МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Відокремлений структурний підрозділ
«Тернопільський фаховий коледж
Тернопільського національного технічного університету імені Івана Пулюя»

Відділення телекомунікацій та електронних систем

Циклова комісія комп'ютерних наук

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи

фахового молодшого бакалавра

на тему: Розробка програмного забезпечення для мультиканального виконавчого пристрою високої надійності на базі мікроконтролера МК STM32

Виконав: студент IV курсу, групи КН-421
спеціальності: 122 «Комп'ютерні науки»
Ярослав ГРИНЧИШИН

Керівник Галина МАРЦІЯШ

Рецензент Андрій ЛУЦКІВ
(ім'я та прізвище)

м. Тернопіль – 2026

ВІДОКРЕМЛЕНИЙ СТРУКТУРНИЙ ПІДРОЗДІЛ
«ТЕРНОПІЛЬСЬКИЙ ФАХОВИЙ КОЛЕДЖ
ТЕРНОПІЛЬСЬКОГО НАЦІОНАЛЬНОГО ТЕХНІЧНОГО УНІВЕРСИТЕТУ
ІМЕНІ ІВАНА ПУЛЮЯ»

Відділення телекомунікацій та електронних систем
Циклова комісія комп'ютерних наук
Освітньо-професійний ступінь «фаховий молодший бакалавр»
Спеціальність 122 Комп'ютерні науки
Галузь знань 12 Інформаційні технології

ЗАТВЕРДЖУЮ
Голова циклової комісії комп'ютерних наук
_____ Галина МАРЦІЯШ
« 02 » березня 2026 року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТЦІ

Гринчишин Ярослав Юрійович
(прізвище, ім'я, по батькові)

1. Тема роботи: Розробка програмного забезпечення для мультиканального виконавчого пристрою високої надійності на базі мікроконтролера МК STM32

керівник роботи Марціяш Галина Ярославівна,
(прізвище, ім'я, по батькові)
затверджені наказом вищого навчального закладу № 4/9-132 від 27.02.2026 р.
2. Строк подання студентом роботи: 19.06.2026 р.
3. Вихідні дані до роботи: технічне завдання на розробку програмного забезпечення, мови програмування: C, C++; фреймворки: Qt; бібліотека HAL, стандарти IEEE 29148-2018, IEEE 29119, ДСТУ 8302:2015.

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

1 Загальний розділ

1.1 Аналітичний огляд існуючих рішень

1.2 Технічне завдання

1.2.1 Найменування та область застосування

1.2.2 Призначення розробки

1.2.3 Вимоги до програмного забезпечення

1.2.4 Вимоги до програмної документації

1.2.5 Техніко-економічні показники

1.2.6 Стадії та етапи розробки

1.2.7 Порядок контролю та прийому

Виконання роботи із розробкою програмного забезпечення для мультиканального виконавчого пристрою високої надійності на базі МК STM32.

5. Перелік графічного матеріалу:

1. Структурна схема програмно-апаратної системи
2. Схема взаємодії програмних модулів
3. Схема структури протоколу обміну даними
4. Блок-схема алгоритму діагностики периферійних вузлів
5. Таблиця техніко-економічних показників

6. Консультанти розділів роботи

Розділ	**Ім'я, прізвище та посада консультанта**	**Підпис, дата**	
		завдання видав	**завдання прийняв**
Економічний розділ	Любов КАЛУШКА		
Охорона праці, техніка безпеки та екологічні вимоги	Генадій ГОРЯЧЕК		

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Отримання і аналіз технічного завдання	20.03.2026	
2	Збір і узагальнення інформації	01.05.2026	
3	Написання першого розділу	15.05.2026	
4	Розробка технічного та робочого проєкту	29.05.2026	
5	Написання спеціального розділу	05.06.2026	
6	Розрахунок економічної частини	08.06.2026	
7	Написання розділу охорони праці	09.06.2026	
8	Виконання графічної частини	10.06.2026	
9	Оформлення пояснювальної записки	11.06.2026	
10	Погодження нормоконтролю	12.06.2026	
11	Попередній захист кваліфікаційної роботи	11.06.2026	
12	Захист кваліфікаційної роботи	22.06.2026	

7. Дата видачі завдання: 05 березня 2026 р.

Студент _____ Ярослав ГРИНЧИШИН
(підпис)

Керівник роботи _____ Галина МАРЦІЯШ
(підпис)

ЗМІСТ

АНОТАЦІЯ

Тема кваліфікаційної роботи: «Розробка програмного забезпечення для мультиканального виконавчого пристрою високої надійності на базі мікроконтролера МК STM32».

Метою кваліфікаційної роботи є розробка програмного забезпечення для мультиканального виконавчого пристрою на базі STM32, що забезпечує надійну обробку та передавання даних, керування виконавчими каналами, діагностику периферійних вузлів і стабільну роботу системи в режимі реального часу.

Пояснювальна записка складається з п’яти розділів.

У загальній частині наведено аналітичний огляд існуючих рішень у сфері вбудованих систем та мультиканальних виконавчих пристроїв, виконано аналіз технічного завдання, визначено основні вимоги до програмного забезпечення та апаратної платформи.

У другому розділі представлено процес розробки програмного забезпечення, описано структуру системи, алгоритми обробки та передавання даних, взаємодію програмних модулів, а також особливості роботи з периферією мікроконтролера STM32. Окрему увагу приділено організації протоколу обміну даними, контролю цілісності пакетів та обробці помилкових ситуацій. Наведено результати тестування та налагодження програмного забезпечення.

У спеціальній частині описано порядок налаштування, інсталяції та експлуатації програмного забезпечення. В економічному розділі виконано розрахунок вартості розробки та оцінку економічної ефективності. У п’ятому розділі розглянуто питання охорони праці та техніки безпеки.

Обсяг пояснювальної записки становить 112 аркушів.

Графічна частина кваліфікаційної роботи складається з п’яти аркушів формату А1: структурної схеми програмно-апаратної системи, схеми взаємодії програмних модулів, схеми структури протоколу обміну даними, блок-схеми алгоритму діагностики периферійних вузлів та таблиці техніко-економічних показників.

ANNOTATION

The topic of the qualification work is “Development of software for a multichannel high-reliability actuator device based on the STM32 MC microcontroller”.

The aim of the qualification work is to develop software for a multichannel actuator device based on STM32, which provides reliable data processing and transmission, control of actuator channels, diagnostics of peripheral nodes, and stable real-time system operation.

The explanatory note consists of five sections.

The general section presents an analytical review of existing solutions in the field of embedded systems and multichannel actuator devices, analyzes the technical specification, and defines the main requirements for the software and hardware platform.

The second section presents the software development process, describes the system structure, data processing and transmission algorithms, interaction between software modules, and features of working with STM32 microcontroller peripherals. Special attention is paid to the organization of the data exchange protocol, packet integrity control, and handling of error situations. The results of software testing and debugging are presented.

The special section describes the procedure for software configuration, installation, and operation. The economic section contains the calculation of the software development cost and the assessment of economic efficiency. The fifth section considers occupational safety and safety engineering issues.

The explanatory note consists of 112 pages.

The graphical part of the qualification work consists of five A1-format sheets: the structural diagram of the hardware and software system, the diagram of software module interaction, the diagram of the data exchange protocol structure, the flowchart of the peripheral node diagnostics algorithm, and the table of technical and economic indicators.

ВСТУП

Сучасні вбудовані системи широко застосовуються у промисловості, автоматизованих системах керування, телекомунікаціях та спеціалізованому обладнанні. Одним із важливих напрямків розвитку таких систем є створення багатоканальних виконавчих пристроїв, здатних забезпечувати стабільну та надійну роботу в умовах підвищених вимог до швидкодії, відмовостійкості та точності керування.

Актуальність теми полягає у необхідності розробки програмного забезпечення для мультиканальних виконавчих систем, які забезпечують одночасне керування декількома каналами з мінімальними затримками та високою стабільністю роботи. Особливу роль у таких системах відіграє програмна реалізація механізмів обробки даних, взаємодії між периферійними модулями та забезпечення безперервної роботи мікроконтролерної системи.

Метою кваліфікаційної роботи є розробка програмного забезпечення для мультиканального виконавчого пристрою високої надійності на базі мікроконтролера STM32. Для досягнення поставленої мети необхідно виконати аналіз існуючих рішень, розробити структуру програмного забезпечення, реалізувати механізми обміну даними між модулями системи та провести тестування створеного програмного забезпечення.

Об’єктом дослідження є процес функціонування мультиканальних виконавчих пристроїв на базі мікроконтролерних платформ.

Предметом дослідження є програмне забезпечення керування мультиканальним виконавчим пристроєм та методи забезпечення його надійної роботи.

Під час розробки використовуються засоби STM32CubeIDE, бібліотеки HAL, механізми UART та DMA для організації обміну даними між компонентами системи. Практичне значення роботи полягає у створенні програмного забезпечення, яке може бути використане у складі спеціалізованих систем автоматизації та керування.

1 ЗАГАЛЬНИЙ РОЗДІЛ

1.1 Аналітичний огляд існуючих рішень

У сучасних умовах розвитку інформаційних технологій та автоматизованих систем керування особливого значення набувають програмно-апаратні комплекси, призначені для централізованого керування виконавчими пристроями та збору даних з периферійних модулів. Такі системи широко застосовуються у промисловій автоматизації, системах диспетчеризації, технологічному обладнанні, системах моніторингу та керування розподіленими об’єктами.

Одним із найбільш поширених підходів до побудови подібних систем є архітектура з головним пристроєм та периферійними вузлами, у якій центральний керуючий пристрій виконує функції координації роботи периферійних модулів, збору інформації та відображення поточного стану системи. Даний підхід дозволяє забезпечити централізоване керування великою кількістю периферійних пристроїв, спростити обмін даними та підвищити надійність функціонування системи в цілому.

Для реалізації подібних систем широко використовуються сучасні мікроконтролери сімейства STM32, які характеризуються високою продуктивністю, наявністю апаратних засобів обробки даних, підтримкою DMA, великою кількістю периферійних інтерфейсів та можливістю роботи в режимі реального часу. Особливе місце займають мікроконтролери серії STM32H7, які містять високопродуктивне ядро ARM Cortex-M7 та дозволяють реалізовувати складні алгоритми обробки інформації, багатоканальний обмін даними та засоби забезпечення високої надійності роботи системи, що робить їх придатними для побудови вбудованих систем реального часу [1, 2].

Під час розробки вбудованих систем керування одним із ключових завдань є організація стабільного та надійного обміну даними між пристроями. Для цього широко використовуються інтерфейси UART, SPI, I2C, CAN та RS-485. Серед наведених рішень інтерфейс RS-485 є одним із найбільш поширених у

промислових системах керування завдяки можливості передачі даних на значні відстані, високій завадостійкості та підтримці багатовузлової топології.

Інтерфейс RS-485 часто використовується у системах автоматизації, промислових контролерах, системах збору телеметрії та багатоканальних пристроях керування. Передача даних за допомогою RS-485 зазвичай реалізується на базі UART із використанням спеціалізованих перетворювачів фізичного рівня. Такий підхід забезпечує простоту реалізації програмної частини та дозволяє використовувати стандартні апаратні модулі мікроконтролера.

У більшості сучасних систем керування застосовуються готові промислові протоколи обміну даними, зокрема Modbus RTU, CANOpen та інші. Проте використання універсальних протоколів не завжди є оптимальним рішенням для спеціалізованих вбудованих систем, оскільки вони можуть містити надлишковий функціонал, збільшувати затримки передачі даних та ускладнювати програмну реалізацію. У таких випадках доцільним є використання власного спеціалізованого протоколу передачі даних, який враховує особливості конкретної системи та дозволяє оптимізувати швидкодію й структуру обміну.

Однією з основних вимог до сучасних систем керування є забезпечення високої надійності передачі даних. Під час обміну інформацією можливе виникнення помилок, викликаних електромагнітними завадами, нестабільністю лінії зв’язку або порушенням часових характеристик передачі. Для підвищення достовірності передачі даних використовуються механізми контролю помилок, зокрема CRC-контроль, перевірка структури пакетів, контроль часових інтервалів між повідомленнями та повторна передача даних у разі виявлення помилки.

Додатковим фактором підвищення надійності є використання апаратних можливостей сучасних мікроконтролерів, таких як DMA та система переривань. Технологія DMA дозволяє виконувати передачу даних між периферійними модулями та пам’яттю без безпосередньої участі центрального процесора, що зменшує навантаження на ядро мікроконтролера та підвищує стабільність роботи системи. Використання переривань дозволяє оперативно реагувати на події обміну даними та забезпечувати роботу системи в режимі реального часу.

Для реалізації багатоканальних систем керування важливим фактором є можливість підключення великої кількості периферійних модулів. У сучасних системах кількість периферійних пристроїв може становити десятки окремих вузлів, кожен з яких передає службову або технологічну інформацію центральному пристрою. У зв’язку з цим особливого значення набувають питання оптимізації структури пакетів даних, контролю навантаження на канал зв’язку та забезпечення стабільної роботи системи при одночасному обслуговуванні великої кількості пристроїв.

Важливою складовою сучасних систем керування є людино-машинний інтерфейс, який забезпечує взаємодію оператора із програмно-апаратним комплексом. Для реалізації локального інтерфейсу користувача у вбудованих системах широко використовуються TFT-дисплеї, які дозволяють відображати службову інформацію, поточний стан системи, параметри роботи та повідомлення про помилки.

У сучасних embedded-системах одним із найбільш поширених графічних контролерів є ILI9341, який використовується у TFT-дисплеях середнього розміру та підтримує кольорове графічне відображення інформації. Використання дисплеїв на базі ILI9341 дозволяє реалізовувати багаторівневі меню керування, системи моніторингу, екрани діагностики та інші елементи інтерфейсу користувача.

У багатоканальних системах керування особливе значення має можливість швидкого контролю стану великої кількості підключених вузлів. Для цього в інтерфейсі користувача можуть використовуватися окремі екрани діагностики, на яких відображається поточний стан кожного підключеного модуля, наявність зв’язку, службові параметри та повідомлення про несправності. Такий підхід дозволяє значно спростити процес контролю працездатності системи та прискорити виявлення помилок.

Одним із важливих елементів програмного забезпечення embedded-систем є організація режимів роботи пристрою. У сучасних системах керування можуть використовуватися різні режими функціонування залежно від призначення системи та сценарію її використання. До основних режимів належать режим одиничної

активації, режим групової активації, режим очікування та режим підготовки системи до роботи. Розділення функціоналу за режимами дозволяє спростити логіку керування та підвищити безпечність експлуатації системи.

Не менш важливим компонентом embedded-систем є протокол обміну даними між пристроями. Протокол визначає структуру пакетів, формат службової інформації, порядок передачі команд та обробки відповідей. Від правильності організації протоколу залежить швидкодія системи, стабільність передачі даних та можливість масштабування кількості підключених пристроїв.

У сучасних системах керування широко використовуються пакетні протоколи передачі даних, у яких кожне повідомлення містить службову частину та корисне навантаження. До службової інформації зазвичай належать поля заголовка пакета, адреса пристрою, тип повідомлення, номер пакета, розмір даних та контрольна сума. Використання нумерації пакетів дозволяє контролювати послідовність передачі даних та виявляти втрату окремих повідомлень.

Для забезпечення достовірності передачі даних у промислових системах широко застосовується CRC-контроль. Контрольна сума дозволяє виявляти помилки передачі, викликані завадами, пошкодженням даних або порушенням цілісності пакета. У випадку невідповідності CRC-перевірки пакет вважається некоректним та відхиляється програмним забезпеченням.

У багатовузлових системах важливим завданням є контроль наявності зв’язку між центральним пристроєм та периферійними модулями. Для цього можуть використовуватися спеціальні ping-повідомлення або службові пакети, які передаються через визначені часові інтервали. Відсутність відповіді від пристрою протягом заданого часу дозволяє виявити втрату зв’язку та перевести систему у відповідний діагностичний стан.

Значна увага під час розробки embedded-систем приділяється забезпеченню надійності функціонування програмного забезпечення. Надійність системи визначається здатністю зберігати працездатність під час виникнення помилок, нестабільного зв’язку або аварійних ситуацій.

Одним із найбільш поширених методів забезпечення надійності є

використання таймаутів зв’язку, які дозволяють контролювати час очікування відповіді від периферійних пристроїв. У випадку перевищення встановленого часу очікування система може виконувати повторну передачу повідомлення або переводити вузол у стан помилки.

Для спрощення діагностики та контролю стану системи програмне забезпечення може використовувати систему службових станів. Найбільш поширеними є стани готовності до роботи, підготовки системи, активного режиму функціонування та аварійного стану. Використання подібної системи дозволяє забезпечити чітке розділення логіки роботи програмного забезпечення та спрощує обробку помилкових ситуацій.

У системах багатоканального керування важливим є відображення поточного стану кожного підключеного вузла. Для цього можуть використовуватись окремі діагностичні стани, які сигналізують про готовність пристрою до роботи, процес підготовки, активний режим функціонування або виникнення аварійної ситуації. Розділення станів типу ready, charging, activated та fault дозволяє підвищити інформативність системи та забезпечити більш ефективний контроль її працездатності.

