може тримати два незалежні підключення до COM-портів через віджети SerialBlock, що зручно в лабораторній постановці, коли один порт слухає пристрій біля передавача, а другий – пристрій біля приймача, або коли обидва пристрої підключені до одного комп'ютера для відлагодження. Вкладка Control концентрує на собі операції керування та журналювання, тоді як вкладка Charts

					ВКРБ-122.26.0003.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		38

відводиться під накопичення статистики в ChartDataStore і її відображення (див. Рисунок 4.1). Таке розділення дозволяє знизити когнітивне навантаження на оператора.

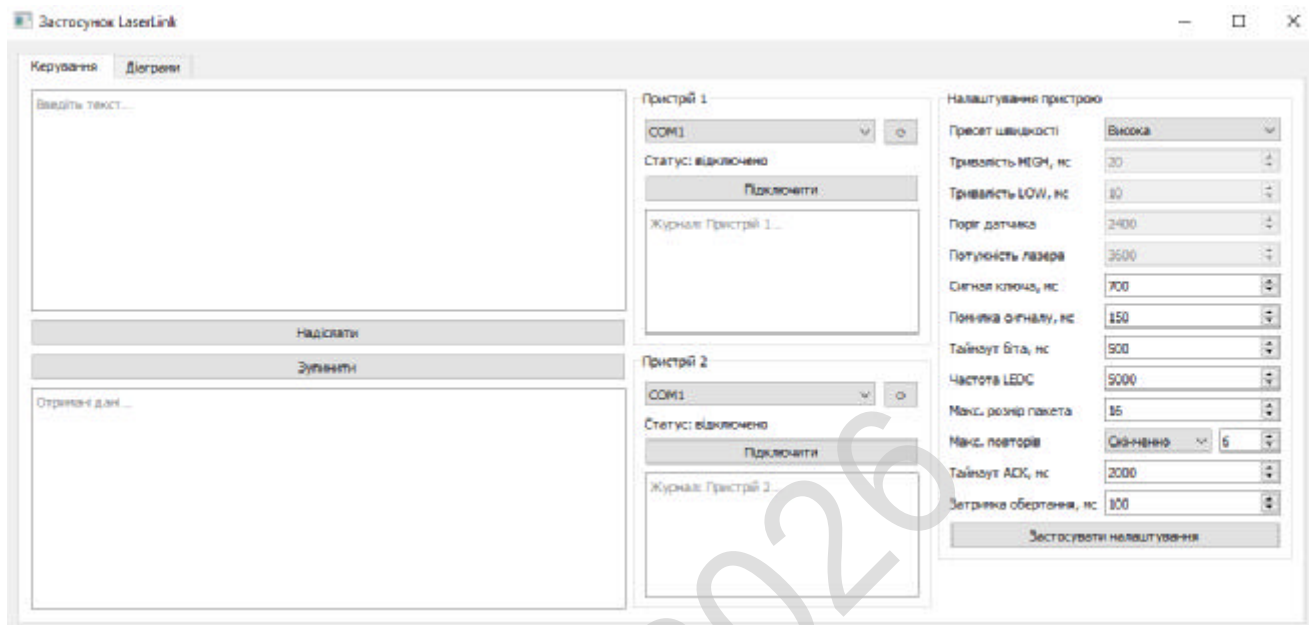


Рисунок 4.1 – Графічний інтерфейс застосунку

USB-UART у цьому контексті є повноцінним каналом керування та телеметрії. Його швидкість узгоджена з ініціалізацією `Serial.begin(115200)` у прошивці, що виключає розбіжність форматів і спрощує відтворення експерименту на іншій машині. Фрагмент ініціалізації з `setup` ілюструє узгодження швидкості UART і безпечний старт периферії:

```
void setup() {
    Serial.begin(115200);

    laser.begin();
    laser.turnOff();

    sensor.begin();
    sensor.begin();
}
```

Мікроконтролер ESP32-C3 реалізує основний цикл setup/loop, розбір рядків, застосування команди @CFG, а також два принципово різні режими оптичної активності: надійну відправку через sendFrameReliable і прийом з підтвердженням через receiveAndAck. Нарешті, лазер і сенсор утворюють фізичний шар: саме вони визначають, наскільки чистим буде сигнал на межі шуму й наскільки стабільно приймач зможе відрізнити службовий ключовий імпульс від інформаційних бітів.

З погляду потоків даних варто розрізняти три типові ситуації, які у реальному сеансі чергуються або накладаються одна на одну. Налаштування починається з дії оператора на формі, після чого у пам'яті формується довгий рядок команди @CFG з переліком пар ключ=значення; цей рядок відправляється на кожен підключений пристрій, а прошивка, пройшовши через applyDeviceConfigCommand, або атомарно оновлює deviceConfig і відповідає @CFG OK, або повертає зрозуміле повідомлення про помилку – що важливо для діагностики без перезавантаження. Передача повідомлення виглядає для користувача як одна дія «Надіслати», але всередині ланцюжка це спочатку буферизація тексту в SerialReader, потім розпізнавання повного рядка в handleSerialLine, виставлення прапорця shouldSend і, нарешті, довга фаза оптичної передачі в sendFrameReliable. Телеметрія, яка активується макросом ENABLE_DEBUGGING у config.h, не є обов'язковою для чистої передачі, але саме вона перетворює систему з чорного ящика на об'єкт дослідження: рядки @S, @BIT, межі прийому @RX_START/@RX_END, підсумки @TX_DONE потрапляють у LogParser і далі осмислюються як події для графіків.

Інтерфейс між прикладним програмним забезпеченням і операційною системою у даній роботі є стандартним послідовним портом, який у сучасних відладочних платах ESP32-C3 зазвичай з'являється як пристрій USB-CDC. Це рішення має ту перевагу, що воно не вимагає написання власного драйвера й добре підтримується бібліотеками на Python. Водночас саме тут проявляється типова для GUI-застосунків проблема: тривале блокування читання COM-порту в

					ВКРБ-122.26.0003.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		40

головному потоці призвело б до «заморожування» вікна. Тому об'єкт SerialReader переноситься в окремий QThread, а результати читання доставляються в інтерфейс через сигнали data_received, error_occurred та finished. Такий підхід відповідає рекомендованій моделі Qt і одночасно слугує ілюстрацією того, як у дипломній роботі поєднуються вимоги реального часу на мікроконтролері та відгуку інтерфейсу на ПК.

На рівні прошивки взаємодія з персональним комп'ютером організована як простий рядковий протокол: команди керування, діагностичні запити та повідомлення для користувача передаються у вигляді тексту – читабельного без спеціальних інструментів, зручного для ручного введення з терміналу й для легкого посимвольного або порядкового парсингу на стороні ПК. Такий формат знижує поріг входу для налагодження й документування поведінки пристрою.

Натомість коли мова йде про фактичну доставку користувацького змісту по оптичному каналу, нижньорівневим носієм уже не виступає «голий» рядок символів, а структурований бінарний кадр FRAME_DATA з чітко визначеними полями: тип кадру або повідомлення, порядковий номер для відстеження послідовності та повторів, довжина корисного навантаження, самі дані користувача та контрольна сума CRC для виявлення пошкоджень і випадкових спотворень під час передачі. Це відповідає вимогам надійності та передбачуваного розбору на приймачі в умовах обмеженої пропускної здатності й можливих перешкод. Таким чином у системі одночасно співіснують два рівні представлення інформації: текст по UART, орієнтований на зручність експлуатації та інтеграції зі стандартним серійним терміналом; і канално-орієнтований – компактний кадр для оптичного лінку, орієнтований на цілісність, адресацію логіки протоколу та ефективне використання середовища передачі.