Важливим елементом систем забезпечення надійності є індикація помилок та службових повідомлень. Відображення інформації про стан вузлів, втрату зв’язку, помилки передачі даних або аварійні режими дозволяє оператору оперативно реагувати на несправності та контролювати працездатність системи в реальному часі.

Для тестування та налагодження вбудованих систем широко використовуються програмні засоби емуляції периферійних пристроїв. Це дозволяє виконувати перевірку працездатності програмного забезпечення ще до завершення розробки всіх апаратних компонентів системи. Особливо актуальним такий підхід є під час створення багатовузлових систем із великою кількістю підключених пристроїв, де одночасне використання фізичних модулів на ранніх етапах розробки може бути ускладненим.

Для реалізації програмних емуляторів та тестових середовищ часто

використовується фреймворк Qt, який забезпечує можливість створення багатоплатформних застосунків із графічним інтерфейсом та підтримкою послідовного обміну даними. Використання програмного емулятора дозволяє моделювати роботу периферійних пристроїв, генерувати тестові пакети даних, перевіряти коректність функціонування протоколу обміну, виконувати відлагодження алгоритмів обробки повідомлень та оцінювати стабільність роботи системи в різних режимах функціонування.

Застосування програмного емулятора під час розробки дозволяє суттєво скоротити час тестування, спростити пошук помилок та забезпечити незалежність розробки програмного забезпечення центрального пристрою від готовності периферійних модулів системи.

Під час проєктування сучасних embedded-систем важливим фактором також є забезпечення масштабованості програмного забезпечення. Архітектура системи повинна дозволяти підключення додаткових периферійних модулів без необхідності суттєвого перепроєктування програмної частини. Модульний підхід до розробки програмного забезпечення спрощує подальшу модернізацію системи, розширення функціональних можливостей та адаптацію програмного комплексу до нових умов експлуатації. Особливо важливою така властивість є для багатовузлових систем керування, у яких кількість підключених пристроїв може змінюватися залежно від конфігурації об’єкта автоматизації.

Проведений аналіз існуючих підходів до побудови багатоканальних embedded-систем показав, що сучасні програмно-апаратні комплекси повинні поєднувати високу швидкодію, надійність передачі даних, масштабованість, зручний інтерфейс користувача та розвинуті засоби діагностики. Це підтверджує актуальність розробки спеціалізованого програмного забезпечення для мультиканального виконавчого пристрою на базі STM32 із використанням власного протоколу обміну даними та засобів забезпечення високої надійності функціонування системи.

1.2 Технічне завдання

1.2.1 Найменування та область застосування

Найменування програмного виробу – програмне забезпечення мультиканального виконавчого пристрою високої надійності на базі мікроконтролера STM32.

Програмне забезпечення призначене для використання у складі вбудованих систем керування та моніторингу, які використовуються у промисловій автоматизації, системах диспетчеризації, керуванні периферійними модулями та інших розподілених програмно-апаратних комплексах.

Область застосування програмного забезпечення охоплює системи, у яких необхідно забезпечити централізоване керування великою кількістю підключених вузлів, стабільний обмін даними між пристроями, контроль стану периферійних модулів та відображення службової інформації у режимі реального часу.

Програмне забезпечення використовується у складі апаратної платформи на базі мікроконтролера STM32H750VBT6 та забезпечує обмін даними з периферійними пристроями за допомогою інтерфейсу RS-485 [1, 3].

1.2.2 Призначення розробки

Експлуатаційним призначенням програмного забезпечення є забезпечення централізованого керування підключеними периферійними пристроями, контроль їхнього стану, приймання та обробка службових повідомлень, а також відображення інформації про роботу системи на дисплеї користувача.

Функціональне призначення програмного забезпечення полягає у:

– реалізації обміну даними між центральним пристроєм та периферійними модулями;

– забезпеченні передачі команд та приймання відповідей;

– обробці пакетів даних;

– контролі цілісності повідомлень за допомогою CRC;
– контролі наявності зв’язку із периферійними вузлами;
– реалізації системи діагностичних станів;
– відображенні інформації на TFT-дисплеї;
– реалізації меню керування та режимів роботи системи;
– забезпеченні роботи системи у режимі реального часу.

Програмне забезпечення також повинно забезпечувати можливість тестування системи за допомогою програмного емулятора периферійних пристроїв, реалізованого із використанням фреймворку Qt [4, 5].

1.2.3 Вимоги до програмного забезпечення

Програмне забезпечення повинно забезпечувати:

– стабільний обмін даними між центральним пристроєм та периферійними вузлами;
– роботу з великою кількістю підключених периферійних модулів;
– передачу даних за допомогою інтерфейсу UART із використанням фізичного рівня RS-485 [3];
– підтримку власного протоколу обміну даними;
– контроль правильності пакетів за допомогою CRC [6];
– обробку службових та інформаційних повідомлень;
– контроль втрати зв’язку із периферійними пристроями;
– обробку помилок передачі даних;
– підтримку режимів одиничного виконання операцій, послідовного виконання групових операцій та режиму підготовки системи до роботи;
– відображення поточного стану системи на TFT-дисплеї [7];
– відображення діагностичної інформації про підключені вузли;
– індикацію помилок та аварійних режимів;
– підтримку службових станів ready, charging, activated та fault;

– можливість масштабування кількості підключених вузлів;
– роботу у режимі реального часу [8, 9].

Вхідними даними програмного забезпечення є:

– пакети даних, що надходять від периферійних пристроїв;
– службові повідомлення протоколу;
– відповіді на команди центрального пристрою;
– дані про поточний стан підключених вузлів;
– сигнали керування користувача через меню пристрою.

Вихідними даними є:

– команди для периферійних пристроїв;
– службові пакети протоколу;
– інформація для відображення на дисплеї;
– повідомлення про помилки;
– діагностична інформація про стан системи;
– сигнали індикації стану роботи системи.

Програмне забезпечення повинно виконувати:

– ініціалізацію периферійних модулів мікроконтролера;
– обробку приймання та передачі пакетів;
– перевірку CRC-контролю;
– формування пакетів даних;
– контроль таймаутів зв’язку;
– періодичну перевірку наявності зв’язку із вузлами;
– оновлення інформації на дисплеї;
– обробку режимів роботи системи;
– обробку аварійних ситуацій.

Програмне забезпечення повинно забезпечувати стійкість до помилок передачі даних та втрати зв’язку із периферійними пристроями.

Часові характеристики програмного забезпечення повинні забезпечувати обробку пакетів даних у режимі реального часу та своєчасне оновлення інформації

про стан системи**.**

Програмне забезпечення повинно забезпечувати обробку пакетів даних у режимі реального часу та своєчасне оновлення інформації про стан системи. Час реакції системи на втрату зв’язку або виникнення помилки повинен бути мінімальним та визначатися параметрами таймаутів програмного забезпечення.

Для забезпечення надійності роботи системи повинні бути реалізовані:

- CRC-контроль пакетів;
- контроль таймаутів зв’язку;
- перевірка структури пакетів;
- система діагностичних станів;
- індикація помилок;
- контроль втрати зв’язку;
- захист від некоректних пакетів;
- обробка аварійних ситуацій;
- використання DMA та переривань для обробки обміну даними [10, 11].

Програмне забезпечення повинно функціонувати у складі вбудованої системи на базі мікроконтролера STM32H750VBT6 [1, 2].

Для роботи програмного забезпечення необхідні:

- TFT-дисплей на базі контролера ILI9341 [7];
- інтерфейс RS-485 [3];
- периферійні пристрої, підключені до шини обміну даними;
- джерело живлення відповідно до параметрів апаратної платформи.

Розробка програмного забезпечення здійснюється у середовищі STM32CubeIDE із використанням бібліотеки HAL [11, 12].

1.2.4 Вимоги до програмної документації

Програмна документація повинна містити:

– пояснювальну записку;
– опис структури програмного забезпечення;
– опис алгоритмів роботи;
– опис протоколу обміну даними;
– інструкцію з експлуатації програмного комплексу;
– інструкцію з використання тестових наборів;
– інструкцію з тестування;
– опис режимів роботи системи.

Призначення основних програмних модулів, їхні вхідні та вихідні дані, особливості використання і взаємодія між ними повинні бути описані у пояснювальній записці та додатках. У тексті програмної документації необхідно наводити достатні пояснення для розуміння структури програмного забезпечення, алгоритмів роботи, протоколу обміну даними та режимів функціонування системи. Склад, структура та загальні вимоги до оформлення кваліфікаційної роботи визначаються методичними вказівками до виконання кваліфікаційної роботи для спеціальності 122 «Комп’ютерні науки» [13].

Оформлення пояснювальної записки та програмної документації повинно відповідати загальним вимогам до структури й оформлення звітів у сфері науки і техніки [14].

Оформлення бібліографічних посилань у переліку використаних джерел виконується відповідно до вимог ДСТУ 8302:2015 [15].

1.2.5 Техніко-економічні показники

Розробка програмного забезпечення для мультиканального виконавчого пристрою високої надійності на базі STM32 спрямована на підвищення ефективності керування периферійними вузлами, забезпечення надійного обміну даними та спрощення процесу контролю стану системи. Очікуваний ефект від упровадження розробки полягає у централізованому керуванні пристроями, зменшенні часу на діагностику, підвищенні стабільності роботи системи та

скороченні тривалості налагодження за рахунок використання програмного емулятора периферійних вузлів.

До основних техніко-економічних показників розробки можна віднести:

– витрати на оплату праці виконавців, залучених до розробки програмного забезпечення;

– нарахування на заробітну плату;

– витрати на електроенергію, спожиту під час розробки та тестування програмного продукту;

– витрати на амортизацію комп’ютерної техніки та налагоджувального обладнання;

– витрати, пов’язані з використанням апаратної платформи STM32 та периферійних модулів;

– накладні витрати, пов’язані з організацією процесу розробки;

– витрати часу на проєктування, програмування, налагодження і тестування системи.

Узагальнену таблицю техніко-економічних показників розробки наведено у графічній частині 2026.КВР.122.421.06.00.00 ТБ. У таблиці подано основні кількісні результати економічного обґрунтування, зокрема собівартість розробки, плановий прибуток, ціну програмного продукту, чисту теперішню вартість та термін окупності.

Використання мікроконтролера STM32H750VBT6 є технічно та економічно доцільним, оскільки він забезпечує достатню продуктивність для обробки даних, взаємодії з периферійними пристроями та реалізації алгоритмів контролю без застосування складніших і дорожчих апаратних рішень. Обрана платформа є раціональним варіантом для вбудованої системи керування, оскільки поєднує високу швидкодію, підтримку необхідних інтерфейсів зв’язку, компактність та можливість роботи в режимі реального часу. Застосування емулятора периферійних вузлів дає змогу виконувати перевірку програмної логіки без постійного використання повного апаратного комплексу, що зменшує витрати часу

на тестування та пошук помилок.

Кількісні економічні розрахунки, зокрема визначення витрат на оплату праці, електроенергію, амортизацію, накладні витрати, собівартість розробки, ціну програмного продукту, економічну ефективність і термін окупності, наводяться в економічному розділі кваліфікаційної роботи.

1.2.6 Стадії та етапи розробки

Розробку програмного забезпечення для мультиканального виконавчого пристрою високої надійності на базі STM32 доцільно виконувати поетапно, щоб послідовно пройти аналіз вимог, проєктування структури системи, реалізацію програмних модулів, тестування та підготовку програмної документації.

Основні стадії розробки:

1. Аналіз предметної області. На цьому етапі визначаються призначення програмного забезпечення, особливості роботи мультиканального виконавчого пристрою, вимоги до керування периферійними вузлами, обміну даними, діагностики та надійності функціонування системи.

2. Формування технічного завдання. Визначаються функціональні та нефункціональні вимоги до програмного забезпечення, склад програмної документації, техніко-економічні показники, порядок контролю та приймання роботи.

3. Проєктування структури програмного забезпечення. Визначається загальна організація програми, склад основних програмних модулів, розподіл функцій між ними, структура задач керування, обміну даними, обробки подій, індикації та діагностики.

4. Розробка протоколу обміну даними. Визначається структура інформаційних пакетів, призначення службових полів, порядок нумерації пакетів, перевірка цілісності даних, обробка відповідей периферійних вузлів та виявлення помилок зв’язку.

5. Реалізація алгоритмів керування периферійними вузлами. Розробляються алгоритми формування команд, обробки станів периферійних пристроїв, контролю виконання операцій, перевірки доступності вузлів та реакції системи на помилки.

6. Реалізація інтерфейсу користувача. Створюються елементи меню, екрани відображення поточного стану системи, режими керування, повідомлення про помилки та засоби перегляду діагностичної інформації.

7. Реалізація системи діагностики. Забезпечується збір, обробка та відображення інформації про стан периферійних вузлів, результати обміну даними, коди помилок і параметри, необхідні для контролю працездатності системи.

8. Створення програмного емулятора периферійних пристроїв. Розробляється емулятор, який імітує роботу периферійних вузлів та дає змогу перевіряти програмну логіку, обмін даними і реакцію системи на різні стани без постійного використання повного апаратного комплексу.

9. Тестування програмного забезпечення. Перевіряється робота основних сценаріїв: запуск системи, вибір режимів роботи, передача команд, отримання відповідей, контроль стану периферійних вузлів, відображення діагностичних даних та обробка помилкових ситуацій.

10. Налагодження та оптимізація роботи системи. Виконуються пошук і виправлення помилок, перевірка стабільності обміну даними, уточнення часових параметрів, оптимізація роботи програмних модулів та підвищення надійності функціонування системи.

11. Підготовка програмної документації. Описуються структура програмного забезпечення, основні алгоритми, протокол обміну даними, порядок роботи з програмою, результати тестування та інші матеріали, необхідні для супроводу розробки.

12. Контроль виконання та приймання роботи. Перевіряється відповідність реалізованого програмного забезпечення вимогам технічного завдання, працездатність основних функцій, коректність відображення стану системи та готовність роботи до демонстрації.

1.2.7 Порядок контролю та прийому

Контроль працездатності програмного забезпечення для мультиканального виконавчого пристрою високої надійності на базі STM32 здійснюється шляхом перевірки відповідності реалізованих функцій вимогам технічного завдання. Основною метою контролю є підтвердження правильності роботи програмних модулів, стабільності обміну даними між центральним пристроєм і периферійними вузлами, коректності відображення інформації на дисплеї та здатності системи реагувати на помилкові ситуації.

Під час тестування необхідно перевірити:

- коректність запуску програмного забезпечення після подачі живлення;
- ініціалізацію основних периферійних модулів мікроконтролера;
- правильність формування пакетів для обміну даними;
- відповідність структури пакетів розробленому протоколу;
- коректність обчислення та перевірки CRC-коду;
- приймання і обробку відповідей від периферійних вузлів;
- виявлення втрати зв’язку з периферійними пристроями;
- обробку пошкоджених або некоректно сформованих пакетів;
- роботу основних режимів функціонування системи;
- коректність реагування системи на команди користувача;
- відображення поточного стану системи на дисплеї;
- роботу системи діагностики та виведення діагностичних повідомлень;
- коректність обробки помилкових станів;
- роботу програмного емулятора периферійних пристроїв;
- стабільність роботи системи під час тривалого обміну даними.

Окрему увагу під час контролю необхідно приділити перевірці надійності обміну даними, оскільки від правильності приймання, передавання та перевірки пакетів залежить стабільність роботи всієї системи. Для цього доцільно виконати тестування як у штатних умовах, так і в ситуаціях, коли окремі периферійні вузли

не відповідають, передають некоректні дані або перебувають у помилковому стані. Такий підхід дає змогу перевірити не лише основну функціональність, а й поведінку програмного забезпечення у нестандартних умовах.

Приймання програмного забезпечення здійснюється після демонстрації основних функцій системи, виконання контрольних сценаріїв тестування та підтвердження відповідності реалізованої розробки вимогам технічного завдання. Програмне забезпечення вважається працездатним, якщо система стабільно запускається, забезпечує правильний обмін даними між центральним пристроєм і периферійними вузлами, коректно обробляє помилки зв’язку, відображає інформацію про стан системи та не втрачає працездатності під час тривалого тестування.

Контрольний приклад повинен охоплювати основні сценарії роботи програмного забезпечення. До таких сценаріїв належать запуск пристрою після подачі живлення, перехід до режиму діагностики, отримання відповідей від периферійних вузлів, відображення їхнього стану на дисплеї, виконання режиму заряджання, одиночної та групової активації, а також перевірка реакції системи на відсутність відповіді або надходження некоректних даних. Це дає змогу оцінити роботу програмного забезпечення як у штатних умовах, так і під час виникнення помилкових ситуацій.

Під час приймання також необхідно перевірити узгоджену роботу основних програмних модулів у складі єдиної системи. Для цього виконується перевірка взаємодії модуля обміну даними, алгоритмів обробки станів, інтерфейсу користувача, системи діагностики та програмного емулятора периферійних вузлів. Така перевірка дозволяє переконатися, що програмне забезпечення не лише виконує окремі функції, а й забезпечує стабільну роботу пристрою в повному циклі функціонування.

Результатом приймання є висновок про готовність програмного забезпечення до використання у складі мультиканального виконавчого пристрою. Це підтверджує готовність програмного забезпечення до подальшого використання та демонстрації під час захисту кваліфікаційної роботи.

2 РОЗРОБКА ТЕХНІЧНОГО ТА РОБОЧОГО ПРОЄКТУ

2.1 Постановка задачі на розробку програмного забезпечення

У межах кваліфікаційної роботи необхідно розробити програмне забезпечення для мультиканального виконавчого пристрою високої надійності на базі мікроконтролера STM32 [1, 2]. Розроблювана система призначена для керування групою виконавчих вузлів, контролю їхнього стану, обміну службовою інформацією між основним контролером та периферійними модулями, а також для відображення поточного стану системи через людино-машинний інтерфейс.

Основною задачею є створення програмної логіки головного пристрою, який забезпечує керування 32 виконавчими каналами. Кожен канал має відображати власний стан, брати участь у загальному циклі обміну даними та передавати інформацію про готовність, помилки або зміну режиму роботи. Для забезпечення надійності роботи системи необхідно передбачити контроль зв’язку з вузлами, перевірку коректності прийнятих даних, обробку помилок та індикацію несправностей.

Програмне забезпечення повинно працювати на мікроконтролері STM32H750VBT6 та взаємодіяти з периферійними пристроями мікроконтролера. Для обміну даними використовується інтерфейс UART із подальшою передачею сигналів через RS-485, що дозволяє організувати стабільний зв’язок між основним контролером та виконавчими вузлами на відносно великій відстані [3]. Обмін даними має виконуватися за власним протоколом, який передбачає наявність заголовка пакета, службових полів, нумерації повідомлень та контрольної суми CRC [6].

Структурну схему програмно-апаратної системи керування периферійними вузлами наведено у графічній частині 2026.КВР.122.421.06.00.00 Е1. На схемі відображено центральний керуючий пристрій на базі мікроконтролера STM32H750VBT6, людино-машинний інтерфейс на базі TFT-дисплея ILI9341, комунікаційний інтерфейс UART/RS-485 та групу периферійних вузлів, з якими

виконується обмін командами, відповідями та діагностичними даними.

Окремою задачею є реалізація системи діагностики. Головний контролер повинен періодично перевіряти наявність зв’язку з підключеними вузлами, аналізувати відповіді від них та визначати можливі аварійні ситуації. У разі втрати зв’язку, некоректної відповіді або виявлення помилки програмне забезпечення має перевести відповідний канал у стан несправності та відобразити це на екрані.

Для взаємодії користувача з пристроєм необхідно реалізувати людино-машинний інтерфейс на базі TFT-дисплея ILI9341 [7]. Інтерфейс повинен забезпечувати відображення меню, режимів роботи, станів виконавчих каналів та службової діагностичної інформації. Користувач має мати можливість переглядати поточний стан системи, обирати режим роботи та контролювати готовність окремих каналів.

У програмному забезпеченні необхідно передбачити декілька режимів роботи, зокрема режим одиночної активації, режим групової активації та режим підготовки/заряджання. У кожному режимі система повинна виконувати відповідну послідовність дій, контролювати допустимість переходів між станами та не дозволяти виконання команд у разі виявлення несправностей або відсутності підтвердження від відповідних вузлів.

Розробка програмного забезпечення виконується з використанням середовища STM32CubeIDE та бібліотеки HAL [11, 12]. Такий підхід дозволяє спростити налаштування периферійних модулів мікроконтролера, забезпечити зрозумілу структуру програмного коду та прискорити процес налагодження. Для перевірки логіки обміну даними та тестування окремих функцій також може використовуватися допоміжне програмне забезпечення на ПК, яке імітує роботу периферійних вузлів.

Отже, задача розробки полягає у створенні надійного вбудованого програмного забезпечення, яке забезпечує керування багатоканальною системою, обмін даними з виконавчими вузлами, контроль їхнього стану, обробку помилок та відображення інформації користувачу через графічний інтерфейс. Результатом виконання цього етапу має бути структурована програмна система, придатна для

подальшої реалізації, тестування та використання у складі виконавчого пристрою.

Під час постановки задачі також необхідно врахувати, що програмне забезпечення має бути придатним до подальшого розширення та супроводу. Для цього структура програми повинна передбачати чітке розділення функцій керування, обміну даними, діагностики та відображення інформації, що спрощує внесення змін і пошук можливих помилок під час налагодження.

2.2 Опис та обґрунтування вибору структури та методу організації вхідних та вихідних даних

Програмне забезпечення, що розробляється, реалізоване у вигляді проєкту STM32CubeIDE для мікроконтролера STM32H750VBT6 [1, 2]. У проєкті використовується операційна система реального часу FreeRTOS, що дозволяє розділити виконання основних функцій системи між окремими задачами [8, 9]. Такий підхід підвищує структурованість програми, спрощує супровід коду та дозволяє незалежно обробляти обмін даними, оновлення інтерфейсу користувача, обробку кнопок і логіку керування пристроєм.

Основна прикладна логіка програми зосереджена у файлі `main.c`. У ньому реалізовано ініціалізацію периферійних модулів мікроконтролера, запуск операційної системи, частину логіки меню, обробку команд користувача, формування команд для периферійних вузлів, діагностику та відображення інформації на дисплеї. Після початкової ініціалізації класичний нескінченний цикл програми практично не використовується, оскільки подальша робота виконується задачами FreeRTOS.

У файлі app_tasks.c реалізовано задачі FreeRTOS, які відповідають за виконання прикладної логіки, обмін даними, оновлення інтерфейсу та обробку вхідних сигналів від кнопок. Реалізацію основних задач FreeRTOS наведено у додатку А.

Файл app_events.c використовується для організації черг подій і команд між задачами. Реалізацію черг подій між задачами наведено у додатку Б.

Модуль app_comm.c містить функції, пов’язані з обміном даними, зокрема розрахунок CRC, запис багатобайтових значень у форматі little-endian та перевірку контрольної суми прийнятих пакетів.

Для збереження інформації про периферійні вузли використовується стала кількість каналів, що дорівнює 32. Один вузол описується структурою cerber_device_info_t, структуру якої наведено у додатку В. Вона містить номер відповіді, поточний стан вузла, відповідь на команду, значення напруги батареї, значення напруги конденсатора, код помилки, додатковий параметр помилки, час останнього оновлення та номер пакета. Завдяки цьому головний контролер може зберігати не лише факт наявності вузла, а й повну діагностичну інформацію, необхідну для відображення стану системи та прийняття рішень під час роботи.

Обмін даними з периферійними вузлами реалізовано через USART3 із використанням DMA. Реалізацію приймання даних через UART DMA подано у додатку Г. Для приймання застосовується режим HAL_UARTEx_ReceiveToIdle_DMA, який дозволяє приймати дані до моменту простою на лінії, що зручно для роботи з пакетним протоколом [10, 11]. Передавання команд виконується за допомогою HAL_UART_Transmit_DMA. Фізично обмін передбачає використання інтерфейсу RS-485, однак у програмній реалізації немає окремого явного керування лініями напрямку передавання DE/RE. Це означає, що керування напрямком або реалізоване апаратно, або не винесене в окремий програмний драйвер.

Приймання та розбір пакетів від периферійних вузлів виконується у модулі cerber_parser.c. Він відповідає за пошук заголовків пакетів у потоці даних, визначення повного кадру відповіді та передавання коректно прийнятої інформації для подальшої обробки. Реалізацію обробки прийнятих пакетів наведено у додатку Ґ.

Модуль cerber_model.c використовується для очищення та підготовки структур діагностичних даних. Реалізацію оновлення діагностичної моделі подано у додатку Д.

Для роботи з графічним дисплеєм застосовується драйвер ili9341.c, який

забезпечує взаємодію з TFT-дисплеєм ILI9341 через інтерфейс SPI4. Реалізацію оновлення графічного інтерфейсу наведено у додатку Е.

Основні визначення команд, станів, структур пакетів та кодів помилок зосереджені у файлі cerber_defines.h.

Узагальнену схему взаємодії програмних модулів наведено у графічній частині 2026.КВР.122.421.06.00.00 СВ. На схемі відображено взаємозв'язок між основною прикладною логікою, задачами FreeRTOS, чергами подій, модулем обміну даними UART/RS-485, парсером прийнятих пакетів, діагностичною моделлю та модулем графічного інтерфейсу TFT-дисплея. Така схема дає змогу наочно показати розподіл функцій між програмними модулями та порядок передавання подій, команд і діагностичних даних у межах програмного забезпечення.

Особливістю реалізації є те, що інформація про 32 канали зберігається не як таблиця з жорстко закріпленими адресами, а як послідовність відповідей у межах одного кадру обміну. Поле `response_index` використовується як порядковий номер відповіді та має 1-базовану нумерацію. Під час виконання команд активації цей індекс використовується як цільовий номер вузла у полі підкоманди. Такий підхід спрощує обробку послідовних відповідей та дозволяє формувати актуальний список активних вузлів за результатами останнього циклу обміну.

Для групового збереження діагностичної інформації використовуються структури `cerber_response_frame_t` та `app_diag_snapshot_t`. Перша описує кадр відповідей після одного циклу обміну, а друга є знімком діагностичних даних, який використовується для безпечного оновлення графічного інтерфейсу. Також у програмі застосовуються масиви для збереження поточного набору даних про вузли та подвійний буфер діагностичних знімків. Це дозволяє розділити процес приймання даних і процес їх відображення на екрані.

У програмі передбачено декілька станів периферійного вузла. До них належать: неготовий стан, готовність до заряджання, процес заряджання, готовність до активації, активований стан та аварійний стан. Таке представлення дозволяє однозначно визначити, які дії можна виконувати з конкретним вузлом у

поточний момент. Наприклад, активація дозволяється лише для вузлів, які перебувають у стані готовності до активації, а заряджання — для вузлів, які готові до цього режиму.

Загальна логіка роботи системи передбачає декілька режимів: головне меню, діагностику, одиночну активацію, групову активацію, заряджання та режим відображення повідомлень. У головному меню користувач обирає потрібну дію. У режимі діагностики система відображає стан усіх вузлів. В одиночному режимі виконується команда для одного доступного вузла, у груповому — формується послідовність команд для кількох вузлів. Режим заряджання використовується для переведення підготовлених вузлів у відповідний стан. Режим повідомлень застосовується для інформування користувача про помилки, попередження або неможливість виконання команди.

Протокол обміну між головним контролером і периферійними вузлами має фіксовану структуру пакетів. Пакет головного пристрою має розмір 8 байт і містить заголовок, номер пакета, команду, підкоманду та контрольну суму CRC. Для таких пакетів використовується заголовок `RC_HEADER`, значення якого дорівнює `0xFEFE`. Номер пакета є 8-бітним і збільшується під час формування нових команд. Основними командами головного пристрою є `RC_CMD_PING`, `RC_CMD_CHARGING` та `RC_CMD_ACTIVATING`.

Пакет відповіді периферійного вузла має розмір 13 байт. Він містить заголовок, номер пакета, відповідь на команду, стан вузла, параметри вузла та контрольну суму CRC. Для відповідей використовується заголовок `CERBER_HEADER`, значення якого дорівнює `0xFE01`. У параметрах відповіді передаються напруга батареї, напруга конденсатора, код помилки та додатковий параметр помилки. На відміну від універсальних протоколів змінної довжини, у цій реалізації немає окремих полів адреси, довжини та довільного поля даних. Це спрощує структуру пакета й робить обробку швидшою, оскільки розмір повідомлень є наперед визначеним.

Для контролю цілісності даних використовується CRC-16 [6]. Розрахунок виконується функцією `app_comm_crc16` із початковим значенням `0xFFFF`. Алгоритм

відповідає CRC-16 типу CCITT і реалізований побітово, без використання таблиці. Для пакета головного пристрою контрольна сума обчислюється за першими 6 байтами, а для пакета відповіді — за першими 11 байтами. Значення CRC записується у форматі little-endian.

Схему структури протоколу обміну даними наведено у графічній частині 2026.КВР.122.421.06.00.00 СС. Вона відображає формат пакета головного пристрою, формат пакета відповіді периферійного вузла, службові заголовки, номер пакета, командні поля, параметри відповіді та поле контрольної суми CRC. Використання такої схеми дає змогу наочно показати порядок формування, передавання, приймання та перевірки пакетів у процесі обміну даними між основним контролером і периферійними вузлами.

Перевірка коректності прийнятого пакета виконується у парсері. Він аналізує потік байтів, шукає заголовки пакетів головного пристрою та периферійних вузлів, після чого визначає, чи отримано повний пакет. Для пакетів відповіді додатково виконується перевірка CRC. Якщо пакет має правильний заголовок, очікувану довжину та коректну контрольну суму, його дані використовуються для оновлення діагностичної моделі. Якщо пакет некоректний, він не використовується для зміни стану системи.

Втрата зв’язку визначається за часом останнього прийому даних. Для цього використовується змінна, що зберігає час останнього прийнятого пакета. Якщо під час роботи в режимі діагностики протягом встановленого часу не надходять нові дані, система встановлює ознаку втрати зв’язку, очищує сітку каналів і виводить на екран повідомлення про відсутність з’єднання. У реалізації використовується таймаут 2000 мс, що дозволяє оперативно виявляти зникнення обміну з периферійними вузлами.

Людино-машинний інтерфейс реалізовано на базі TFT-дисплея ILI9341 [7]. Дисплей підключений через SPI4, а для керування використовуються окремі GPIO-сигнали підсвітки, скидання, вибору кристала та вибору режиму даних/команд. Інтерфейс містить головне меню, екран діагностики, детальний екран окремого вузла та екрани повідомлень. Головне меню дає змогу обрати одиночну активацію,

групову активацію, діагностику або заряджання.

Стан 32 вузлів у режимі діагностики відображається у вигляді сітки 4×8. Кожна клітинка відповідає окремому вузлу та має колір, що залежить від його поточного стану. Сірий колір використовується для позначення неготового стану, блакитний колір відповідає готовності до заряджання, жовтий колір позначає процес заряджання, зелений колір використовується для позначення готовності до активації, синій колір відповідає активованому стану, а червоний колір позначає аварійний стан. У клітинці також виводиться номер вузла. Додатково на екрані відображаються кількість активних вузлів, напруга джерела живлення та час відповіді.

Детальний екран використовується для перегляду інформації про конкретний вузол. На ньому відображаються номер вузла, відповідь на команду, поточний стан, напруга батареї, напруга конденсатора, код помилки та параметр помилки. Навігація між вузлами здійснюється кнопками, а повернення до попереднього екрана виконується окремою кнопкою керування.

Оновлення дисплея не виконується безпосередньо з переривання UART. Прийняті через DMA дані спочатку обробляються у комунікаційній задачі, після чого формується діагностичний знімок. Далі через чергу подій інформація передається задачі інтерфейсу, яка оновлює відповідний екран. Такий підхід дозволяє уникнути блокування приймання даних під час перемальовування графіки та розділяє логіку обміну, обробки станів і відображення інформації.

Вхідними даними системи є відповіді периферійних вузлів, команди користувача та внутрішні стани програмних модулів. Вихідними даними є команди для вузлів, діагностична інформація та дані, що відображаються на дисплеї.

Таким чином, структура програмного забезпечення побудована за модульним принципом із використанням задач FreeRTOS, черг подій, DMA-обміну та окремих структур для збереження стану периферійних вузлів. Основними елементами системи є модуль комунікації, парсер пакетів, модель діагностичних даних, логіка режимів роботи та графічний інтерфейс користувача. Така організація дозволяє забезпечити стабільний обмін даними, своєчасне оновлення стану 32

каналів і зручне відображення інформації для користувача.

2.3 Розробка алгоритму

Розробка алгоритму роботи програмного забезпечення є одним із ключових етапів створення мультиканального виконавчого пристрою. Алгоритм визначає послідовність дій системи після подачі живлення, порядок обміну даними з периферійними вузлами, обробку команд користувача, контроль стану каналів, оновлення графічного інтерфейсу та реакцію на помилки або втрату зв’язку.

Основною вимогою до алгоритму є забезпечення стабільної роботи системи з 32 виконавчими каналами. Для цього програмне забезпечення повинно регулярно отримувати діагностичні дані від периферійних вузлів, перевіряти коректність прийнятих пакетів, оновлювати внутрішню модель стану системи та своєчасно відображати інформацію користувачу. Оскільки пристрій працює з декількома незалежними процесами, алгоритм побудовано з використанням задач операційної системи реального часу FreeRTOS [8, 9].

Після старту пристрою виконується ініціалізація мікроконтролера, налаштування тактування, портів введення-виведення, інтерфейсів обміну, DMA, дисплея та службових структур даних. Після цього створюються черги обміну повідомленнями та задачі FreeRTOS. Подальша робота системи виконується не в одному нескінченному циклі, а розподіляється між окремими задачами, які відповідають за прикладну логіку, комунікацію, інтерфейс користувача та обробку кнопок.

Алгоритм комунікації передбачає періодичний обмін пакетами між головним контролером і периферійними вузлами. Приймання даних виконується через UART із використанням DMA, що зменшує навантаження на процесор і дозволяє стабільно обробляти потік даних [10, 11]. Після приймання фрагмента повідомлення передається до програмного парсера, який перевіряє структуру пакета та контрольну суму. Коректні пакети використовуються для оновлення діагностичної моделі, а пошкоджені або некоректні повідомлення відкидаються.