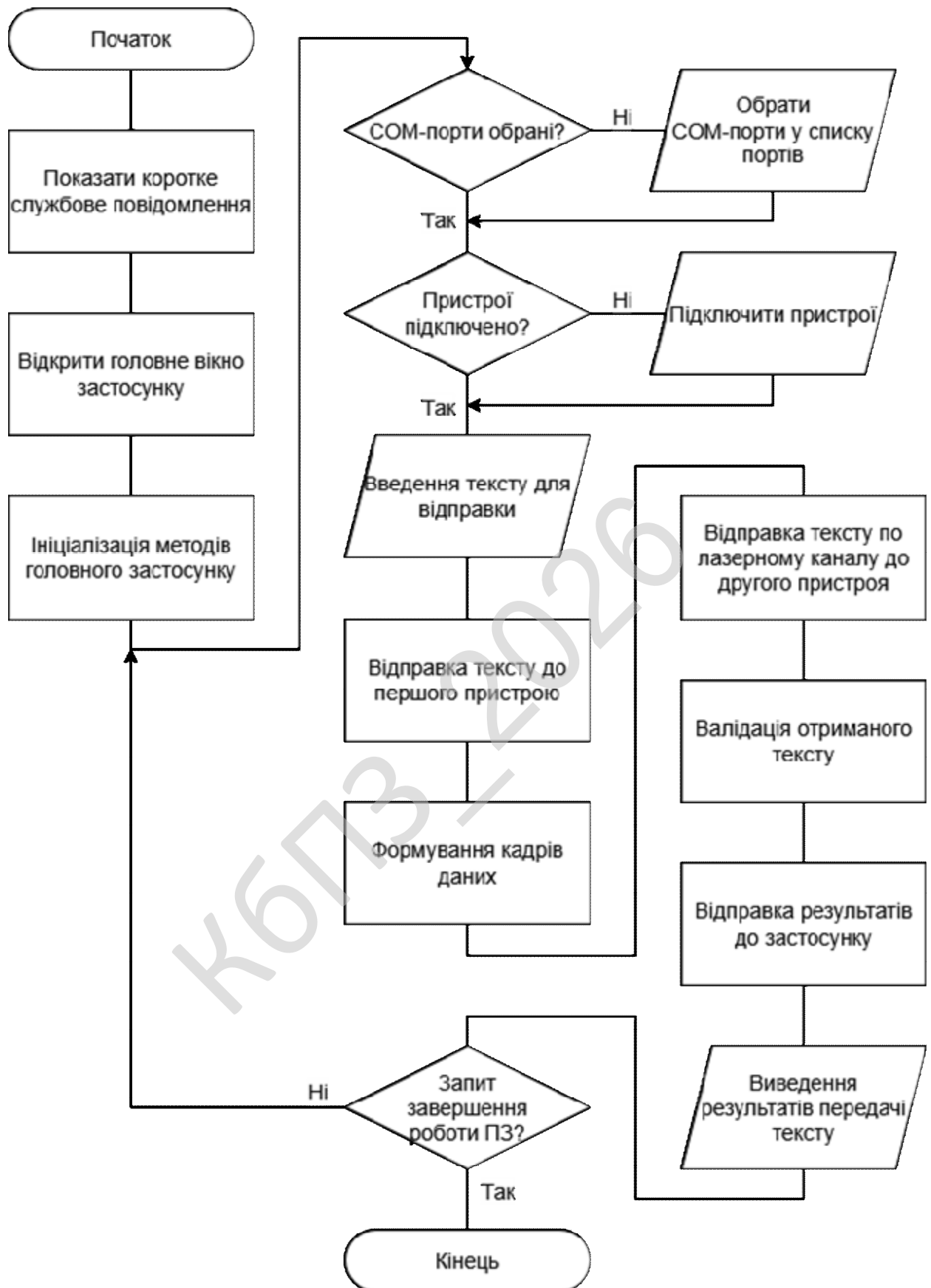


Рисунок 4.2 – Блок схема основної програми

Ініціалізація

Робота вбудованого програмного забезпечення виглядає для зовнішнього спостерігача як безперервний цикл, у якому чергуються слухання ПК і робота оптичного каналу. Проте всередині цього циклу закладено кілька важливих інженерних рішень, які варто пояснити зв'язним текстом, а не лише посиланням на вихідний код: вони показують, як автор урахував безпеку запуску, можливість зміни параметрів на льоту та необхідність не блокувати систему надто довгими очікуваннями там, де це можна уникнути.

Функція `setup` виконує роль холодного старту пристрою. На її початку ініціалізується послідовний порт із швидкістю 115200 бод, що задає загальний ритм обміну з ПК і водночас слугує орієнтиром для лабораторного журналу: усі подальші експерименти природно документувати з цією ж швидкістю, щоб уникнути непорозумінь. Далі створюються об'єкти `Laser` та `Sensor`, прив'язані до пінів, визначених у `config.h`. Важливо підкреслити порядок викликів `laser.begin()`, `laser.turnOff()` та `sensor.begin()`: лазер після ініціалізації одразу переводиться у вимкнений стан, що знижує ризик випадкового випромінювання під час подальших кроків запуску або під час перезавантаження плати. Якщо у збірці ввімкнено `ENABLE_DEBUGGING`, додатково створюється задача FreeRTOS `sensorReportTask` з пріоритетом 1 і стеком 2048 байт; вона працює паралельно з основним циклом і періодично виводить рівень сенсора, що робить поведінку пристрою прозорою для дослідника, хоча й збільшує навантаження на UART.

Окремо слід згадати функцію `applyRuntimeHardwareConfig()`, яка виконує перехід від параметрів у пам'яті до параметрів у залізі. Після успішного застосування команди `@CFG` саме вона оновлює потужність лазера та поріг сенсора відповідно до полів `deviceConfig`, тобто забезпечує узгодженість між тим, що оператор бачить у формі на ПК, і тим, що реально визначає поріг спрацьовування АЦП та яскравість випромінювання.

Основний цикл

Основний цикл loop поєднує дві логічні гілки, які конкурують за процесорний час, але не є паралельними у сенсі витоків гонок даних: вони послідовно обробляються в межах однієї ітерації, що спрощує міркування про коректність. Перша гілка – це обробка вхідних даних з UART. Поки у буфері є символи, вони зчитуються по одному; символ повернення каретки ігнорується, а на символі нового рядка зібраний рядок передається в `handleSerialLine`, після чого буфер рядка очищується. Такий підхід є класичним для мікроконтролерних протоколів «людина–машина», де команди природно мають вигляд текстових рядків. Друга гілка активується, коли попередня логіка встановила необхідність передати повідомлення (`shouldSend`), або коли пристрій слухає ефір і намагається розпізнати початок вхідного ключового сигналу на сенсорі. У першому випадку викликається `sendFrameReliable` із функцією перевірки зупинки, у другому – після проходження фільтрації шуму викликається `receiveAndAck`. У прошивці ця логіка відтворена безпосередньо у функціях `handleSerialLine` та `handleStopCommand` файлу `main.cpp`. Спочатку перевіряються службові команди, далі конфігурація, і лише за відсутності збігів рядок трактується як корисне навантаження для оптичної передачі.

Коли з боку даного вузла не заплановано жодної передачі, основний цикл не простоює у марному очікуванні й не блокує виконання довгими затримками: натомість він періодично виконує неблокуюче спостереження за станом сенсора, залишаючись готовим до інших задач і не втрачаючи можливості вчасно відреагувати на сигнал від партнера по лінії.

Тут варто окремо підкреслити деталь, яку в описах реалізації часто або згадують поверхнево, або взагалі недооцінюють: сирий електричний сигнал з фотоприймача не завжди має ідеальну прямокутну форму й чіткі межі кадру. На практиці він може містити короткі сплески та накладені перехідні процеси, які зовні нагадують початок валідного кадру за формою фронту, але за тривалістю та стабільністю рівня не відповідають вимогам протоколу. Якщо трактувати будь-

який такий імпульс як початок повідомлення, система почне хибно бачити кадри там, де їх немає, що різко знижує надійність обміну.

Саме тому в коді закладено послідовну перевірку: спочатку застосовується поріг `sensorThreshold`, щоб відсіяти слабкі та нестабільні коливання; далі вимагається мінімальна тривалість утримання лінії у стані HIGH – константа `KEY_DETECT_MIN_MS` (за замовчуванням 50 мс у `config.h`), що відсікає надто короткі сплески. Лише після підтвердження, що високий рівень тримається достатньо довго, вимірюється повна тривалість імпульсу й порівнюється з очікуваною `keySignalDurationMs` з урахуванням допуску `signalErrorMs`, який компенсує джиттер таймера, похибки вимірювання та невеликі відхилення апаратури. Тільки коли всі ці умови виконані, викликається `receiveAndAck`.

К6ПЗ-2026

					ВКРБ-122.26.0003.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		45

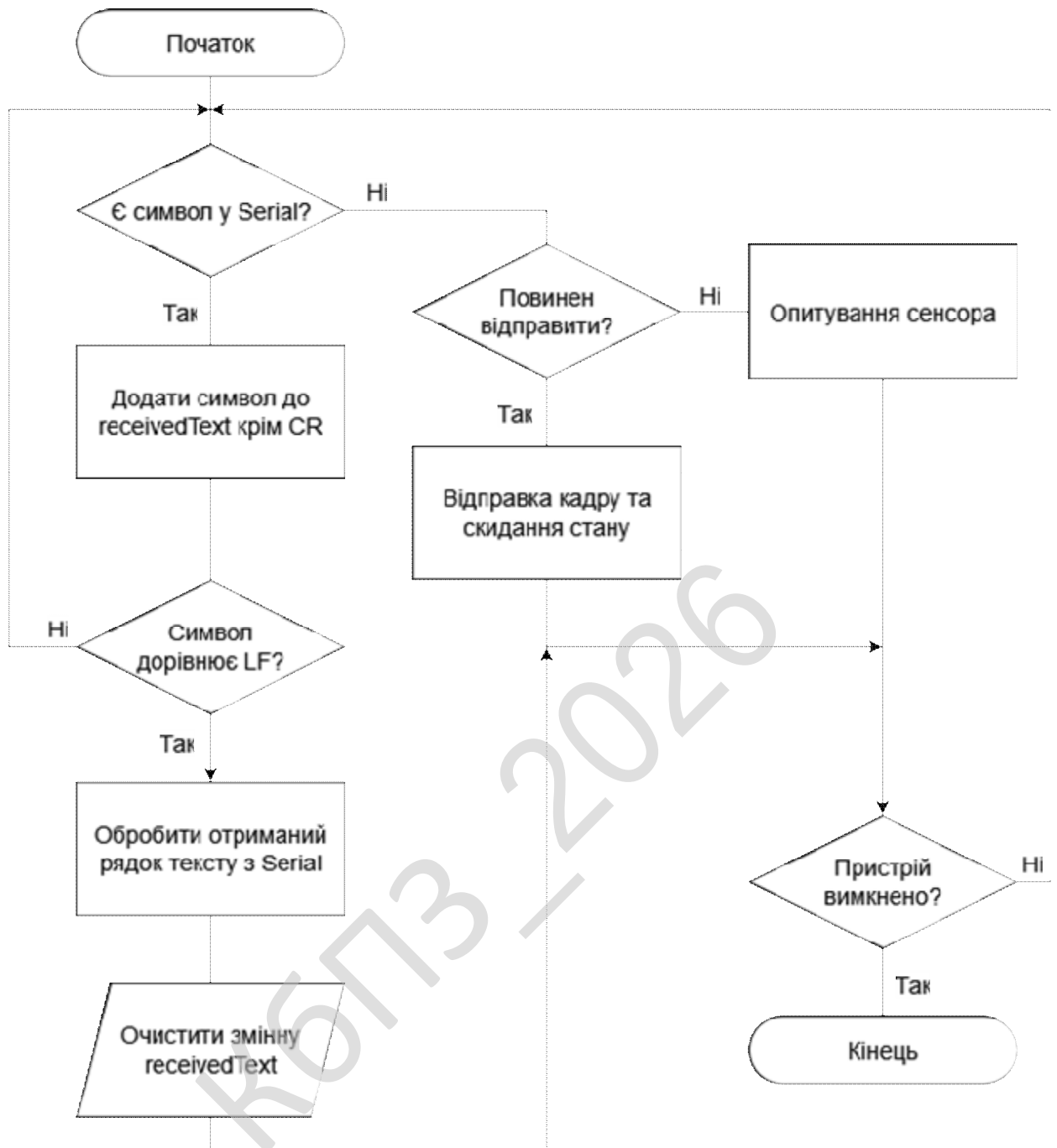


Рисунок 4.3 – Блок-схема роботи підпрограми

Надійня передача повідомлення

Функція `sendFrameReliable`, визначена у `protocol_handler.h`, є серцем транспортної логіки: вона перетворює довгий рядок, який зручно вводити людині, на послідовність кадрів, кожен з яких має шанс бути доставленим навіть за наявності короткочасних перешкод у оптиці. Реалізація свідомо поєднує два

механізми – фрагментацію (обмеження довжини шматка значенням `deviceConfig.maxPayloadSize`) та підтвердження з повтором (очікування імпульсу АСК у вікні `ackTimeoutMs` з кількістю спроб за `maxRetries`).

Під час виконання `sendFrameReliable` передача може тривати довго через повторні спроби та очікування АСК. Щоб користувач міг безпечно перервати процес з ПК, у якості аргументу `abortCheck` передається вказівник на функцію `pollTransferStopRequested`. Вона зчитує всі наявні в буфері UART символи, збирає рядок до символу `\n` і перевіряє рівність `@STOP`. Якщо команда розпізнана, виставляються прапорці `transferStopRequested` та `shouldSend = false`, лазер вимикається, на порт виводиться `@STOP OK`, а `sendFrameReliable` через `finishStoppedTransfer` завершує передачу з результатом `STOP` у діагностичному лозі (`@TX_DONE ... result=STOP`). Таким чином, реалізовано кооперативну схему переривання: прошивка не використовує апаратні переривання UART для цієї логіки, але гарантує регулярні точки виходу між спробами та після відправки кадру.

У кінці `loop` реалізовано фільтр коротких сплесків на сенсорі. Поки значення ADC нижче порогу, стан що сенсор бачить `HIGH` – скидається. Якщо рівень перевищив поріг, фіксується час початку `HIGH`. Якщо `HIGH` тримається менше `KEY_DETECT_MIN_MS` (50 мс), це може бути електромагнітний шум або відблеск – кадр не відкривається. Якщо `HIGH` тримається довше, після переходу до `LOW` вимірюється повна тривалість імпульсу; вона порівнюється з `keySignalDurationMs` з допуском `signalErrorMs` (зверху додано запас +1000 мс для верхньої межі згідно з кодом). Лише при виконанні цих умов викликається `receiveAndAck`. Такий підхід зменшує ймовірність хибного старту прийому та зайвих викликів `receiveFrame`, що економить процесорний час і знижує кількість таймаутів на рівні бітового читання.

Алгоритм оптичного протоколу передачі

Оптичний рівень у даній роботі не є прозорим каналом у байтах, як Ethernet або USB високого рівня. Він вимагає явного узгодження часових

					ВКРБ-122.26.0003.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		47

параметрів і формату пакета. Тому в прошивці введено чітке поняття кадру – структури даних у пам'яті мікроконтролера, яка відображає угоду між передавачем і приймачем про те, що саме вважати початком повідомлення, як відрізняти службові імпульси від інформаційних бітів і як упакувати корисний зміст так, щоб на приймальній стороні можна було перевірити його цілісність.

У заголовку `protocol.h` зафіксовано перелік типів кадрів, зокрема `FRAME_DATA` (0x01), `FRAME_ACK` (0x02) та `FRAME_NAK` (0x03). У поточній реалізації підтвердження для передавача формується не окремим байтовим кадром на оптиці, а імпульсом ключа через `sendKeySignal` у `sender.h`. Для передавача відсутність очікуваного підтвердження в межах тайм-ауту означає ту саму роль, яку у класичних протоколах часто позначають як NAK: необхідність повторити попередню спробу. Такий підхід економить складність приймача, але вимагає ретельного підбору `ackTimeoutMs` і `turnaroundDelayMs`, інакше система або буде надто повільною, або надто чутливою до випадкових пауз.