Окремим елементом алгоритму є контроль втрати зв’язку. Програма зберігає час останнього приймання даних від периферійних вузлів. Якщо протягом заданого інтервалу нові відповіді не надходять, система встановлює ознаку втрати з’єднання, очищує діагностичну сітку та виводить повідомлення про відсутність зв’язку. Після відновлення обміну діагностична інформація знову оновлюється.

Для взаємодії з користувачем алгоритм передбачає роботу головного меню, режиму діагностики, детального екрана вузла та службових повідомлень. Через меню користувач може обрати діагностику, заряджання, одиночну активацію або групову активацію. Перед виконанням керуючих дій система аналізує поточний стан вузлів і виконує команду лише для тих каналів, які перебувають у допустимому стані.

У режимі одиночної активації програма визначає перший вузол, який перебуває у стані готовності до активації, і формує для нього відповідну команду. У режимі групової активації формується список усіх готових вузлів, після чого команди передаються послідовно з фіксованою паузою. Режим заряджання використовується для переведення вузлів у підготовчий стан і виконується тільки для вузлів, готових до цієї операції.

Для запобігання випадковому виконанню керуючих дій у програмі передбачено захисний режим Safe. Якщо цей режим увімкнений, одиночна та групова активація блокуються, а користувач отримує відповідне повідомлення на дисплеї. Це дозволяє підвищити безпечність експлуатації пристрою під час навігації по меню або перегляду діагностичної інформації.

Таким чином, розроблений алгоритм забезпечує початкову ініціалізацію пристрою, циклічний обмін даними з периферійними вузлами, контроль стану 32 каналів, обробку команд користувача, захист від випадкової активації, виявлення втрати зв’язку та відображення актуальної інформації на дисплеї. Деталізація зовнішньої взаємодії користувача з програмою наведена в підпункті 2.3.1, а внутрішня логіка роботи програмного забезпечення розглянута в підпункті 2.3.2.

2.3.1 Зовнішнє проєктування програми

Зовнішнє проєктування програми передбачає визначення способу взаємодії користувача з програмним забезпеченням, опис основних режимів роботи, вхідних і вихідних даних, а також повідомлень, які відображаються під час роботи пристрою. На цьому етапі програма розглядається не з погляду внутрішньої реалізації коду, а з погляду користувача, який керує пристроєм через кнопки та отримує інформацію через графічний дисплей.

Розроблюване програмне забезпечення призначене для роботи у складі мультиканального виконавчого пристрою на базі мікроконтролера STM32. Основним засобом взаємодії користувача з пристроєм є людино-машинний інтерфейс, реалізований на базі TFT-дисплея ILI9341 та кнопок керування [7]. Через цей інтерфейс користувач може переглядати стан виконавчих вузлів, обирати режими роботи, запускати підготовчі або виконавчі дії, а також отримувати повідомлення про помилки та службові стани системи.

Зовнішня структура програми побудована навколо кількох основних екранів. До них належать головне меню, екран діагностики, детальний екран окремого вузла та екран службових повідомлень. Головне меню використовується для вибору режиму роботи. Екран діагностики призначений для одночасного контролю стану всіх 32 виконавчих вузлів. Детальний екран дозволяє переглянути розширену інформацію про конкретний вузол. Екран повідомлень використовується для інформування користувача про виконання дії, попередження або помилку.

У головному меню користувачу доступні основні режими роботи: одиночна активація, групова активація, діагностика та заряджання. Одиночна активація призначена для виконання команди для одного вузла, який перебуває у стані готовності. Групова активація використовується для послідовного виконання команд для кількох готових вузлів. Режим діагностики дозволяє переглянути стан усієї системи. Режим заряджання використовується для переведення вузлів у підготовчий стан.

Для зменшення ризику випадкового виконання команд активації в інтерфейсі передбачено режим блокування Safe. Стан цього режиму відображається в меню та може перемикатися користувачем за допомогою кнопки керування. Якщо Safe увімкнений, виконання одиночної та групової активації блокується, а на екрані відображається повідомлення про заборону виконання дії. Це дозволяє підвищити безпеку користування пристроєм під час перегляду меню або діагностики.

Керування програмою здійснюється за допомогою кнопок. У головному меню кнопки використовуються для переміщення між пунктами, підтвердження вибору та перемикання режиму Safe. У режимі діагностики кнопки дозволяють перейти до детального перегляду вузла, перемикатися між вузлами та повертатися до попереднього екрана. Такий спосіб керування є простим і зручним для вбудованого пристрою, оскільки не потребує складних засобів введення.

Екран діагностики є одним із основних елементів зовнішнього інтерфейсу програми. На ньому стан 32 виконавчих вузлів відображається у вигляді сітки 4×8. Кожна клітинка відповідає одному вузлу та містить його номер. Колір клітинки залежить від поточного стану вузла. Завдяки цьому користувач може швидко оцінити загальний стан системи без необхідності переглядати кожен вузол окремо.

У програмі передбачено декілька станів вузла, які відображаються в інтерфейсі: неготовий стан, готовність до заряджання, процес заряджання, готовність до активації, активований стан та аварійний стан. Відображення цих станів різними кольорами спрощує сприйняття інформації та дозволяє швидко виявити вузли, які готові до роботи, перебувають у процесі підготовки або мають несправність.

На екрані діагностики також відображається службова інформація про систему. До неї належать кількість активних вузлів, окремі параметри першого вузла та інформація про час відповіді. Якщо зв’язок із периферійними вузлами втрачено, сітка очищується, а на екрані з’являється повідомлення про відсутність з’єднання. Це дозволяє користувачу одразу зрозуміти, що дані на екрані не оновлюються через проблему з обміном.

Детальний екран вузла використовується для перегляду розширеної діагностичної інформації. На ньому відображається номер вузла, відповідь на останню команду, поточний стан, значення напруги, код помилки та додатковий параметр помилки. Цей екран потрібний для аналізу стану конкретного вузла у разі несправності або під час перевірки правильності роботи системи.

Вхідними даними для програми з боку користувача є натискання кнопок, за допомогою яких здійснюється вибір режиму, навігація по меню, перехід між екранами та підтвердження дій. Вхідними даними з боку периферійних вузлів є пакети відповідей, які містять інформацію про стан вузлів, службові параметри та коди помилок. Вихідними даними програми є команди, що передаються до периферійних вузлів, а також графічна інформація, яка відображається на дисплеї.

З погляду користувача робота програми відбувається за таким принципом. Після запуску пристрою на екрані з’являється початковий інтерфейс. Користувач обирає потрібний режим роботи через меню або переходить до діагностики. У режимі діагностики він переглядає стан усіх вузлів або відкриває детальну інформацію про окремий вузол. При виборі режиму активації програма перевіряє готовність відповідних вузлів і або виконує команду, або виводить повідомлення про неможливість її виконання. У разі втрати зв’язку або появи несправності користувач отримує відповідне повідомлення через дисплей.

Таким чином, зовнішнє проєктування програми визначає зручний і зрозумілий спосіб взаємодії користувача з мультиканальним виконавчим пристроєм. Інтерфейс забезпечує вибір режимів роботи, контроль стану 32 вузлів, перегляд детальної інформації та відображення повідомлень про помилки. Така організація дозволяє користувачу швидко оцінювати стан системи та виконувати необхідні дії без доступу до внутрішньої логіки програмного забезпечення.

2.3.2 Проєктування логіки програми

Проєктування логіки програми передбачає визначення внутрішньої послідовності роботи програмного забезпечення, взаємодії між його

функціональними частинами, способу обробки команд користувача, обміну даними з периферійними вузлами та оновлення інформації на дисплеї. На відміну від зовнішнього проєктування, де основна увага приділяється взаємодії користувача з пристроєм, у цьому підпункті розглядається внутрішня організація роботи програми.

Основою програмної логіки є структурний підхід до побудови програмного забезпечення. Функції системи поділено на окремі логічні частини: ініціалізацію пристрою, обмін даними, обробку подій користувача, діагностику стану вузлів, виконання режимів роботи та оновлення графічного інтерфейсу. Такий підхід дозволяє зробити програму зрозумілішою, спростити її налагодження та забезпечити можливість подальшого розширення.

Після запуску мікроконтролера виконується початкова ініціалізація апаратної та програмної частини системи. Налаштовуються тактування, порти введення-виведення, інтерфейси обміну, контролер DMA, таймери та дисплей. Після цього готуються внутрішні структури, які використовуються для збереження діагностичної інформації про виконавчі вузли. Також запускається приймання даних через UART із використанням DMA, що дозволяє приймати інформацію від периферійних вузлів без постійного опитування інтерфейсу центральним процесором.

Подальша робота програми організована за допомогою операційної системи реального часу FreeRTOS [8, 9]. Після створення черг і задач керування передається планувальнику, який розподіляє виконання між окремими задачами. У програмі передбачено задачі для прикладної логіки, комунікації, графічного інтерфейсу та обробки кнопок. Такий розподіл дозволяє одночасно підтримувати обмін із вузлами, реагувати на дії користувача та оновлювати дисплей.

Прикладна задача координує загальну логіку роботи пристрою. Вона приймає події від кнопок, обробляє вибрані користувачем режими, виконує періодичну перевірку стану системи та формує команди для інших частин програми. Саме на цьому рівні визначається, чи потрібно перейти до діагностики, виконати одиночну активацію, запустити групову дію, активувати режим

заряджання або показати користувачу службове повідомлення.

Комунікаційна задача відповідає за обмін даними між головним пристроєм і периферійними вузлами. Вона обробляє прийняті через UART дані, передає їх до парсера пакетів, формує команди для передавання та забезпечує періодичну підтримку зв’язку. Для приймання використовується режим DMA з визначенням простою на лінії UART. Завдяки цьому програма може визначити завершення прийнятого фрагмента даних і передати його на подальшу обробку без використання складного механізму ручного опитування інтерфейсу.

Пакети, отримані від периферійних вузлів, проходять перевірку на відповідність очікуваній структурі. Програма аналізує службовий заголовок, визначає повноту прийнятого пакета та перевіряє контрольну суму CRC для відповідей вузлів. Якщо пакет прийнято коректно, його дані використовуються для оновлення діагностичної моделі. Якщо пакет пошкоджений або не відповідає очікуваному формату, він відкидається і не змінює поточний стан системи.

Для підтримання актуального стану вузлів головний пристрій періодично надсилає службову команду обміну. У звичайному режимі вона використовується для перевірки зв’язку та отримання відповідей від периферійних вузлів. Якщо активовано певний режим роботи, під час чергового циклу обміну може передаватися команда заряджання або активації. Таким чином, логіка передавання команд поєднана з періодичним контролем стану системи.

Блок-схему алгоритму діагностики периферійних вузлів наведено у графічній частині 2026.КВР.122.421.06.00.00 БС. Вона відображає послідовність періодичного опитування периферійних вузлів, формування службового запиту, передавання пакета через інтерфейс UART/RS-485, приймання відповіді, перевірку структури пакета та контрольної суми CRC, обробку помилок і таймаутів, а також оновлення діагностичної моделі системи.

Діагностична модель програми використовується для збереження актуальних даних про вузли. Після приймання та перевірки пакетів формується знімок стану системи, який містить інформацію про активні вузли, їхні стани та службові параметри. Такий знімок передається до частини програми, що відповідає

за графічне відображення. Це дозволяє розділити процес приймання даних і процес оновлення дисплея, що підвищує стабільність роботи програми.

Окремо реалізовано контроль втрати зв’язку. Програма зберігає час останнього приймання даних від периферійних вузлів. Якщо в режимі діагностики протягом заданого інтервалу нові дані не надходять, система вважає зв’язок втраченим. У такому випадку діагностична сітка очищується, а користувачу виводиться повідомлення про відсутність з’єднання. Після відновлення приймання даних система повертається до штатного оновлення діагностичної інформації.

Групова активація реалізована як послідовне виконання команд для кількох вузлів. Спочатку програма формує список вузлів, які перебувають у стані готовності до активації. Якщо список порожній, користувач отримує повідомлення про відсутність готових вузлів. Якщо список сформовано, програма послідовно передає команди активації з фіксованою паузою між ними. Такий підхід дозволяє виконувати групову дію керовано та не створювати одночасного навантаження на систему.

Режим заряджання відрізняється від режиму активації тим, що відповідна команда зберігається як поточний режим обміну. Приклад реалізації режиму заряджання наведено у додатку Є. Перед її встановленням програма перевіряє, чи є вузли, готові до заряджання. Якщо таких вузлів немає, команда не виконується. Якщо готовий хоча б один вузол, команда заряджання встановлюється як активна і передається під час наступних циклів періодичного обміну. Якщо частина вузлів не готова, користувачу може бути показано попередження.

Логіка одиночної активації передбачає пошук вузла, який перебуває у стані готовності до активації. Приклад реалізації одиночної та групової активації наведено у додатку Ж. Якщо такий вузол знайдено, програма формує команду активації для відповідного каналу та передає її до комунікаційної задачі. Команда активації виконується як одноразова дія, після чого система повертається до штатного періодичного обміну. Якщо жоден вузол не перебуває у потрібному стані, команда не виконується, а користувачу виводиться повідомлення про неможливість активації.

У програмі передбачено захисну логіку, пов’язану з режимом Safe. Цей режим блокує виконання одиночної та групової активації, якщо він увімкнений. Перевірка виконується перед викликом відповідних дій, тому заборонена команда не передається до комунікаційної частини програми. При цьому підготовчі операції, зокрема режим заряджання, не блокуються цим параметром.

Задача інтерфейсу користувача відповідає за відображення актуального стану системи на дисплеї. Інші частини програми не виконують перемальовування екрана безпосередньо, а лише формують запити на оновлення. Залежно від типу події задача інтерфейсу виконує повне перемальовування екрана або оновлює лише окремі елементи, наприклад вибраний пункт меню, клітинки діагностичної сітки чи поля детального екрана. Це дозволяє зменшити навантаження на мікроконтролер і зробити роботу інтерфейсу швидшою.

Обробка кнопок винесена в окрему задачу. Вона періодично перевіряє стан елементів керування, визначає факти натискання та формує події для прикладної задачі. Завдяки цьому основна логіка програми не прив’язана безпосередньо до апаратного опитування кнопок, а працює з уже сформованими подіями. Такий підхід спрощує обробку навігації по меню та виконання дій користувача.

Аварійні стани вузлів обробляються через діагностичну модель і графічний інтерфейс. Якщо вузол передає стан несправності, він позначається відповідним чином у діагностичній сітці, а на детальному екрані можуть відображатися код помилки та додатковий параметр. Загальне блокування всієї системи при несправності одного вузла не виконується, оскільки алгоритми активації працюють лише з тими вузлами, які перебувають у стані готовності. Це дозволяє продовжувати контроль системи навіть за наявності несправних каналів.

Узагальнений алгоритм логіки програми можна описати як циклічну взаємодію кількох функціональних частин. Комунікаційна задача періодично підтримує обмін із периферійними вузлами та оновлює діагностичні дані. Прикладна задача обробляє події користувача й визначає поточний режим роботи. Задача введення формує події від кнопок, а задача інтерфейсу відображає актуальну інформацію на дисплеї. Усі ці частини взаємодіють через черги, що

забезпечує впорядковану передачу команд і подій.

Передавання подій через черги впорядковує взаємодію між задачами та робить виконання алгоритму більш передбачуваним під час обміну даними, обробки команд користувача й оновлення інтерфейсу.

Таким чином, логіка програми побудована за модульним принципом із використанням структурного програмування, задач FreeRTOS, черг подій і пакетного обміну даними. Така організація дозволяє забезпечити стабільну роботу пристрою, своєчасне оновлення стану 32 виконавчих вузлів, виконання команд користувача, контроль втрати зв’язку та відображення діагностичної інформації на графічному дисплеї.

2.4 Визначення інформаційних зв’язків

Інформаційні зв’язки програмного забезпечення визначають порядок передавання даних між користувачем, головним контролером, периферійними вузлами та графічним інтерфейсом. Для мультиканального виконавчого пристрою такі зв’язки є важливою частиною проєктування, оскільки система повинна одночасно приймати команди від користувача, обмінюватися даними з виконавчими вузлами, контролювати їхній стан і відображати актуальну інформацію на дисплеї.

Основним центром обробки інформації є головний контролер на базі STM32. Він приймає вхідні дані від кнопок керування, формує команди для периферійних вузлів, приймає відповіді від них, оновлює внутрішню діагностичну модель і передає дані для відображення на дисплей. Таким чином, головний контролер виконує роль координатора між усіма частинами системи.

Зовнішні інформаційні зв’язки програми можна поділити на три основні групи: зв’язок із користувачем, зв’язок із периферійними вузлами та зв’язок із засобами відображення. Користувач взаємодіє з пристроєм через кнопки керування та отримує інформацію через TFT-дисплей. Периферійні вузли обмінюються з головним контролером службовими пакетами, які містять команди, стани, відповіді

та діагностичні параметри. Дисплей використовується для виведення меню, діагностичної сітки, детальної інформації про вузли та службових повідомлень.

Вхідними даними з боку користувача є натискання кнопок. Вони використовуються для переміщення по меню, вибору режиму роботи, підтвердження дії, переходу до режиму діагностики, перегляду детальної інформації про вузол і перемикання захисного режиму Safe. Після фіксації натискання кнопки програма формує відповідну подію, яка передається до прикладної логіки. Такий підхід дозволяє відокремити апаратне опитування кнопок від основного алгоритму роботи програми.

До вихідних даних для користувача належить уся інформація, що виводиться на дисплей. Це головне меню, поточний стан режиму Safe, діагностична сітка 32 вузлів, кольорове відображення станів, кількість активних вузлів, службові параметри, детальна інформація про вибраний вузол, попередження та повідомлення про помилки. Завдяки цьому користувач отримує зрозумілий спосіб контролю системи без потреби аналізувати внутрішні дані програми.

Інформаційний зв’язок між головним контролером і периферійними вузлами реалізовано через послідовний інтерфейс UART із використанням DMA-приймання. На фізичному рівні обмін призначений для роботи через лінію RS-485, що дозволяє підключати групу виконавчих вузлів до головного контролера. Головний пристрій періодично формує пакети команд, а периферійні вузли передають у відповідь пакети зі своїм поточним станом і службовими параметрами.

Дані, які передаються від головного контролера до периферійних вузлів, мають командний характер. До них належать службова команда перевірки зв’язку, команда заряджання та команда активації. Команда перевірки зв’язку використовується для підтримання регулярного обміну й отримання актуальної інформації про стан вузлів. Команда заряджання переводить підготовлені вузли у відповідний режим. Команда активації використовується для виконання дії вибраним або визначеним алгоритмом вузлом.

Дані, які надходять від периферійних вузлів до головного контролера, мають діагностичний характер. Вони містять інформацію про відповідь на

команду, поточний стан вузла, службові електричні параметри, код помилки та додатковий параметр помилки. Після приймання такі дані проходять перевірку, обробляються парсером і використовуються для оновлення внутрішньої моделі стану системи.

Для забезпечення достовірності обміну використовується контроль цілісності пакетів. Прийняті відповіді перевіряються за службовими ознаками та контрольною сумою CRC. Якщо пакет не відповідає очікуваній структурі або має некоректну контрольну суму, він не використовується для оновлення стану системи. Завдяки цьому зменшується ймовірність неправильного відображення стану вузлів у разі пошкодження даних під час передавання.

Внутрішні інформаційні зв’язки програми організовано через задачі FreeRTOS і черги повідомлень. Окрема задача відповідає за обробку вхідних сигналів від кнопок, інша — за прикладну логіку, ще одна — за комунікацію з периферійними вузлами, а окрема задача — за оновлення графічного інтерфейсу. Передавання даних між цими частинами виконується не через пряме звернення до всіх функцій одразу, а через події, команди та запити на перемальовування.

Задача обробки введення формує події натискання кнопок і передає їх до прикладної задачі. Прикладна задача аналізує ці події, визначає поточний режим роботи та приймає рішення про подальші дії. Якщо користувач вибирає режим активації або заряджання, прикладна задача формує команду для комунікаційної задачі. Якщо потрібно оновити екран, вона формує запит до задачі інтерфейсу.

Комунікаційна задача отримує команди від прикладної логіки та формує пакети для передавання до периферійних вузлів. Також вона приймає дані через UART DMA, передає їх до парсера, перевіряє коректність прийнятих відповідей і оновлює діагностичні дані. Після отримання нового стану вузлів формується знімок діагностичної інформації, який надалі використовується для оновлення інтерфейсу.

Задача інтерфейсу користувача отримує запити на перемальовування екрана. Вона не виконує приймання даних від вузлів і не приймає рішень щодо режимів роботи, а лише відображає актуальну інформацію, підготовлену іншими

частинами програми. Такий поділ дозволяє уникнути ситуації, коли перемальовування дисплея затримує приймання даних або виконання команд.

Інформаційний зв’язок між діагностичною моделлю та інтерфейсом реалізовано через знімки стану системи. Після приймання відповідей від вузлів програма формує актуальний набір даних про їхній стан. Цей набір використовується для оновлення діагностичної сітки, детального екрана вузла та службової інформації. Завдяки такому підходу графічний інтерфейс працює з уже підготовленими даними, а не з необробленим потоком UART.

Окремим інформаційним зв’язком є контроль втрати з’єднання. Програма зберігає час останнього приймання даних від периферійних вузлів. Якщо протягом заданого часу нові дані не надходять, система встановлює ознаку втрати зв’язку. Ця інформація передається до інтерфейсу користувача, після чого на екрані відображається повідомлення про відсутність з’єднання. Після відновлення приймання даних діагностична інформація знову оновлюється.

У режимі одиночної активації інформаційний зв’язок має таку послідовність: користувач вибирає відповідний пункт меню, прикладна логіка перевіряє стан вузлів, визначає перший вузол, готовий до активації, формує команду для комунікаційної задачі, після чого команда передається до периферійного вузла. Результат виконання відображається через подальші відповіді вузлів і оновлення діагностичної моделі.

У режимі групової активації прикладна логіка формує список вузлів, які перебувають у стані готовності до активації. Далі команди передаються послідовно, з паузою між окремими вузлами. Інформація про виконання дій надалі надходить у вигляді відповідей від периферійних вузлів і відображається в діагностичній сітці. Такий порядок забезпечує кероване виконання групової операції.

У режимі заряджання програма спочатку перевіряє наявність вузлів, готових до переходу в підготовчий стан. Якщо такі вузли є, команда заряджання встановлюється як поточна команда обміну та передається під час наступних циклів зв’язку. Якщо частина вузлів не готова, користувач отримує попередження

через екран повідомлень. Таким чином, інформаційний зв’язок у цьому режимі охоплює перевірку станів, формування команди, передавання її до вузлів і відображення результату.

У режимі діагностики головним інформаційним потоком є передавання станів від периферійних вузлів до головного контролера. Програма періодично отримує відповіді, оновлює діагностичну модель і передає дані до графічного інтерфейсу. Користувач у цьому режимі переважно отримує інформацію, але також може відкривати детальний екран вузла або повертатися до меню.

Отже, інформаційні зв’язки програмного забезпечення побудовані за принципом розділення джерел даних, логіки обробки та засобів відображення. Кнопки формують події користувача, прикладна логіка приймає рішення, комунікаційна задача виконує обмін із периферійними вузлами, діагностична модель зберігає актуальний стан системи, а задача інтерфейсу відображає інформацію на дисплеї. Така організація забезпечує впорядковану передачу даних, стабільну роботу пристрою та зручний контроль стану 32 виконавчих вузлів.

2.5 Написання текстів програм

Написання тексту програми є завершальним етапом розробки технічного та робочого проєкту програмного забезпечення. На цьому етапі результати постановки задачі, опису структур даних, проєктування алгоритмів та визначення інформаційних зв’язків реалізуються у вигляді програмного коду для мікроконтролера STM32. Основною метою цього етапу є створення працездатної програмної системи, яка забезпечує керування мультиканальним виконавчим пристроєм, обмін даними з периферійними вузлами, діагностику їхнього стану та відображення інформації користувачу.

Програмне забезпечення головного пристрою розроблено у середовищі STM32CubeIDE з використанням мови програмування C. Це середовище є зручним для розробки вбудованих систем на базі мікроконтролерів STM32, оскільки поєднує засоби налаштування периферії, генерації початкового коду, компіляції,

прошивання та налагодження. Для роботи з периферійними модулями мікроконтролера використовується бібліотека HAL, яка забезпечує уніфікований доступ до апаратних блоків мікроконтролера.

Основою реалізації є структурне програмування. Програма поділена на окремі функціональні частини, кожна з яких відповідає за певний напрям роботи системи: ініціалізацію апаратних модулів, обмін даними, обробку команд користувача, діагностику вузлів, формування команд, оновлення дисплея та обробку службових станів. Такий підхід дозволяє зробити код зрозумілішим, зменшити залежність між окремими частинами програми та спростити подальше налагодження.

У процесі написання тексту програми основна увага приділялась стабільності роботи системи та передбачуваності виконання команд. Оскільки пристрій працює з 32 виконавчими вузлами, програмний код повинен забезпечувати регулярний обмін даними, своєчасне оновлення діагностичної інформації, контроль втрати зв’язку та коректну реакцію на дії користувача. Для цього програмна логіка реалізована з використанням операційної системи реального часу FreeRTOS.

Застосування FreeRTOS дозволило розділити програму на декілька задач. Окрема задача відповідає за прикладну логіку, інша — за комунікацію з периферійними вузлами, ще одна — за оновлення графічного інтерфейсу, а окрема задача використовується для обробки кнопок. Такий розподіл дає змогу уникнути блокування всієї системи під час виконання окремих операцій. Наприклад, приймання даних через UART не залежить безпосередньо від перемальовування дисплея, а обробка кнопок не заважає періодичному обміну з вузлами.

Для передавання подій і команд між задачами використовуються черги FreeRTOS, приклад реалізації яких наведено у додатку Б. Через них передаються події натискання кнопок, команди для комунікаційної частини програми та запити на оновлення екрана. Це дозволяє організувати впорядковану взаємодію між різними частинами програми без прямого жорсткого зв’язування всіх модулів між собою. Такий підхід підвищує надійність роботи програмного забезпечення та

полегшує його супровід.

Обмін даними з периферійними вузлами реалізовано через UART із використанням DMA. DMA дає змогу приймати та передавати дані з меншим навантаженням на центральне ядро мікроконтролера. Приймання виконується з визначенням простою на лінії UART, що дозволяє обробляти пакети після завершення прийнятого фрагмента. Такий підхід є зручним для пакетного обміну, оскільки програма може накопичувати прийняті байти, передавати їх до парсера та перевіряти коректність повідомлень.

Для контролю цілісності даних у програмі реалізовано перевірку CRC. Контрольна сума використовується для перевірки відповідей від периферійних вузлів. Якщо прийнятий пакет має некоректну структуру або неправильну контрольну суму, він не використовується для оновлення стану системи. Це дозволяє зменшити ризик помилкової зміни діагностичної інформації у разі спотворення даних під час передавання.

Пакети, отримані від периферійних вузлів, обробляються окремим парсером, приклад реалізації якого наведено у додатку Ґ. Його задача полягає у пошуку службових ознак пакета, визначенні повного повідомлення, перевірці коректності даних і передаванні результату до діагностичної моделі. Після успішної обробки відповіді програма оновлює інформацію про стан вузлів, їхні службові параметри та можливі помилки. Завдяки цьому головний контролер підтримує актуальне уявлення про стан усієї системи.

Для збереження даних про виконавчі вузли використовуються структури, які містять номер відповіді, стан вузла, відповідь на команду, службові електричні параметри, код помилки, параметр помилки, час останнього оновлення та номер пакета. Такі структури дозволяють компактно зберігати інформацію про кожен вузол і використовувати її як для логіки керування, так і для відображення на дисплеї.

Графічний інтерфейс користувача реалізовано для TFT-дисплея ILI9341, приклад оновлення якого наведено у додатку Е. У програмі передбачено виведення головного меню, діагностичної сітки 32 вузлів, детального екрана окремого вузла

та службових повідомлень. Для зменшення навантаження на мікроконтролер перемальовування екрана виконується не постійно, а за запитом. У разі зміни стану системи або дії користувача формується подія оновлення, після чого задача інтерфейсу перемальовує потрібний екран або його окремі елементи.

Під час написання тексту програми було реалізовано основні режими роботи пристрою: діагностику, одиночну активацію, групову активацію і заряджання. У режимі діагностики програма відображає стан усіх вузлів і оновлює інформацію після отримання нових відповідей. У режимі одиночної активації виконується команда для першого вузла, який перебуває у стані готовності до активації. У режимі групової активації формується список готових вузлів, після чого команди передаються послідовно з фіксованою паузою. У режимі заряджання команда встановлюється як поточна та передається під час періодичного обміну.

Окремо реалізовано захисну логіку режиму Safe. Якщо цей режим увімкнений, програма блокує виконання команд одиночної та групової активації. У такому випадку користувач отримує відповідне повідомлення на дисплеї, а команда не передається до периферійних вузлів. Це дозволяє зменшити ймовірність випадкового виконання дії під час навігації по меню або перегляду діагностичної інформації.

Контроль втрати зв’язку реалізовано на основі часу останнього приймання даних від периферійних вузлів. Якщо протягом заданого інтервалу нові відповіді не надходять, програма встановлює ознаку втрати з’єднання, очищує діагностичну сітку та виводить на дисплей повідомлення про відсутність зв’язку. Після відновлення обміну даними діагностична інформація знову оновлюється, а інтерфейс повертається до штатного відображення стану системи.

Під час написання коду також було враховано необхідність подальшого налагодження та перевірки роботи системи. Для цього програму побудовано таким чином, щоб окремі частини логіки можна було перевіряти незалежно: обробку кнопок, оновлення інтерфейсу, приймання пакетів, формування команд та оновлення діагностичної моделі. Такий підхід спрощує пошук помилок і дозволяє поступово перевіряти працездатність усієї системи.

Допоміжно для перевірки окремих сценаріїв роботи може використовуватися програмний емулятор на ПК. Він не є основною частиною вбудованого програмного забезпечення головного пристрою, але дає змогу перевіряти логіку обміну та реакцію системи на різні стани периферійних вузлів без постійного використання повного апаратного стенда. Детальніше використання такого засобу доцільно розглядати в розділі, присвяченому тестуванню програмного забезпечення.

Для контролю змін у програмному коді та збереження проміжних версій проєкту доцільно використовувати систему керування версіями Git, яка дає змогу відстежувати історію змін, повертатися до попередніх версій і спрощує супровід програмного забезпечення [16].

У результаті написання тексту програми було отримано програмне забезпечення, яке забезпечує ініціалізацію мікроконтролера, обмін даними з периферійними вузлами, контроль стану 32 каналів, обробку команд користувача, відображення інформації на дисплеї та реакцію на помилки зв’язку або аварійні стани вузлів. Реалізація з використанням структурного підходу, FreeRTOS, UART DMA та графічного інтерфейсу дозволяє забезпечити надійну й зручну роботу мультиканального виконавчого пристрою.

2.6 Тестування та налагодження програм

Для перевірки працездатності розробленого програмного забезпечення було розглянуто контрольний приклад роботи мультиканального виконавчого пристрою. Контрольний приклад дозволяє показати послідовність взаємодії користувача з пристроєм, порядок обміну даними між головним контролером і периферійними вузлами, а також реакцію програми на зміну станів системи.

Початковою умовою контрольного прикладу є запуск головного пристрою на базі мікроконтролера STM32. Після подачі живлення програма виконує ініціалізацію апаратних модулів, запускає приймання даних через UART, готує графічний дисплей і створює задачі FreeRTOS. Після завершення початкової

ініціалізації на дисплеї відображається початковий екран або головне меню, через яке користувач може перейти до потрібного режиму роботи.

На цьому етапі перевіряється правильність переходу програми від початкової ініціалізації до основного режиму роботи. Наявність головного меню підтверджує, що графічний інтерфейс запущено коректно, задачі операційної системи створені, а пристрій готовий до подальшого приймання команд користувача та відображення діагностичної інформації. Загальний вигляд головного меню, що відображається на дисплеї головного пристрою, показано на рисунку 2.1.

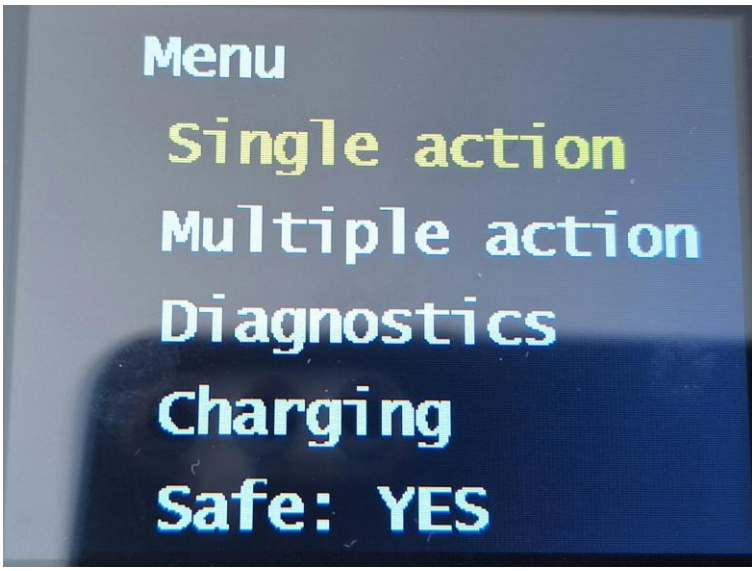


Рисунок 2.1 – Головне меню інтерфейсу головного пристрою

Інтерфейс головного меню побудований таким чином, щоб користувач міг швидко обрати один з основних режимів роботи пристрою. Наявність окремих пунктів для діагностики, заряджання та активації спрощує перевірку роботи програмного забезпечення під час контрольного прикладу. Додатково на екрані відображається стан Safe, що дозволяє користувачу одразу визначити, чи дозволене виконання команд активації.