Структура `Frame` у пам'яті пристрою включає поля `type`, `seq`, `len`, масив `payload` розміру `FRAME_PAYLOAD_CAPACITY` (64 байти за замовчуванням у `config.h`) та байт `crc`. Сенс полів інтуїтивний: тип задає семантику пакета, `seq` дозволяє відрізняти нові шматки повідомлення від повторів, `len` фіксує фактичну довжину корисного навантаження, а `crc` підсумовує узгодженість усіх попередніх байтів заголовка й даних. Нижче приведено Таблицю 4.1 відповідності поле-значення структури `Frame`:

Таблиця 4.1 – відповідність поле-значення структури `Frame`.

Поле	Тип	Призначення
<code>type</code>	<code>uint8_t</code>	Тип кадру (<code>FRAME_DATA</code> тощо).
<code>seq</code>	<code>uint8_t</code>	Порядковий номер шматка повідомлення.
<code>len</code>	<code>uint8_t</code>	Довжина корисного навантаження в байтах.
<code>payload</code>	масив <code>uint8_t</code>	Дані користувача.
<code>crc</code>	<code>uint8_t</code>	CRC-8 по байтах <code>type</code> , <code>seq</code> , <code>len</code> та <code>payload</code> .

У кодї структура кадру та обчислення контрольної суми зафіксовані компактно в одному заголовку `protocol.h`, що полегшує супровід і унеможливорює розбіжність між розміром буфера та константами протоколу:

Обмеження на `len` виконують одночасно дві функції: вони не дають «роздути» кадр понад узгоджену з конфігурацією межу `maxPayloadSize` і не дозволяють вийти за фізичну ємність буфера `FRAME_PAYLOAD_CAPACITY`. Якщо поле довжини виявляється некоректним, `receiveFrame` повертає `RX_CRC_ERR` у цьому спеціальному випадку, що змушує передавача повторити спробу; з точки зору безпеки пам'яті це краще, ніж намагатися читати дані в невідомий за розміром буфер.

Алгоритм CRC-8

Контрольна сума реалізована як CRC-8 з поліномом `0x31` і початковим значенням `0x00` у функції `crc8` файлу `protocol.h`. На кожному байті даних алгоритм виконує операцію XOR між поточним CRC і байтом, після чого вісім разів зсуває CRC вліво, застосовуючи XOR з поліномом, коли виставлений старший біт. CRC не шифрує дані й не приховує зміст, але дає високу ймовірність виявлення пакетів, які спотворилися через шум або короткі збої фотоприймача. Функція `computeFrameCrc` формує тимчасовий буфер з полів `type`, `seq`, `len` і фактично переданих байтів навантаження, після чого повертає однобайтовий результат – саме він відправляється останнім у межах кадру між завершальними ключовими імпульсами.

Приклад розрахунку CRC

Розглянемо мінімальний кадр без корисного навантаження: `type = 0x01`, `seq = 0x00`, `len = 0x00`. Буфер для CRC складається з трьох байтів `{0x01, 0x00, 0x00}`. За реалізацією, ідентичною коду прошивки (перевірено окремим скриптом), отримуємо `CRC = 0x46`. Якщо ж додати один байт даних `0x41` (символ «А») при `seq = 2` та `len = 1`, буфер `{0x01, 0x02, 0x01, 0x41}` дає `CRC = 0xEF`. Ці два приклади показують, що зміна будь-якого байта заголовка або навантаження

кардинально змінює контрольну суму, тобто механізм дійсно чутливий до помилок, а не декоративний.

Формування та відправка кадру

Відправка кадру виглядає комплексно, але при цьому має чітку структуру. Спочатку випромінюється стартовий ключовий сигнал `sendKeySignal`, який задає початок повідомлення у часі. Далі послідовно передаються байти `type`, `seq` і `len`, кожен з яких розкладається на біти через `sendByte`, а кожен біт кодується чергуванням HIGH/LOW тривалостями з `DeviceConfig`. Після цього в ефір виходить саме корисне навантаження, а завершує кадр байт `crc` і ще один ключовий імпульс. Таким чином, приймач завжди має шанс відрізнити тишу від неповного кадру і від кадру, який закінчився – хоча практична надійність усе одно залежить від правильного підбору порогів і тайм-аутів на рівні `readBit`. Нижче приведено реалізацію відправки кадру:

```
inline void sendFrame(const Laser &laser, const Frame &frame) {
    sendKeySignal(laser);
    sendByte(laser, frame.type);
    sendByte(laser, frame.seq);
    sendByte(laser, frame.len);

    for (uint8_t i = 0; i < frame.len; i++) {
        sendByte(laser, frame.payload[i]);
    }

    sendByte(laser, frame.crc);
    sendKeySignal(laser);
}
```

Після стартового ключового імпульсу кожен інформаційний біт зчитується як послідовність трьох фаз: HIGH, LOW, HIGH. Тривалості HIGH між першим і другим переходами (`rxHighDuration`) та LOW між другим і третім (`rxLowDuration`) порівнюються функцією `oneOrZero`: якщо перша HIGH-фаза довша за LOW, біт інтерпретується як 1, інакше як 0 (молодший біт першим, узгоджено з `sendByte`). Якщо перша HIGH-фаза за тривалістю нагадує ключовий сигнал (у межах `keySignalDurationMs` та `signalErrorMs`), повертається

					ВКРБ-122.26.0003.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		50

BIT_TIMEOUT – це розділяє службові імпульси від даних. Глобальний лічильник rxStartTime та timeoutMs обмежують максимальну тривалість читання кадру, запобігаючи вічному блокуванню при втраті синхронізації.

Прийом кадру

Після того як у loop розпізнано початковий ключовий сигнал, функція receiveFrame послідовно збирає кадр із ефіру. Спочатку зчитуються три службові байти – тип, порядковий номер і довжина, – кожен як послідовність із восьми бітів, отриманих через readBit. Якщо на будь-якому кроці очікування порушується часова логіка, повертається RX_TIMEOUT, що відповідає ситуації втрати синхронізації або сильного заглушення сигналу. Якщо ж поле довжини виявляється зовеликим, подальше читання навантаження не розпочинається, а результат інтерпретується як RX_CRC_ERR, що змушує передавача повторити кадр і водночас захищає пам'ять від некоректного обсягу даних. У штатному сценарії після успішної перевірки довжини зчитуються len байтів навантаження та байт CRC, після чого обчислена контрольна сума порівнюється з прийнятою. Невідповідність знову ж таки завершує гілку RX_CRC_ERR, тоді як успіх повертає RX_OK. Завершальний виклик drainEndKeySignal виглядає технічною дрібницею, але має змістовне обґрунтування. Він знімає хвіст імпульсу передавача, щоб наступний цикл не сприйняв залишкове світіння як початок нового кадру. Нижче наведено фрагмент receiveFrame, де видно перевірку довжини до читання payload та порівняння прийнятого CRC з обчисленням:

```
if (frame.len > deviceConfig.maxPayloadSize || frame.len > FRAME_PAYLOAD_CAPACITY)
{
    frame.crc = 0;
    logRxEnd("CRC_ERR");
    return RX_CRC_ERR;
}
for (uint8_t i = 0; i < frame.len; i++) {
    if (!receiveByte(sensor, frame.payload[i])) {
        logRxEnd("TIMEOUT");
        return RX_TIMEOUT;
    }
}
```

					ВКРБ-122.26.0003.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		51

```

if (!receiveByte(sensor, frame.crc)) {
    logRxEnd("TIMEOUT");
    return RX_TIMEOUT;
}
uint8_t expected = computeFrameCrc(frame);
if (frame.crc != expected) {
    logRxEnd("CRC_ERR");
    return RX_CRC_ERR;
}

```

Функція `receiveAndAck` з'єднує фізичний прийом із семантикою транспортного рівня. Якщо `receiveFrame` повернув успіх і тип кадру відповідає `FRAME_DATA`, приймач порівнює `frame.seq` із збереженим `lastReceivedSeq`. Збіг означає, що передавач повторив уже прийнятий раніше шматок – типова ситуація після втрати першого ACK. У такому випадку приймач знову відправляє підтвердження ключовим імпульсом, але не дублює вивід тексту в послідовний порт, щоб уникнути подвоєння повідомлення на прикладному рівні. Якщо ж CRC не збігся, приймач обирає стратегію мовчання: передавач не отримує ACK, інтерпретує це як невдачу спроби й повторює передачу. Надійність цього кола в цілому задається параметрами `ackTimeoutMs`, `maxRetries` (зокрема режим нескінченних повторів при значенні -1) та паузою `turnaroundDelayMs`, яка відокремлює момент відправлення від моменту прослуховування, зменшуючи ймовірність хибного спрацювання на власний сигнал.