Вибір пунктів меню та перехід між ними здійснюються за допомогою кнопок керування, підключених до головного пристрою. Для перемикання стану Safe використовується окрема кнопка, що дозволяє швидко дозволяти або

блокувати виконання команд активації без переходу до додаткових налаштувань. Такий підхід спрощує роботу користувача під час тестування та зменшує ймовірність випадкового виконання небажаної дії.

На першому етапі користувач переходить до режиму діагностики. У цьому режимі головний контролер починає відображати поточний стан периферійних вузлів у вигляді сітки 4×8. Кожна клітинка сітки відповідає одному виконавчому вузлу. Якщо від вузлів надходять коректні відповіді, програма оновлює діагностичну модель і відображає стани каналів відповідними кольорами. У нижній частині екрана виводиться службова інформація, зокрема кількість активних вузлів та параметри обміну. Відображення сітки станів виконавчих вузлів подано на рисунку 2.2.

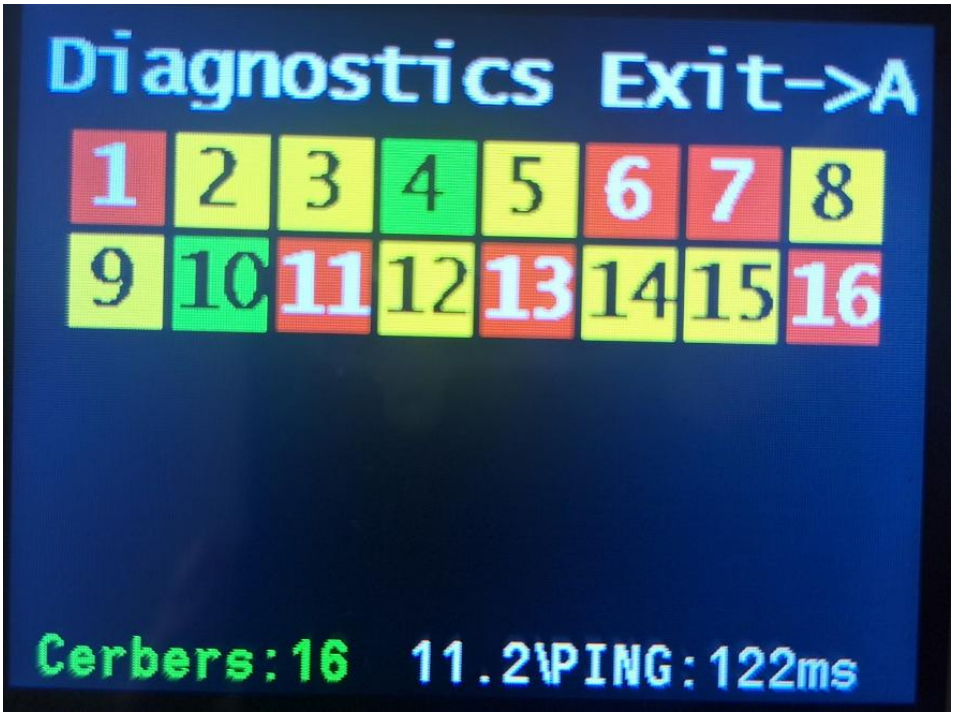


Рисунок 2.2 – Відображення сітки станів виконавчих вузлів

Колірне відображення станів використовується для швидкої візуальної оцінки працездатності системи. Завдяки цьому користувач може без додаткового перегляду службових параметрів визначити, які вузли готові до подальшої роботи, які перебувають у проміжному стані, а які потребують уваги через відсутність

готовності або наявність несправності. Такий підхід є зручним під час налагодження, оскільки дозволяє оперативно виявляти зміни станів після передавання керуючих команд.

У контрольному прикладі частина вузлів може перебувати у стані готовності до заряджання, частина — у стані готовності до активації, а окремі вузли можуть бути неготовими або перебувати в аварійному стані. Програма не вимагає, щоб усі 32 вузли мали однаковий стан. Кожен вузол обробляється окремо на основі отриманої від нього відповіді. Це дозволяє користувачу швидко оцінити реальний стан системи та визначити, які канали можуть брати участь у подальших діях.

Далі користувач може відкрити детальний екран одного з вузлів. На цьому екрані відображається розширена інформація про вибраний вузол: його номер, поточний стан, відповідь на останню команду, службові параметри, код помилки та додатковий параметр помилки. Якщо вузол перебуває у стані несправності, користувач може переглянути відповідну діагностичну інформацію без необхідності аналізувати весь масив даних вручну. Екран діагностики окремого виконавчого вузла зображено на рисунку 2.3.

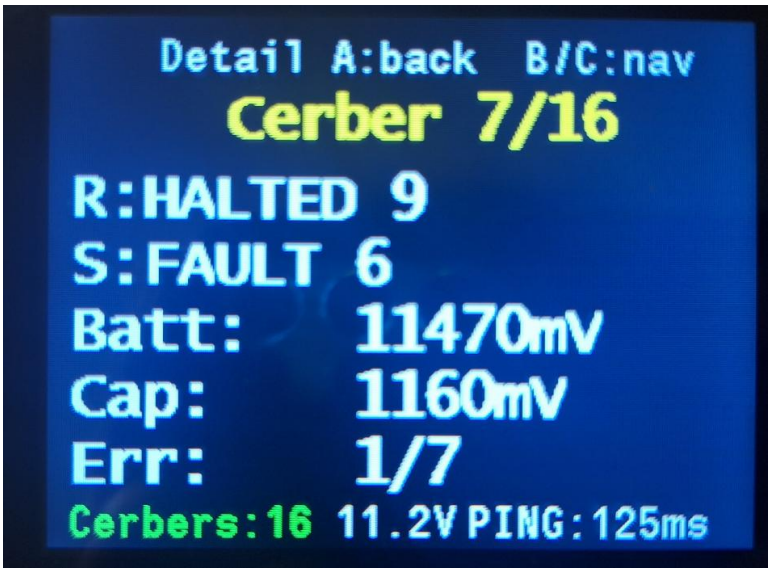


Рисунок 2.3 – Екран діагностики окремого виконавчого вузла

Використання детального екрана дозволяє не лише переглядати загальний стан вузла, а й перевіряти правильність обробки отриманих від нього даних. Під

час налагодження це дає змогу переконатися, що програма коректно розпізнає відповідь на команду, значення діагностичних параметрів і ознаки помилки. Таким чином, детальний екран є допоміжним засобом контролю правильності роботи протоколу обміну.

Наступним етапом контрольного прикладу є перевірка режиму заряджання. Користувач повертається до головного меню та обирає відповідний пункт. Перед виконанням дії програма аналізує поточний стан вузлів і визначає, чи є серед них вузли, готові до заряджання. Якщо таких вузлів немає, на дисплей виводиться повідомлення про неможливість виконання команди. Якщо хоча б один вузол перебуває у відповідному стані, програма формує команду заряджання та передає її до комунікаційної частини.

Після переходу до режиму заряджання команда встановлюється як поточна команда обміну. Це означає, що вона передається під час наступних циклів зв’язку з периферійними вузлами. У відповідь вузли мають змінити свій стан, а головний контролер після приймання нових пакетів оновлює діагностичну сітку. У результаті користувач може бачити, які вузли перейшли до процесу заряджання, а які залишились у попередньому стані. Повідомлення про перехід системи в режим заряджання показано на рисунку 2.4.

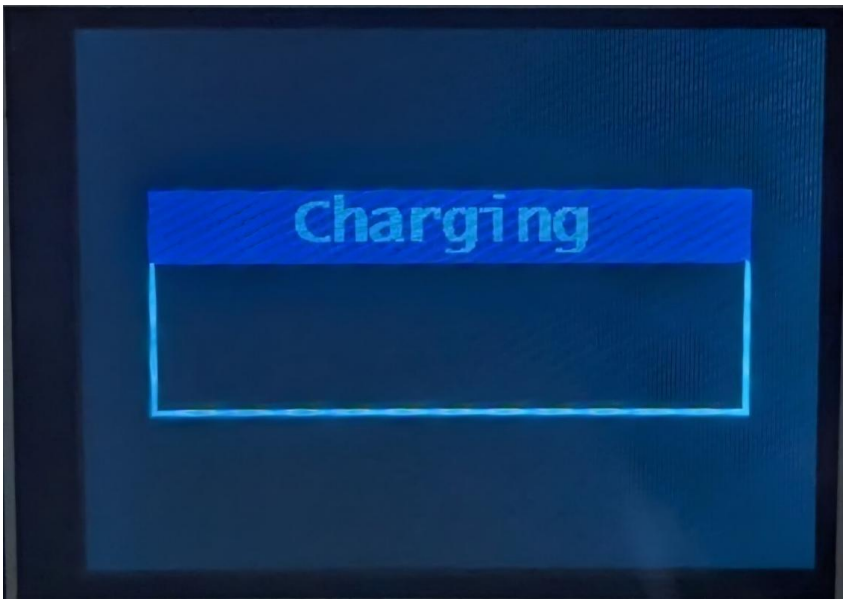


Рисунок 2.4 – Повідомлення про перехід системи в режим заряджання

Наступним етапом контрольного прикладу є перевірка ситуації, коли виконання режиму заряджання неможливе. Користувач повертається до головного меню та обирає режим заряджання, однак перед формуванням команди програма аналізує поточні стани периферійних вузлів. Якщо серед підключених вузлів немає жодного, що перебуває у стані готовності до заряджання, команда не передається до комунікаційної частини програми, а на дисплеї відображається відповідне повідомлення. Повідомлення про неможливість заряджання через відсутність готових вузлів подано на рисунку 2.5.

Рисунок 2.5 – Повідомлення про неможливість заряджання через відсутність готових вузлів

Після цього перевіряється режим групової активації. Користувач обирає відповідний пункт у головному меню, після чого програма аналізує поточний стан усіх периферійних вузлів і формує список каналів, які перебувають у стані готовності до активації. Якщо таких вузлів немає, команда групової активації не виконується, а користувачу виводиться повідомлення про неможливість виконання дії. Якщо готові вузли наявні, програма послідовно передає команди активації для

сформованого списку каналів із фіксованою паузою між окремими вузлами. Такий підхід дозволяє виконувати групову дію контрольовано, не створювати одночасного навантаження на систему та забезпечувати перевірку результату після оновлення відповідей від периферійних вузлів.

Під час перевірки групової активації важливо, що команда не застосовується до всіх вузлів одночасно. Програма попередньо аналізує поточні стани каналів і відбирає лише ті вузли, які перебувають у допустимому стані. Це дозволяє уникнути помилкового виконання команди для неготових або несправних вузлів та підтверджує правильність роботи алгоритму фільтрації каналів. Повідомлення про виконання групової активації представлено на рисунку 2.6.

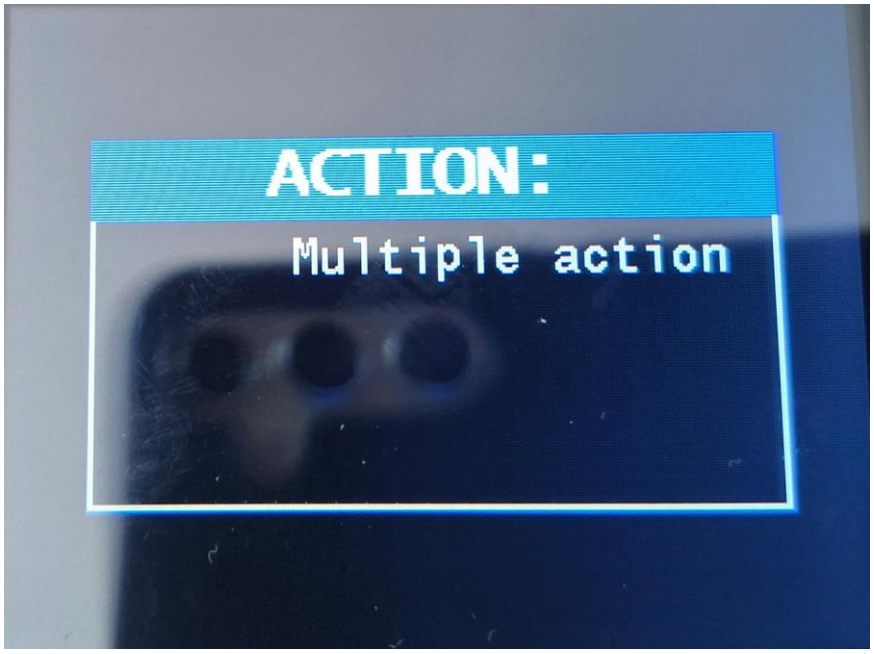


Рисунок 2.6 – Повідомлення про виконання групової активації

Окремо у контрольному прикладі перевіряється реакція системи на втрату зв’язку. Якщо протягом заданого часу від периферійних вузлів не надходять нові дані, програма встановлює ознаку втрати з’єднання. У такому випадку діагностична сітка очищується, а на дисплеї відображається повідомлення про відсутність зв’язку. Після відновлення приймання даних програма знову оновлює стан вузлів і повертається до штатного відображення діагностичної інформації.

Повідомлення про втрату зв’язку з периферійними вузлами відображено на рисунку 2.7.

Рисунок 2.7 – Повідомлення про втрату зв’язку з периферійними вузлами

Також перевіряється обробка аварійного стану окремого вузла. Якщо один із вузлів передає стан несправності, він відображається у діагностичній сітці відповідним кольором. На детальному екрані для такого вузла можуть бути переглянуті код помилки та додатковий параметр. При цьому несправний вузол не використовується для виконання команд активації, оскільки алгоритм вибирає тільки вузли, які перебувають у стані готовності.

Контрольний приклад підтверджує, що програмне забезпечення забезпечує основні сценарії роботи пристрою: запуск системи, перехід між екранами, діагностику 32 виконавчих вузлів, виконання режиму заряджання, одиночної та групової активації, блокування активації через Safe, обробку втрати зв’язку та відображення аварійних станів. Усі ці дії виконуються через графічний інтерфейс користувача та супроводжуються оновленням інформації на дисплеї.

Таким чином, контрольний приклад демонструє працездатність розробленої логіки програмного забезпечення та відповідність основним вимогам до мультиканального виконавчого пристрою. Програма забезпечує приймання команд від користувача, формування керуючих пакетів, обробку відповідей від

периферійних вузлів, контроль стану каналів і відображення результатів роботи на дисплеї.

2.7 Розробка допоміжного програмного емулятора

Для налагодження та перевірки роботи програмного забезпечення головного пристрою було розроблено допоміжний програмний емулятор периферійних виконавчих вузлів. Він призначений для імітації роботи підключених вузлів, формування діагностичних відповідей, перевірки протоколу обміну та тестування реакції STM32-пристрою на різні стани системи. Емулятор не є основною частиною цільового пристрою, а використовується як стендовий засіб налагодження.

Необхідність створення емулятора зумовлена тим, що перевірка головного контролера з повним набором фізичних периферійних вузлів не завжди є зручною на етапі розробки. Програмний засіб дозволяє швидко формувати тестові ситуації, змінювати стани окремих каналів, перевіряти обмін через послідовний інтерфейс та аналізувати реакцію основної програми без постійного використання повного апаратного стенда.

Емулятор реалізовано у вигляді настільного застосунку з графічним інтерфейсом користувача. Для його створення використано середовище Qt, мову програмування C++ та технологію Qt Widgets [4, 5]. Такий підхід дозволив реалізувати головне вікно, елементи керування, роботу з послідовним портом, візуальне представлення станів вузлів і журнал обміну даними. Для взаємодії з STM32-пристроєм використовується модуль роботи з послідовним портом [17].

Архітектура емулятора побудована з використанням об’єктно-орієнтованого підходу. Основне вікно програми відповідає за графічний інтерфейс, налаштування з’єднання, обробку команд, відображення журналу обміну та керування списком вузлів. Окремий клас використовується для опису одного периферійного вузла, приклад реалізації якого наведено у додатку З. У ньому зберігаються стан вузла, службові параметри та логіка формування відповіді. Такий

поділ дозволяє відокремити інтерфейс користувача від моделі даних.

Для кожного змодельованого вузла користувач може задавати поточний стан, ознаку несправності, параметри помилки, напругу конденсатора та інші службові значення, необхідні для формування діагностичної відповіді. У програмі передбачені стани, які відповідають логіці основної STM32-програми: неготовий стан, готовність до заряджання, процес заряджання, готовність до активації, активований стан та стан несправності.

Обмін даними між емулятором і STM32-пристроєм здійснюється через COM-порт [17], приклад реалізації роботи з COM-портом у Qt-емуляторі наведено у додатку И. Емулятор приймає службові пакети від головного контролера, перевіряє їхню структуру та контрольну суму, після чого формує відповіді від змодельованих вузлів. Підтримується обробка службової команди перевірки зв’язку, команди заряджання готових вузлів та команди активації окремого каналу.

Графічний інтерфейс емулятора містить елементи для вибору COM-порту, встановлення швидкості обміну, підключення або відключення від послідовного інтерфейсу, додавання та видалення вузлів, вибору пресетів конфігурації, групового встановлення станів і перегляду службового журналу. Для відображення вузлів використовується сітка карток, де кожна картка відповідає окремому виконавчому каналу. Приклад реалізації інтерфейсу керування станами вузлів наведено у додатку І.

Під час роботи емулятор формує відповіді для всіх активних вузлів. У разі отримання команди перевірки зв’язку він повертає поточні стани змодельованих вузлів. Після команди заряджання вузли, які перебувають у стані готовності до заряджання, переводяться у відповідний проміжний стан, а після завершення підготовки можуть перейти у стан готовності до активації. У разі отримання команди активації змінюється стан відповідного вузла.

Окрему роль у програмі виконує вікно UART-діагностики. Воно використовується для перегляду прийнятих і переданих пакетів, кількості байтів, оброблених команд, сформованих відповідей і помилок обміну. Наявність такого журналу спрощує налагодження, оскільки дозволяє контролювати не лише

зовнішню реакцію STM32-пристрою, а й сам процес передавання даних.

За допомогою програмного емулятора було перевірено основні сценарії роботи системи: періодичний обмін з периферійними вузлами, відображення станів у режимі діагностики, перехід вузлів у режим заряджання, одиночну та групову активацію, відображення аварійних станів і реакцію на некоректні або відсутні відповіді. Також емулятор дозволяє перевіряти втрату зв’язку шляхом розриву послідовного з’єднання або вимкнення активних змодельованих вузлів.

Важливою перевагою емулятора є можливість швидкої зміни станів вузлів без перепідключення фізичного обладнання. Користувач може створити потрібну конфігурацію, наприклад частину вузлів перевести у стан готовності до заряджання, частину — у стан готовності до активації, а окремі вузли — у стан несправності. Після цього можна перевірити, як головний пристрій відображає ці стани на дисплеї та які команди формує у відповідь на дії користувача.

У межах дипломної роботи допоміжний емулятор розглядається як засіб тестування та налагодження, а не як обов’язкова частина кінцевого пристрою. Його використання дозволило перевірити основні алгоритми роботи STM32-програми ще до повного підключення всіх фізичних периферійних вузлів, зменшити кількість помилок на етапі стендової перевірки та прискорити процес налагодження програмного забезпечення.

Головне вікно програмного емулятора містить засоби вибору COM-порту, встановлення швидкості обміну, підключення до послідовного інтерфейсу, застосування готових пресетів і керування кількістю активних вузлів. Основна частина інтерфейсу представлена у вигляді карток периферійних вузлів, у яких відображається поточний стан, напруга конденсатора та службові ознаки несправності. Під час перевірки головного вікна емулятора основна увага приділялася коректності відображення змодельованих вузлів і можливості швидкої зміни їхніх параметрів. Зокрема перевірялося, чи правильно оновлюються стани вузлів після застосування групових налаштувань, чи відповідає візуальне позначення поточному стану каналу та чи не виникають помилки під час зміни кількості активних вузлів.

Під час налагодження через головне вікно емулятора виконувалася підготовка тестових конфігурацій для подальшої перевірки STM32-програми. Користувач міг змінювати кількість активних вузлів, задавати їхні стани та одразу спостерігати, як ці зміни впливають на відповіді, що формуються для головного пристрою. Це дозволяло швидко відтворювати потрібні умови перевірки без зміни фізичного підключення периферійних вузлів. Приклад головного вікна емулятора наведено на рисунку 2.8.

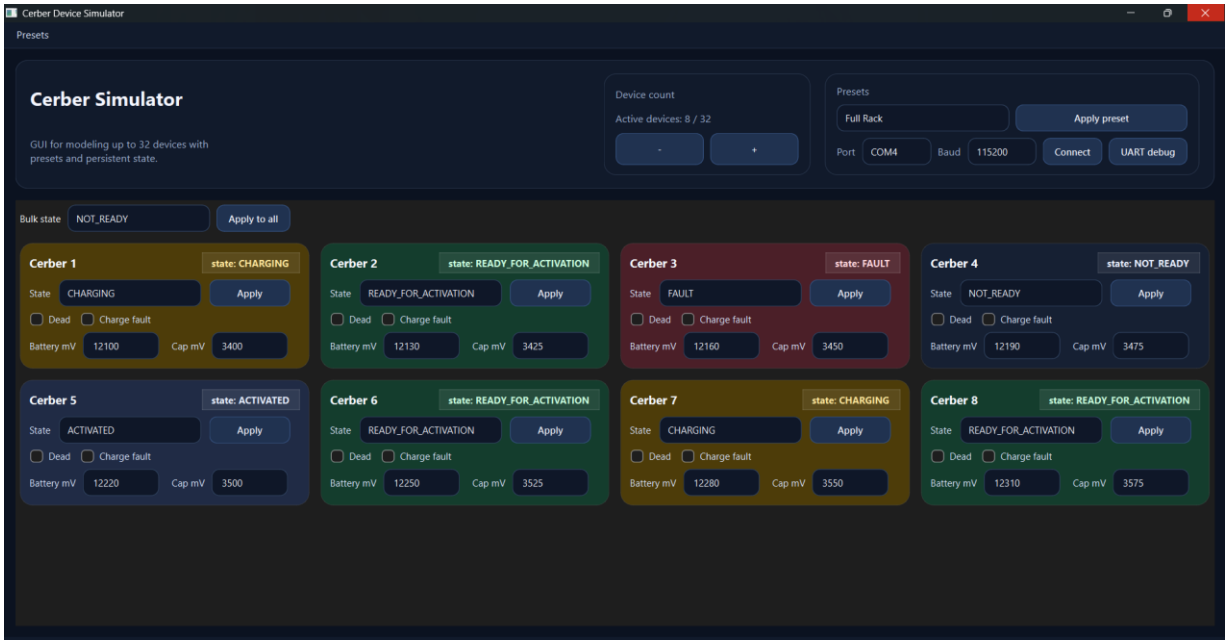


Рисунок 2.8 – Головне вікно програмного емулятора периферійних вузлів

Як видно з рисунка 2.8, кожен змодельований вузол має окрему картку з власним станом і параметрами. Колір картки відповідає поточному стану вузла, що дозволяє швидко оцінити загальну конфігурацію системи під час тестування. Користувач може змінювати стани вузлів вручну або застосовувати групові налаштування, що спрощує підготовку тестових сценаріїв для перевірки діагностики, заряджання та активації каналів.

Для підвищення зручності користування в програмному емуляторі передбачено механізм пресетів, який дозволяє швидко застосовувати типові конфігурації набору периферійних вузлів. Це дає змогу скоротити час підготовки тестових сценаріїв і прискорити перевірку роботи основного пристрою в різних

режимах.

Під час налагодження пресети використовувалися для швидкого відтворення однакових умов перевірки без ручного налаштування кожного вузла окремо. Наприклад, за допомогою такого механізму можна було сформувати конфігурацію з усіма неготовими вузлами, змішаний набір станів або ситуацію, коли частина каналів перебуває у стані несправності. Це підвищує повторюваність тестування та дозволяє порівнювати реакцію головного пристрою за однакових початкових умов. Приклад вибору пресету в інтерфейсі емулятора наведено на рисунку 2.9.

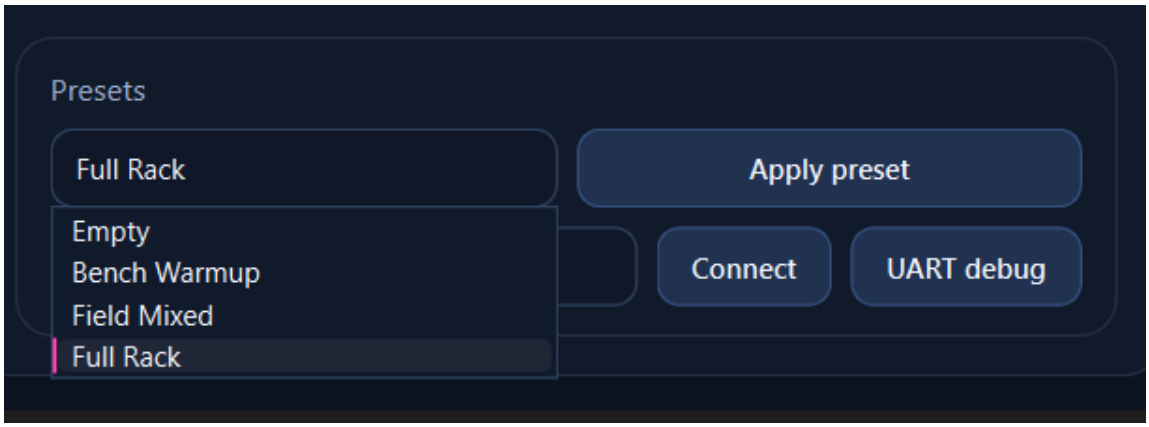


Рисунок 2.9 – Вибір пресетів конфігурації в програмному емуляторі

Як видно з рисунка 2.9, користувач може обрати наперед підготовлену конфігурацію зі списку та застосувати її до поточного набору змодельованих вузлів. Такий підхід спрощує відтворення типових сценаріїв тестування, зокрема початкового стану системи, робочих конфігурацій та змішаних наборів станів каналів. Використання пресетів підвищує зручність роботи з емулятором і забезпечує швидкий перехід між різними режимами перевірки.

Таким чином, розроблений Qt-емулятор забезпечує імітацію роботи до 32 периферійних виконавчих вузлів, формування діагностичних відповідей, перевірку команд заряджання та активації, а також аналіз службового обміну через COM-порт. Його застосування підвищило зручність тестування, дало змогу моделювати різні стани системи та забезпечило ефективне налагодження програмного забезпечення головного пристрою.

3 СПЕЦІАЛЬНИЙ РОЗДІЛ

До складу програмного комплексу входить вбудоване програмне забезпечення головного пристрою на базі STM32 та допоміжний програмний емулятор периферійних вузлів, який використовується для стендового тестування й налагодження [1, 2].

Основне вбудоване програмне забезпечення забезпечує керування мультиканальним виконавчим пристроєм, обмін даними з периферійними вузлами, діагностику стану каналів, обробку команд користувача та відображення інформації на TFT-дисплеї. Допоміжний емулятор дозволяє перевіряти роботу основної програми без повного апаратного стенда, імітуючи відповіді периферійних вузлів через послідовний інтерфейс [17].

3.1 Інструкція з інсталяції програмного забезпечення

Інсталяція програмного забезпечення передбачає підготовку середовища розробки, завантаження вбудованої програми у мікроконтролер STM32 та, за потреби, запуск допоміжного програмного емулятора на персональному комп’ютері. Перед початком інсталяції необхідно переконатися, що наявні всі необхідні апаратні й програмні засоби.

Для роботи з вбудованим програмним забезпеченням необхідні: персональний комп’ютер, середовище STM32CubeIDE, програматор або налагоджувач ST-Link, плата з мікроконтролером STM32H750VBT6, з’єднувальні кабелі, джерело живлення та, за потреби, периферійні виконавчі вузли або програмний емулятор [1, 3]. Для роботи допоміжного емулятора потрібна операційна система Windows та зібраний виконуваний файл програми. У разі необхідності внесення змін або повторної збірки може використовуватися середовище Qt Creator, а також доступний COM-порт або USB-UART перетворювач [4, 17].

Першим етапом є підготовка проєкту STM32. Для цього необхідно відкрити

середовище STM32CubeIDE та імпортувати проєкт вбудованого програмного забезпечення. Імпорт виконується через меню відкриття існуючого проєкту [12, 18]. Після імпорту потрібно переконатися, що структура проєкту відображається коректно, а основні файли програми доступні в каталогах `Core/Src` та `Core/Inc`.

Після відкриття проєкту необхідно перевірити налаштування цільового мікроконтролера, конфігурацію тактування, параметри периферії та наявність згенерованих файлів ініціалізації. Особливу увагу потрібно звернути на налаштування UART, DMA, SPI, GPIO та FreeRTOS, оскільки ці компоненти забезпечують обмін даними, роботу дисплея, обробку кнопок і виконання задач програми [8, 10].

Наступним етапом є компіляція проєкту. Для цього в STM32CubeIDE потрібно виконати команду збирання. Якщо всі залежності підключені правильно, проєкт має скомпілюватися без помилок. У разі появи повідомлень про помилки необхідно перевірити шляхи до файлів, налаштування компілятора, наявність бібліотек HAL, файлів FreeRTOS та коректність конфігурації проєкту.

Після успішної компіляції виконується завантаження програми у мікроконтролер. Для цього плата STM32 підключається до персонального комп’ютера через ST-Link. У середовищі STM32CubeIDE обирається режим прошивання або налагодження. Після початку завантаження середовище записує скомпільований код у пам’ять мікроконтролера. Після завершення прошивання пристрій може бути перезапущений для переходу до штатного режиму роботи.

Після завантаження програми необхідно перевірити базову працездатність пристрою. При подачі живлення має вмикатися дисплей, відображатися початковий екран або головне меню, а кнопки керування повинні реагувати на натискання. Якщо дисплей не вмикається або інтерфейс не відображається, потрібно перевірити живлення, підключення дисплея, сигнали SPI та GPIO-лінії керування екраном.

Для перевірки обміну даними з периферійними вузлами необхідно підключити фізичні вузли або використати програмний емулятор. Якщо використовується фізичний стенд, потрібно перевірити правильність підключення

лінії RS-485, живлення вузлів і відповідність параметрів обміну. Якщо використовується емулятор, потрібно підключити STM32-пристрій до ПК через відповідний послідовний інтерфейс.

Інсталяція допоміжного Qt-емулятора полягає у запуску зібраного виконуваного файлу або відкритті проєкту в середовищі Qt Creator. Якщо використовується готовий виконуваний файл, достатньо запустити програму на ПК. Якщо необхідно зібрати програму з вихідного коду, потрібно відкрити Qt-проєкт, перевірити наявність модулів Qt Widgets і Serial Port, після чого виконати збірку.

Після запуску емулятора необхідно вибрати COM-порт, встановити швидкість обміну та натиснути кнопку підключення. Далі користувач може створити потрібну кількість змодельованих вузлів, задати їхні стани або застосувати готовий пресет конфігурації. Після цього емулятор готовий до обміну з STM32-пристроєм і може використовуватися для перевірки діагностики, заряджання, активації каналів та обробки несправностей.

Після завершення інсталяції програмного забезпечення необхідно виконати коротку перевірку: переконатися, що STM32-пристрій запускається, дисплей відображає меню, кнопки працюють, емулятор або фізичні вузли відповідають на команди, а в режимі діагностики оновлюється стан виконавчих каналів. Якщо всі ці умови виконуються, програмний комплекс готовий до подальшого тестування та експлуатації.

3.2 Інструкція з використання тестових наборів

Тестові набори використовуються для перевірки працездатності програмного комплексу в різних режимах роботи. Вони дають змогу моделювати типові ситуації, які можуть виникати під час експлуатації мультиканального виконавчого пристрою: наявність готових вузлів, процес заряджання, готовність до активації, несправність окремого вузла, відсутність відповідей або змішаний стан системи.

Основним засобом формування тестових наборів є допоміжний програмний емулятор. Він дозволяє створювати до 32 змодельованих периферійних вузлів і задавати для кожного з них окремий стан. Для пришвидшення підготовки тестування в емуляторі передбачено пресети, тобто наперед підготовлені конфігурації вузлів. Пресети дозволяють швидко перейти до потрібного тестового сценарію без ручного налаштування кожного каналу.

Перед початком тестування потрібно запустити STM32-пристрій і допоміжний емулятор. У емуляторі необхідно вибрати COM-порт, встановити швидкість обміну, підключитися до послідовного інтерфейсу та створити потрібну кількість вузлів. Після цього можна або вручну задати стани вузлів, або скористатися одним із готових пресетів.

Перший тестовий набір призначений для перевірки початкового стану системи. У цьому випадку всі або більшість вузлів переводяться у неготовий стан. Після переходу STM32-пристрою в режим діагностики користувач повинен переконатися, що сітка каналів відображає відповідні стани, а команди активації не виконуються для вузлів, які не готові до роботи.

Другий тестовий набір використовується для перевірки режиму заряджання. Для цього частина вузлів переводиться у стан готовності до заряджання. Після вибору режиму заряджання на STM32-пристрої програма повинна сформувати відповідну команду, а емулятор — змінити стан готових вузлів. У діагностичній сітці має відображатися перехід вузлів у процес заряджання, а після завершення підготовки — у стан готовності до активації.

Третій тестовий набір призначений для перевірки одиночної активації. Для цього один або декілька вузлів переводяться у стан готовності до активації. Після вибору відповідного режиму в меню STM32-пристрій повинен визначити перший доступний готовий вузол і сформувати для нього команду активації. Після обробки команди стан цього вузла має змінитися, а оновлена інформація повинна відобразитися на екрані діагностики.

Четвертий тестовий набір використовується для перевірки групової активації. У цьому випадку декілька вузлів переводяться у стан готовності до

активації. Після вибору режиму групової активації програма повинна сформувати список готових вузлів і послідовно передати команди для кожного з них. У результаті стани відповідних вузлів мають змінюватися поступово, відповідно до заданого інтервалу між командами.

П’ятий тестовий набір призначений для перевірки аварійних станів. Для цього один або декілька вузлів переводяться у стан несправності. У режимі діагностики такі вузли повинні відображатися окремим кольором, а на детальному екрані мають бути доступні службові параметри помилки. При цьому такі вузли не повинні брати участь у режимах активації, оскільки алгоритм працює лише з вузлами, які перебувають у стані готовності.