Прикладне програмне забезпечення

Прикладне програмне забезпечення в цій роботі виконує роль моста між людиною й апаратурою, і тому його реалізацію доцільно розглядати як послідовність узгоджених рішень про те, де зберігається стан, хто його змінює і як він доходить до графіка. Низькорівневий ввід-вивід із портом відокремлено від логіки розбору повідомлень, а та – від візуалізації. Такий поділ не є самоціллю, він полегшує супровід і зменшує ризик помилок при розширенні функціональності, наприклад при додаванні нових типів діагностичних рядків.

					ВКРБ-122.26.0003.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		52

Клас SerialReader

Клас SerialReader у файлі services/serial_reader.py інкапсулює роботу з serial.Serial у методі run(), який виконується в окремому потоці Qt. Читання з COM-порту потенційно блокуюче, і винесення його з GUI-потoku зберігає відчуття відгуку інтерфейсу. Усередині циклу читання використовується комбінація ser.in_waiting і короткого timeout, що дозволяє регулярно повертати керування й обробляти зупинку. Вихідні дані до пристрою ставляться в чергу через метод send, а запис на порт розбивається на фрагменти по 64 байти з невеликою паузою між фрагментами; інженерний сенс цього прийому полягає в тому, що довгі рядки команд або повідомлень менше б'ють по буферах USB-CDC і рідше призводять до втрат символів на стику драйвер–застосунок. Отримані з порту байти декодуються у рядок із політикою errors='ignore', після чого через сигнал data_received вони потрапляють у головне вікно. Таким чином, навіть якщо в потоці з'являться рідкісні пошкоджені UTF-8 послідовності, це не руйнує увесь сеанс. Нижче наведено реалізацію метода run():

Віджет SerialBlock

Кожен пристрій у головному вікні представлено віджетом SerialBlock (див. Рисунок 4.4), який поєднує вибір порту, індикатор стану, кнопку підключення та область журналу. Список портів оновлюється через get_available_ports() з serial_provider.py, тобто застосунок не вимагає ручного введення імені COM-порту й тим самим знижує кількість операційних помилок. Обмеження довжини журналу (MAX_LOG_CHARS) відображає практичний досвід: під час довгої передачі текстовий лог може рости швидше, ніж зручно читати, і обрізання старих фрагментів зберігає продуктивність UI.

					ВКРБ-122.26.0003.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		53

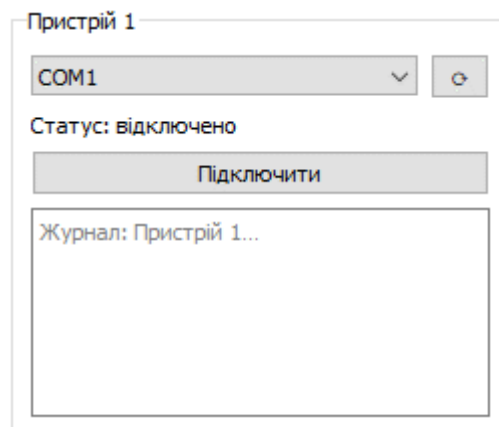


Рисунок 4.4 – Приклад віджета SerialBlock

Точка входу `main.py` навмисно залишена спрощеною. Вона створює `QApplication`, показує `MainWindow` і запускає цикл подій; додатковий виклик `push_license_notification()` слугує для відображення службового повідомлення про авторське право й не впливає на логіку протоколу. Уся предметна логіка зосереджена в модулях.

Клас `LogParser` виконує роль мовного перекладача між текстовим журналом прошивки й типізованими подіями Python. Його можна уявити як набір правил: якщо рядок починається з певного префікса, будується відповідний об'єкт події, який далі легше обробляти в одному місці, ніж розкидати `startswith` по всьому інтерфейсу. Нижче – ядро методу `parse_line`, де видно розгалуження за префіксами діагностичних рядків:

```
def parse_line(self, line: str):
    line = line.strip()
    if not line:
        return None
    if line.startswith("@S "):
        return self._parse_sensor(line)
    if line.startswith("@BIT "):
        return self._parse_bit_timing(line)
    if line == "@RX_START":
        return RxWindowEvent(active=True)
    if line.startswith("@RX_END "):
        return RxWindowEvent(active=False, result=line.split(maxsplit=1)[1])
```

```

if line.startswith("@TX_DONE "):
    return self._parse_tx_done(line)
if line.startswith("@CHUNK "):
    return self._parse_chunk(line)
if line.startswith("@CFG "):
    return self._parse_config(line)
if self.include_legacy_result_logs:
    return self._parse_legacy_line(line)
return None

```

Для підвищення наочності та спрощення аналізу механізму обробки службових повідомлень у системі доцільно представити узагальнену таблицю відповідності текстових префіксів типам подій. Такий підхід дозволяє компактно відобразити логіку перетворення вхідних діагностичних рядків у внутрішні структури даних застосунку, а також полегшує сприйняття матеріалу під час ознайомлення з архітектурою програмного забезпечення. У таблиці наведено основні префікси повідомлень, що надходять від мікроконтролера, та відповідні типи подій, які створюються модулем парсингу:

Таблиця 4.2 – відповідність текстових префіксів типам подій.

Префікс або умова	Результат
@S	SensorEvent (значення ADC; опційно threshold=, laser=)
@BIT	BitTimingEvent (H=, L=)
@RX_START	RxWindowEvent (active=True)
@RX_END	RxWindowEvent (active=False, result=...)
@TX_DONE	TxDoneEvent (bytes, ms, retries, result)
@CHUNK	ChunkEvent (seq, attempt, result)
@CFG	ConfigEvent (статус OK/ERR)

Також важливо зазначити, що нерозпізнані рядки не ламають парсер, вони просто не породжують подій для графіків, але можуть залишатися у зовнішньому журналі пристрою, що коректно відображає їхню другорядну роль.

ChartDataStore (services/chart_data_store.py) зосереджує накопичення історії: обмежені за розміром deque для сенсора, вікна прийому, бітового

таймінгу та оцінки пропускної здатності, а також лічильники результатів передачі. Обмеження `maxlen` тут слугує не лише економії пам'яті, а й тому, щоб графіки залишались читабельними: надто довга історія на екрані з малою роздільною здатністю перетворюється на суцільну пляму. Метод `update_config` підтримує узгодженість між тим, що користувач вважає еталонними тривалостями HIGH/LOW у формі, і тим, що порівнюється з фактичними вимірами з @BIT.

Головне вікно та графіки

`MainWindow` зшиває докупи всі описані шари. На момент підключення порту створюється потік, у який переноситься `SerialReader`, а сигнали з'єднуються з обробниками, що збирають неповні рядки в буфері й викликають `parse_line` та `record_event` лише для завершених рядків. Окремо варто згадати нюанс інтерфейсу: рядки @S та @BIT можуть бути приховані в логу пристрою, щоб не засмічувати вікно, але як події для графіків вони все одно обліковуються – це приклад того, як одна й та сама інформація може подаватися в різних варіаціях для різних цілей (спостереження очима оператора проти аналізу кривих). Передача користувацького тексту за замовчуванням адресується до першого пристрою, а команда зупинки розсилається на всі підключені блоки, що відповідає логіці зупинити все, що зараз активно слухає порт. Вкладка `ChartsTab` оновлює `SensorChart`, `TxResultChart` та `ThroughputChart` таймером з періодом 100 мс; цей інтервал є компромісом між плавністю анімації й навантаженням на CPU.

Експериментальна частина припускає наявність двох пристроїв ESP32-C3 з однаковою прошивкою, підключених до персонального комп'ютера (часто через два USB-порти). Застосунок дозволяє вести два журнали паралельно, що зручно, коли один канал використовується для передачі, а другий – для спостереження сигналу на приймальній стороні або для паралельного моніторингу. Перед серією вимірювань доцільно зафіксувати пресет швидкості (висока, середня та низька) або власний режим із явно записаними `high_ms` та

low_ms, а також зовнішні умови: відстань між оптичними модулями, рівень фонового освітлення, наявність або відсутність дифузора.

4.2 Захист розробленого програмного забезпечення

Захист ПЗ для систем на Arduino-споріднених платформах ускладнюється відкритістю екосистеми й можливістю фізичного доступу до UART, SWD/JTAG або оновлення образу зовнішнім інструментом. Для LaserLink найближчі до реалізації пункти: зосереджена логіка протоколу й конфігурації; явна валідація @CFG; обмеження len і розміру payload; команда @STOP; фіксація залежностей Python у requirements.txt для відтворюваного середовища. Шифрування USB-трафіку, підпис прошивки та апаратний захист від запису залишаються резервом, а не поточним кодом. Розглянемо більш детально доступні методи:

Ізоляція блокуючих операцій

Обмін із COM-портом виконується в окремому потоці (SerialReader), у головний потік GUI дані потрапляють через сигнали Qt, що знижує ризик зависання вікна й примусового завершення процесу в неконтрольований момент.

Стійкість до пошкоджених байтів

Декодування вхідного потоку з errors='ignore' запобігає аварійному завершенню через некоректні UTF-послідовності; помилки відкриття порту й запису перехоплюються й передаються через error_occurred (контрольована деградація).

Обмеження навантаження на пристрій і ОС

Відправлення довгих рядків дробиться на WRITE_CHUNK_SIZE із паузою WRITE_PAUSE_MS; для serial.Serial задано timeout та write_timeout, що обмежує тривалість блокування.

Обмеження пам'яті для графіків

ChartDataStore використовує deque з обмеженою довжиною – захист доступності застосунку при тривалих сеансах.

					ВКРБ-122.26.0003.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		57

Парсинг телеметрії

У LogParser окремі гілки захищені від некоректних числових полів обробкою винятків, щоб один пошкоджений рядок не порушував оновлення графіків.

Правовий супровід

Виклик `push_license_notification()` у `main.py` інформує про ліцензійні умови сторонніх компонентів – це не криптографія, але відповідає практиці прозорого використання стороннього ПЗ.

Узгоджений захист прошивки та оптичного каналу

Поведінка застосунку спирається на передбачувані відповіді пристрою; тому механізми прошивки є частиною загального захисту програмного комплексу. CRC-8 у `protocol.h` виявляє випадкові спотворення в оптиці; у `receiveFrame` перевірка `frame.len` до читання `payload` запобігає переповненню буфера; поєднання АСК, таймаутів і `maxRetries` забезпечує надійність без нескінченних циклів; облік `lastReceivedSeq` усуває подвоєння даних після втрати АСК. Детальний розбір CRC і прийому кадру наведено в підрозділі 4.1; тут достатньо зафіксувати зв'язок захист ПЗ комплексу із цими перевітками.

Навіть при коректній роботі CRC і повторів залишається класична проблема транспортних протоколів: підтвердження могло бути втрачене на шляху назад, і тоді передавач повторить той самий кадр з тим самим seq. Змінна `lastReceivedSeq` дозволяє приймачу відрізнити новий шматок від повтору; для повтору знову відправляється АСК без дублювання корисних даних.

5 ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

На Рисунку 5.1 зображено розроблене у бакалаврської дипломної роботі програмне забезпечення системи LaserLink. З рисунку видно розподіл на такі функціональні зони:

- Навігація між режимами роботи – горизонтальні вкладки «Керування» та «Діаграми» (QTabWidget): на першій зосереджено налаштування й обмін даними, на другій – графіки телеметрії.

- Зона введення та керування передачею – багаторядкове поле для тексту повідомлення з підказкою «Введіть текст...», кнопки «Надіслати» та «Зупинити» (з підказкою про зупинку активної передачі); це основні функціональні кнопки прикладного рівня.

- Зона роботи з пристроями – два згруповані блоки «Пристрій 1» і «Пристрій 2» (SerialBlock): для кожного – вибір COM-порту зі списку, кнопка оновлення списку портів (піктограма ) , індикатор статусу підключення, кнопка «Підключити» та окремий журнал обміну з відповідним портом.

- Зона налаштувань пристрою – група «Налаштування пристрою» з полями пресету швидкості (висока / середня / низька / власна), тривалостей HIGH/LOW, порогу датчика, потужності лазера, параметрів ключа й таймаутів, режиму повторів тощо; значення з цієї форми передаються на МК у вигляді команди @CFG.

- Зона виведення результату – під полем керування розташоване текстове поле лише для читання з підказкою «Отримані дані...», куди збирається потік даних із активного з'єднання; додатково повний діагностичний вивід видно в журналах кожного пристрою.

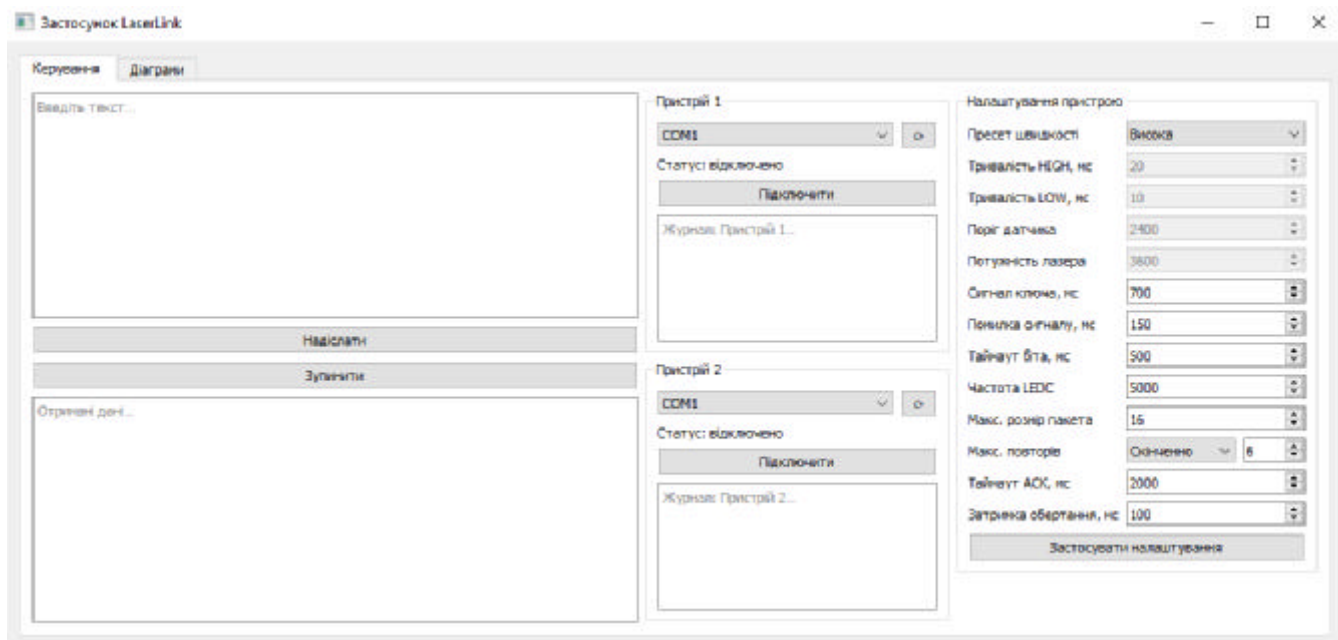


Рисунок 5.1 – Головне вікно ПЗ

Розроблена програма має простий і зрозумілий інтерфейс, який можна освоїти без довгого навчання, оскільки елементи керування підписані українською мовою й згруповані за змістом. Якщо застосунок не повідомляє про помилку підключення до порту й журнал оновлюється після натискання «Підключити», робочий цикл «налаштував – надіслав – переглянув журнал» можна вважати доступним для практичного використання. Якщо ж з'являється повідомлення про збій або в журналі пристрою фіксується @CFG ERR, слід виправити параметри згідно з текстом відповіді прошивки й повторити спробу – це відповідає загальній ідеї слідувати підказкам програми та пристрою.

Після підключення COM-портів та застосування налаштувань (Apply Settings) і надсилання тексту (або інші команди з вкладки керування), телеметрія з прошивки розбирається парсером і накопичується у вкладці «Діаграми» з оновленням приблизно кожні 100 мс відображаються три графіки pyplot (див. Рисунок 5.2):

- «Сигнали RX датчика / TX лазера» – часові ряди аналогового сигналу приймача, потужності лазера передавача та пунктиром – поріг детекції

- «Частка успіху / збоїв ТХ» – стовпчикова діаграма лічильників підтверджень (ACK), помилок CRC, таймаутів і збоїв
- «Пропускна здатність ТХ» – залежність швидкості передачі (байт/с) від номера передачі.



Рисунок 5.2 – Вкладка «Діаграми»

На Рисунку 5.3 зображено авторські дані розробленого програмного забезпечення:

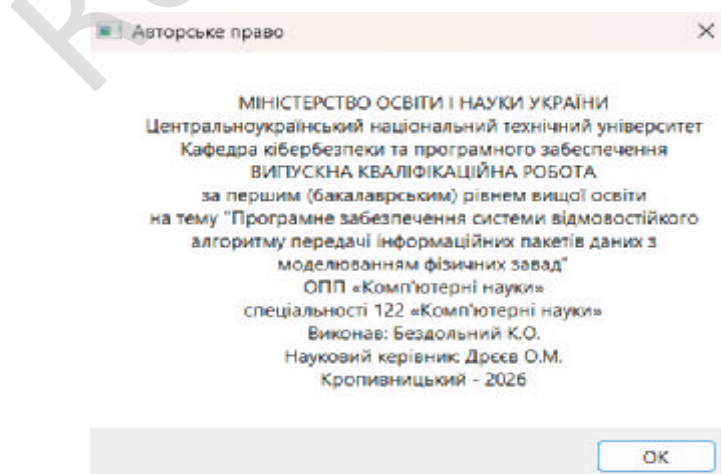


Рисунок 5.3 – Авторське право

6 ОСНОВНІ ВИСНОВКИ

У межах даної роботи було розроблено апаратно-програмну систему LaserLink, яка реалізує передачу текстових даних між двома мікроконтролерними пристроями через оптичний (лазерний) канал зв'язку. У процесі виконання роботи було послідовно вирішено задачі від аналізу принципів оптичної передачі даних до створення повноцінного прототипу з можливістю керування, діагностики та дослідження.

Розроблена система поєднує кілька рівнів: настільний застосунок, мікроконтролерну логіку та фізичний оптичний канал. Настільна частина забезпечує зручний інтерфейс користувача, налаштування параметрів і візуалізацію процесів. Мікроконтролери на базі ESP32-C3 реалізують протокол передачі, включаючи формування кадрів, обчислення CRC, обробку підтверджень і повторів. Фізичний рівень представлений лазерним випромінювачем і фотосенсором, які забезпечують безпосередню передачу та прийом сигналу.

У ході роботи було обґрунтовано вибір усіх ключових компонентів системи. Використання ESP32-C3 дозволило поєднати достатню продуктивність із простотою розробки. Фотодіод BPW34 разом із підсилювачем MCP602 забезпечили надійний прийом слабкого оптичного сигналу. Обрані програмні засоби – C++ з Arduino framework та PlatformIO для прошивки, а також Python із PyQt для настільного застосунку – дозволили ефективно реалізувати як низькорівневу, так і прикладну частину системи.

Особливу увагу приділено розробці протоколу передачі даних, який включає структуру кадрів, контроль цілісності за допомогою CRC, механізми підтвердження прийому (ACK) і повторної передачі. Це дозволило забезпечити базову надійність зв'язку навіть за наявності шумів і нестабільних умов оптичного каналу.

					ВКРБ-122.26.0003.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		62

Розроблені структурна, функціональна схеми та діаграма взаємодії процесів дозволили формалізувати архітектуру системи та наочно представити логіку її роботи. Це спрощує розуміння принципів функціонування LaserLink і може бути використано як основа для подальшого розвитку або навчальних цілей.

У підсумку було створено гнучкий і доступний прототип системи оптичного зв'язку, який може слугувати як для демонстрації принципів передачі даних, так і для експериментального дослідження впливу різних параметрів сигналу. Розробка також придатна для вивчення побудови протоколів обміну на практиці. Таким чином, поставлена мета роботи була досягнута, а отримані результати мають як прикладну, так і навчальну цінність.

К6ПЗ-2026

					ВКРБ-122.26.0003.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		63

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Espressif Systems. *ESP32-C3 Datasheet*. – [Електронний ресурс]: <https://www.espressif.com>
2. Arduino. *Arduino Documentation*. – Режим доступу: <https://docs.arduino.cc>
3. PlatformIO. *PlatformIO Documentation*. – [Електронний ресурс]: <https://docs.platformio.org>
4. What is Infrared Data Association. – [Електронний ресурс]: <https://www.electronicsforu.com/technology-trends/learn-electronics/infrared-data-association-irda>
5. VLC vs LiFi: Key Differences and Comparison. – [Електронний ресурс]: <https://www.rfwireless-world.com/terminology/vlc-vs-lifi-comparison>
6. Understanding Protocol, Working, and Applications. – [Електронний ресурс]: <https://www.rfwireless-world.com/tutorials/lifi-tutorial-protocol-working-applications>
7. Microchip Technology. *MCP602 Operational Amplifier Datasheet*. – [Електронний ресурс]: <https://mikroshop.ch/pdf/mcp602.pdf>
8. Vishay Semiconductors. *BPW34 Photodiode Datasheet*. – [Електронний ресурс]: <https://www.vishay.com/docs/81521/bpw34.pdf>
9. PySerial Documentation. – [Електронний ресурс]: <https://pyserial.readthedocs.io>
10. PyQt5 Documentation. – [Електронний ресурс]: <https://doc.qt.io/qtforpython/>
11. PyQtGraph Documentation. – [Електронний ресурс]: <https://www.pyqtgraph.org>
12. NumPy Documentation. – [Електронний ресурс]: <https://numpy.org/doc/>

13. Tanenbaum A. S., Wetherall D. J. *Computer Networks*. – 5th ed. – Pearson, 2011.
14. Stallings W. *Data and Computer Communications*. – 10th ed. – Pearson, 2013.
15. Kahn J. M., Barry J. R. *Wireless Infrared Communications*. – Proceedings of the IEEE, 1997.
16. Komine T., Nakagawa M. *Fundamental Analysis for Visible-Light Communication System using LED Lights*. – IEEE Transactions, 2004.
17. Sklar B. *Digital Communications: Fundamentals and Applications*. – Prentice Hall, 2001.
18. A Look Inside ADAS Modules. URL: <https://amkor.com/semiconductor-story/alook>.
19. AV and ADAS Sensors. URL <https://community.sw.siemens.com/s/article/AVand-ADAS-Sensors> (accessed 2023 June 1)
20. Саліхов М. М. Arduino – перспективний інструмент протитопування у робототехніці // Концептуальні шляхи розвитку науки та освіти: матеріали VIII Міжнародної науково-практичної конференції м. Львів, 9-10 червня 2023 року.–Львів: Львівський науковий форум, 2023. – с. 73-79.
21. Bräunl T. *Embedded Robotics: From Mobile Robots to Autonomous Vehicles with RaspberryPi and Arduino* Springer Nature, 2022. 519 p.
22. Лакоза; КПП ім. Ігоря Сікорського, 2021. — 80 с.
23. Imad M. et al. Navigation system for autonomous vehicle: A survey. JCSTS., vol. 2, no. 2, pp. 20–35, 2020.
24. Смірнов О.А., Гнатюк С.О., Кавун С.В., Терейковський І.А., Жмурко Т.О., Смірнов С.А., Коваленко А.С. *Основи безпеки в комп'ютерних мережах. Навчальний посібник* – Кропивницький: вид. Лисенко В.Ф. 2018. – 177 с.

25. HTML [Електронний ресурс] – Режим доступу до ресурсу:
<https://developer.mozilla.org/uk/docs/Web/HTML>
26. Aguilo I. et al. Artificial Intelligence Research and Development Hardcover by IOS Press. 2003. 500 p.
27. Смірнов О.А., Євсєєв С.П., Жукарев В.Ю., Король О.Г., Сорокін В.Є., Мелешко Є.В. Технології і стандарти комп'ютерних мереж. Навчальний посібник. – Кіровоград: КНТУ 2012. – 454 с
28. Смірнов О.А., Стасєв Ю.В. Бараннік В.В. Захист інформації в автоматизованих системах управління. Навчальний посібник – Харків: ХУПС, 2015. – 264 с.
29. Lim, B.S. et al. Autonomous Vehicle Ultrasonic Sensor Vulnerability and Impact Assessment. In Proceedings of the IEEE World Forum on Internet of Things (WFIoT), Singapore, 5–8 February 2018; pp. 231–236.
30. Смірнов О.А., Котелянець В.В. Стійкі до колізій стохастичні моделі функціонування безпроводових сенсорних мереж. Вісник інженерної академії України, №3, с. 145-152, 2018
31. Смірнов О.А., Коноплицька-Слободенюк О.К., Смірнов С.А., Буравченко К.О., Смірнова Т.В. Поліщук Л.І. Проектування комп'ютерних систем та мереж. Навчальний посібник – Кропивницький: вид. Лисенко В.Ф. 2019. – 264 с.
32. Smirnov, O., Kuznetsov, A., Kiian, A., Zamula, A., Rudenko, S., Hryhorenko, V., «Variance Analysis of Networks Traffic for Intrusion Detection in Smart Grids», *2019 IEEE 6th International Conference On Energy Smart Systems (2019 IEEE ESS)*, Kyiv, Ukraine April 17-19, 2019 P. 353-358.
33. Smirnov, O., Kuznetsov, A., Kavun, S., Babenko, B., Nakisko, O., Kuznetsova, K., «Malware Correlation Monitoring in Computer Networks of Promising Smart Grids», *2019 IEEE 6th International Conference On Energy Smart Systems (2019 IEEE ESS)*, Kyiv, Ukraine April 17-19, 2019 P. 347-352.

34. Смірнов О.А., Коноплицька-Слободенюк О.К., Смірнов С.А., Буравченко К.О., Смірнова Т.В., Поліщук Л.І. Інформаційна безпека в комп'ютерних мережах. Навчальний посібник – Кропивницький: вид. Лисенко В.Ф. 2020. – 294 с.

35. О.А. Смірнов, П.С. Усік, «Дослідження перспектив використання технологічних рішень в мережах 5G» у *Кибербезпека та інформаційні технології: монографія*. – Х. : ТОВ «ДІСА ПЛЮС», 2020.С. 122-135.

36. Смірнов О.А., Дреєва Г.М., Дреєв О.М., Смірнова Т.В. «Фрактальний аналіз генератора самоподібного трафіку на основі ланцюга Маркова». *Центральноукраїнський науковий вісник. Технічні науки*. № 2(33). с. 161-172, 2019.

37. Смірнов О.А., Коноплицька-Слободенюк О.К., Смірнов С.А., Буравченко К.О., Смірнова Т.В. Поліщук Л.І. Проектування комп'ютерних систем та мереж. Навчальний посібник – Кропивницький: вид. Лисенко В.Ф. 2019. – 264 с.

38. Smirnov, O., Kuznetsov, A., Kuznetsova., K. Synthesis of Discrete Signals with Improved Correlation Properties. Монографія: In.: ISCI'2019: Information Security in Critical Infrastructures. Collective monograph. Edited by Ivan D. Gorbenko and Alexandr A. Kuznetsov, ASC Academic Publishing, USA, 2019, pp. 281-299. – ISBN: 978-0-9989826-8-7 (Hardback), ISBN: 978-0-9989826-9-4 (Ebook).

39. Смірнов О.А., Дреєва Г.М. Метод генерування фрактального трафіку за допомогою моделі генератора на графі. Монографія: Інформаційна безпека та інформаційні технології : монографія / за заг. ред. В. С. Пономаренка. – Х. : Вид. Рожко С.Г. 2019. С. 123-139

40. Kuznetsov, O., Kryvinska, N., Ilchenko, O., Smirnova, T., Ulianovska, Y. «Comparative Analysis of Cryptocurrency Trading Platforms Using the Analytic Hierarchy Process». *CEUR Workshop Proceedings*, 2023, 3628, pp. 106-115.

41. Al-Mudhafar Aqeel, A.M., Smirnova, T., Buravchenko, K., Smirnov, O. «The method of assessing and improving the user experience of subscribers in

software-configured networks based on the use of machine learning». *Advanced Information Systems*, 2023, 7(2), pp. 49-56.

42. Smirnov, O., Sydorenko, V., Aleksander, M., Zhyharevych, O., Yenchев, S. «Simulation of the cloud IoT-based monitoring system for critical infrastructures». *CEUR Workshop Proceedings*, Volume 3530, 2023, pp. 256-265.

43. Smirnov, O., Odarchenko, R., Smirnova, T., Bondar, S., Volosheniuk, D. «Optimal Structure Construction of Private 5G Network for the Needs of Enterprises». *Lecture Notes on Data Engineering and Communications Technologies*, 2023, 178, pp. 208–223.

44. Аль-Мудхафар Акіл Абдулхуссейн М., Смірнова Т.В., Буравченко К.О., Смірнов О.А. «Метод оцінки та підвищення користувальницького досвіду абонентів в програмно-конфігурованих мережах на основі використання машинного навчання». *Сучасні інформаційні системи*, 2023, том 7, № 2, С. 49-56.

45. Smirnova, T., Gnatyuk, S., Yudin, O., Sydorenko, V., Polozhentsev, A., «The Model for Calculating the Quantitative Criteria for Assessing the Security Level of Information and Telecommunication Systems». *CEUR Workshop Proceedings* Volume 3156, 2022, Pages 390-399.

46. Смірнова Т.В., Гнатюк С.О., Сидоренко В.М., Юдін О.Ю., Сидоренко С.Ю., «Модель визначення критичності галузевих інформаційно-телекомунікаційних систем». *Проблеми інформатизації та управління*, № 2(70). 2022. С. 28-37.

47. Смірнов О.А., Смірнова Т.В., Якименко Н.М., Смірнов С.А., Поліщук Л.І., «Дослідження стійкості до диференціального криптоаналізу запропонованої функції гешування удосконаленого модуля криптографічного захисту в інформаційно-комунікаційних системах» *Системи управління, навігації та зв'язку*, 2022, № 3(69). С. 93-98.

48. Smirnov, O., Markovets, O. Vovk, N., Turchyn, Y., «Model of informational support for social network administrators' content creation». *CEUR Workshop Proceedings* Volume 2616, 2020, Pages 125-136.

					ВКРБ-122.26.0003.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		68

49. Smirnov, O., Drieieva, H., Drieiev, O., Polishchuk, Y., Brzhanov, R., Aleksander, M. «Method of fractal traffic generation by a model of generator on the graph». *CEUR Workshop Proceedings* Volume 2616, 2020, Pages 366-379.

50. Smirnov, O., Drieieva, H., Drieiev, O., Simakhin, V., Bondar, S., Odarchenko, R. «Managing multifractal properties of the binary sequence generated with the Markov chains», *CEUR Workshop Proceedings* Volume 2608, 2020, Pages 633-645.

51. Smirnov O. Kuznetsov A., Zaichenko Yu., Pastukhov M., Oleshko O., Kuznetsova K., «Formation of Discrete Signals with Special Correlation Properties». *International Conference on Information and Telecommunication Technologies and Radio Electronics, UkrMiCo 2019*; Odessa; Ukraine; 9-13 September 2019. P.22-28.

К6ПЗ-2026

					ВКРБ-122.26.0003.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		69