Шостий тестовий набір використовується для перевірки захисного режиму Safe. Для цього в системі мають бути вузли, готові до активації, але режим Safe повинен бути увімкнений. У такому випадку при спробі виконати одиночну або групову активацію команда не повинна передаватися до вузлів, а на дисплеї має з’явитися повідомлення про блокування дії. Після вимкнення Safe активація повинна виконуватися відповідно до логіки вибраного режиму.

Окремо перевіряється ситуація втрати зв’язку. Для цього можна тимчасово відключити COM-порт, розірвати з’єднання або зупинити формування відповідей емулятором. Якщо STM32-пристрій протягом заданого часу не отримує дані від периферійних вузлів, у режимі діагностики має з’явитися повідомлення про відсутність зв’язку, а сітка каналів повинна очиститися або перейти у стан, що показує недоступність даних.

Під час використання тестових наборів потрібно контролювати не лише відображення станів на дисплеї, а й журнал обміну в емуляторі. У ньому можна переглядати прийняті й передані пакети, кількість оброблених команд, кількість сформованих відповідей і можливі помилки обміну. Це дозволяє швидше визначити, на якому етапі виникає помилка: у формуванні команди, передаванні даних, обробці відповіді або відображенні стану.

Результати проходження тестових наборів доцільно фіксувати у вигляді таблиці. Для кожного тесту можна зазначати назву сценарію, початковий стан

вузлів, дію користувача, очікуваний результат і фактичний результат. Такий підхід дозволяє систематизувати перевірку програмного забезпечення та підтвердити правильність роботи основних режимів.

Застосування тестових наборів дозволяє перевірити роботу програмного комплексу в типових і нестандартних ситуаціях. Завдяки цьому можна оцінити коректність діагностики, обміну даними, режиму заряджання, одиночної та групової активації, захисного блокування Safe, а також реакції на втрату зв’язку й аварійні стани окремих вузлів.

3.3 Інструкція з експлуатації програмного комплексу

Експлуатація програмного комплексу передбачає використання головного пристрою на базі STM32, підключених периферійних виконавчих вузлів або програмного емулятора, а також графічного інтерфейсу на дисплеї пристрою. Перед початком роботи необхідно переконатися, що всі компоненти системи підключені правильно, живлення подано, а лінія обміну даними перебуває у справному стані.

Після подачі живлення головний пристрій виконує початкову ініціалізацію. У цей час налаштовуються апаратні модулі, запускається приймання даних через UART, ініціалізується дисплей і створюються задачі операційної системи FreeRTOS. Після завершення цього етапу на екрані з’являється головне меню або початковий екран. Якщо дисплей не вмикається або меню не відображається, необхідно перевірити живлення пристрою та підключення дисплея.

Керування програмним комплексом здійснюється за допомогою кнопок. Кнопки використовуються для переміщення між пунктами меню, вибору режиму роботи, перемикання захисного режиму Safe, переходу до діагностики, відкриття детального екрана вузла та повернення до попереднього екрана. Перед виконанням будь-якої керуючої дії користувач повинен переконатися, що на дисплеї відображається актуальна інформація про стан системи.

Основним екраном для контролю системи є режим діагностики. У цьому

режимі на дисплеї відображається сітка 4×8, що відповідає 32 виконавчим вузлам. Кожна клітинка містить номер вузла та має колір, який відповідає його стану. За допомогою цієї сітки користувач може швидко визначити, які вузли не готові, які перебувають у процесі заряджання, які готові до активації, які вже активовані, а які мають несправність.

Для перегляду розширеної інформації про конкретний вузол використовується детальний екран. На ньому відображається номер вузла, поточний стан, відповідь на останню команду, напруга конденсатора та службові параметри помилки. Цей екран доцільно використовувати під час налагодження, перевірки окремого каналу або аналізу несправності.

Перед виконанням активації необхідно перевірити стан Safe. Якщо режим Safe увімкнений, одиночна та групова активація заблоковані. Це потрібно для запобігання випадковому виконанню керуючих дій. Якщо користувач намагається виконати активацію при увімкненому Safe, програма виводить відповідне повідомлення, а команда не передається до периферійних вузлів.

Для виконання одиночної активації користувач обирає відповідний пункт у головному меню. Після цього програма перевіряє актуальний стан вузлів і визначає перший вузол, який перебуває у стані готовності до активації. Якщо такий вузол знайдено, програма формує команду активації та передає її через інтерфейс обміну. Якщо готових вузлів немає, на дисплеї відображається повідомлення про неможливість виконання дії.

Для виконання групової активації користувач обирає відповідний пункт меню. Програма формує список усіх вузлів, які готові до активації, після чого послідовно передає команди для кожного з них. Команди виконуються з паузою між окремими вузлами, що забезпечує контрольоване виконання групової дії. Після завершення операції користувач може перейти до діагностики та перевірити оновлені стани вузлів.

Режим заряджання використовується для підготовки вузлів до подальшої роботи. Перед виконанням цього режиму програма перевіряє, чи є вузли, готові до заряджання. Якщо таких вузлів немає, користувач отримує повідомлення про

помилку. Якщо готові вузли є, команда заряджання встановлюється як поточна та передається під час періодичного обміну.

Під час експлуатації потрібно звертати увагу на повідомлення системи. Повідомлення можуть інформувати про виконання дії, неможливість виконання команди, увімкнений режим Safe, відсутність готових вузлів, несправність окремого вузла або втрату зв’язку. У разі появи повідомлення про відсутність зв’язку необхідно перевірити підключення лінії обміну, живлення периферійних вузлів та правильність налаштування інтерфейсу.

Якщо використовується програмний емулятор, порядок експлуатації доповнюється запуском емулятора на ПК. У ньому потрібно вибрати COM-порт, встановити швидкість обміну, підключитися до STM32-пристрою та сформувати потрібну конфігурацію вузлів. Після цього головний пристрій сприйматиме емулятор як набір периферійних вузлів і відображатиме їхні стани в режимі діагностики.

У разі виникнення несправності окремого вузла він відображається у відповідному стані на екрані діагностики. Для уточнення причини можна перейти до детального екрана й переглянути службові параметри помилки. Несправні вузли не беруть участі в режимах активації, оскільки алгоритм виконує керуючі дії лише для вузлів, які перебувають у стані готовності.

Після завершення роботи з програмним комплексом необхідно перевести систему у безпечний стан, увімкнути режим Safe, завершити активні операції, за потреби відключити емулятор або фізичні периферійні вузли та вимкнути живлення пристрою. Такий порядок завершення роботи дозволяє уникнути випадкового виконання команд і забезпечує коректне завершення експлуатації системи.

Дотримання наведеної інструкції забезпечує правильне використання програмного комплексу, дозволяє виконувати діагностику стану 32 виконавчих вузлів, проводити заряджання, одиночну та групову активацію, контролювати службові параметри та своєчасно виявляти втрату зв’язку або несправність окремих каналів.

4 ЕКОНОМІЧНИЙ РОЗДІЛ

Метою економічної частини кваліфікаційної роботи є проведення економічних розрахунків, необхідних для визначення витрат на розробку програмного забезпечення для мультиканального виконавчого пристрою високої надійності на базі мікроконтролера STM32. Отримані результати дають змогу оцінити економічну доцільність виконання розробки, її подальшого розвитку та можливого впровадження.

Об’єктом розрахунку є програмне забезпечення вбудованої системи керування, призначене для організації роботи мультиканального виконавчого пристрою. Розроблюване програмне забезпечення забезпечує обробку команд керування, контроль стану виконавчих каналів, відображення діагностичної інформації, а також надійну взаємодію між апаратними вузлами пристрою.

Розрахунок вартості розробки виконується поетапно. Спочатку визначається технологічний процес створення програмного забезпечення та трудомісткість окремих операцій. Після цього розраховуються витрати на оплату праці основного й допоміжного персоналу, відрахування на соціальні заходи, витрати на електроенергію, амортизаційні відрахування та накладні витрати. На основі отриманих значень складається кошторис, визначається собівартість і ціна робіт, а також оцінюється економічна ефективність розробки та термін її окупності.

4.1 Визначення стадій технологічного процесу та загальної тривалості проведення НДР

У цьому підрозділі розглянуто основні етапи технологічного процесу розробки програмного забезпечення для мультиканального виконавчого пристрою високої надійності на базі мікроконтролера STM32.

Для визначення загальної тривалості проведення робіт дані щодо витрат часу за окремими операціями технологічного процесу зведено у таблиці 4.1.

Таблиця 4.1 – Середній час виконання робіт по обслуговуванню та стадії (операції) технологічного процесу

№ п/п	Назва операції (стадії)	Виконавець	Середній час виконання операції, год.
1	Аналіз вимог до програмного забезпечення пристрою	Кер. проєкту Pm	5
		Інженер (l2)	5
2	Розробка технічного завдання	Кер. проєкту (Pm)	6
		Інженер (l2)	4
3	Аналіз платформи STM32 та периферії	Інженер (l2)	8
		Інженер (l1)	5
4	Розробка структури ПЗ	Інженер (l2)	10
5	Розробка модуля керування виконавчими каналами	Інженер (l1)	20
6	Розробка модуля команд та обміну даними	Інженер (l1)	8
		Інженер (l2)	15
7	Розробка модуля діагностики стану каналів	Інженер (l1)	16
8	Розробка людино-машинного інтерфейсу	Інженер (l1)	14
9	Інтеграція програмних модулів	Інженер (l2)	10
		Інженер (l1)	5
10	Тестування та налагодження ПЗ	Тестувальник	12
		Інженер (l1)	5
11	Підготовка програмної документації	Інженер (l1)	6
12	Управління проєктом та контроль виконання робіт	Кер. проєкту (Pm)	12
Разом			166

Сумарний час виконання операцій технологічного процесу становить 166 годин.

4.2 Визначення витрат на оплату праці та відрахувань на соціальні заходи

У даному підрозділі проводиться аналіз і розрахунок витрат, пов'язаних з оплатою праці та відрахуваннями на соціальні заходи, що необхідні для розробки програмного забезпечення для мультиканального виконавчого пристрою високої надійності на базі мікроконтролера МК STM32.

Розмір заробітної плати залежить від складності та умов виконуваної роботи, професійно-ділових якостей працівника, результатів його праці та діяльності підприємства.

Основна заробітна плата розраховується за формулою:

$$З_{осн.} = Т_с * К_г \quad (4.1)$$

де: T_c – тарифна ставка, грн. (приймаємо для керівника проєкту (Pm) – 450 грн./год, інженера (l2) – 272 грн./год.), інженера (l1) – 113 грн./год., тестувальник – 100 грн./год.; $K_г$ – кількість відпрацьованих годин.

Отже, основна заробітна плата для:

Керівника проєкту (Pm) $З_{осн}$= 23 * 450 = 10 350 грн.

Інженера (l2) $З_{осн}$= 52 * 272 = 14 144 грн.

Інженера (l1) $З_{осн}$= 79 * 113 = 8927 грн.

Тестувальник $З_{осн}$= 12 * 100 = 1200 грн.

Сумарна основна заробітна плата становить

$$З_{осн} = 10\,350 + 14\,144 + 8\,927 + 1\,200 = 34\,621 \text{ грн.}$$

Додаткова заробітна плата становить 10 – 15 % від суми основної заробітної плати.

$$З_{дод.} = З_{осн.} \cdot К_{допл.} \quad (4.2)$$

де: $К_{допл.}$ – коефіцієнт додаткових виплат працівникам.

Отже додаткова заробітна плата по категоріях працівників становить:

Керівника проєкту $З_{дод}$= 10 350 · 0,1 = 1 035 грн.

Інженера (l2) $З_{дод}$= 14 144 · 0,1 = 1 414 грн.

Інженера (l1) $З_{дод}$= 8 927 · 0,1 = 893 грн.

Тестувальник $З_{дод}$= 1 200 · 0,1 = 120 грн.

Загальна додаткова заробітна плата становить:

$$З_{дод}= 1\,035 + 1\,414 + 893 + 120 = 3\,462 \text{ грн.}$$

Звідси загальні витрати на оплату праці ($B_{о.п.}$) визначаються за формулою:

$$В_{о.п.} = З_{осн.} + З_{дод.} \quad (4.3)$$

$$В_{о.п}= 34\,621 + 3\,462 = 38\,083 \text{ грн.}$$

Єдиний соціальний внесок (ЄСВ – 22%) визначається за формулою:

$$В_{єсв} = Воп * 0{,}22 \quad (4.4)$$

$$В_{єсв} = 38\,083 \cdot 0{,}22 = 8\,378 \text{ грн.}$$

Проведені розрахунки витрат на оплату праці наведено у таблиці 4.2.

Таблиця 4.2 – Зведені розрахунки витрат на оплату праці

№ п/п	Категорія працівників	Основна заробітна плата, грн.			Додаткова заробітна плата, грн.	ЄСВ, грн.	Всього витрати на оплату праці, грн. 6 = 3+4+5
		Тарифна ставка, грн.	К-сть годин	Фактично нарах. зарплати, грн.			
		1	2	3	4	5	6
1	Кер. проєкту (Pm	450	23	10 350	1 035	2 505	13 890
2	Інженера (l2)	272	52	14 144	1 414	3 423	18 981
3	Інженера (l1)	113	79	8 927	893	2 160	11 980
4	Тестувальник	100	12	1 200	120	290	1 610
Разом				34 621	3 462	8 378	46 461

Отже, загальні витрати на оплату праці становлять 46 461 грн.

4.3 Розрахунок витрат на електроенергію

Розрахуємо вартість електроенергії. Затрати на електроенергію 1-ці обладнання визначаються за формулою:

$$З_в = W * T * S \quad (4.5)$$

де: W – необхідна потужність, кВт; Т – кількість годин роботи обладнання; S – вартість кіловат-години електроенергії (приймаємо 15,94 грн).

У процесі розробки програмного забезпечення використовується один персональний комп'ютер.

Витрати на електроенергію для цього комп'ютера обчислимо окремо, взявши за основу, що час роботи обладнання обчислюється в залежності від виконуваних робіт (згідно табл. 4.1) і споживані потужності наступні: комп'ютер – 0,82 кВт/год.

Споживання відлагоджувальної плати STM32 та допоміжних периферійних пристроїв є незначним порівняно зі споживанням персонального комп'ютера, тому в межах даного розрахунку воно враховується у загальних витратах на електроенергію.

$$З_{ек}= 0,82*166*15,94= 2170 \text{ грн.}$$

Витрати на електроенергію становлять 2170 грн.

4.4 Розрахунок суми амортизаційних відрахувань розробки програмного забезпечення для мультиканального виконавчого пристрою високої надійності на базі мікроконтролера МК STM32

Характерною особливістю застосування основних фондів у процесі виробництва є їх відновлення. Для відновлення засобів праці у натуральному виразі необхідне їх відшкодування у вартісній формі, яке здійснюється шляхом амортизації.

Амортизація – це процес перенесення вартості основних фондів на вартість новоствореної продукції з метою їх повного відновлення.

Комп'ютери та оргтехніка належать до четвертої групи основних фондів [19].

Амортизація на них нараховується лише у випадку, якщо мінімально допустимі строки їх корисного використання 2 роки [19]. Для визначення амортизаційних відрахувань застосовуємо формулу:

$$А = \frac{Б_В * Н_А}{100\%} * T, \quad (4.6)$$

де: А – амортизаційні відрахування за звітний період, грн.;

$Б_В$ – балансова вартість групи основних фондів на початок звітного періоду, грн.;

$Н_А$ – норма амортизації, 0,04 %.

Оскільки для написання програми та її тестування використовується один ПК, вартістю 45000,00 грн., то сума амортизаційних відрахувань становитиме:

$$А = \frac{45\,000{,}00 * 0{,}04}{150} * 166 = 1\,992 грн.$$

4.5 Обчислення накладних витрат

Накладні витрати пов’язані з обслуговуванням виробництва, утриманням апарату управління підприємства (фірми) та створення необхідних умов праці. В залежності від організаційно-правової форми діяльності господарюючого суб’єкта, накладні витрати можуть становити 20–60 % від суми основної та додаткової заробітної плати працівників.

$$Н_в = В_{о.п.} * \ 0{,}2..0{,}6 \quad (4.7)$$

де: $Н_В$ – накладні витрати.

$$Н_В = 38\,083 * 0{,}4 = 15\,233 грн.$$

4.6 Складання кошторису витрат та визначення собівартості розробки програмного забезпечення для мультиканального виконавчого пристрою високої надійності на базі мікроконтролера МК STM32

Для складення кошторису витрат та визначення собівартості, результати проведених вище розрахунків зведемо у таблиці 4.3.

Таблиця 4.3 - Кошторис витрат розробки програмного забезпечення для мультиканального виконавчого пристрою високої надійності на базі мікроконтролера МК STM32

№	Зміст витрат	Сума, грн.	В % до загальної суми
1.	Витрати на оплату праці	46 461	71
2.	Витрати на електроенергію	2 170	3
3.	Амортизаційні відрахування	1 992	3
4.	Накладні витрати	15 233	23
5.	Собівартість	65 856	100

Собівартість ($С_В$) НДР розрахуємо за формулою:

$$Св = Во.п + Вс.з + Зе + А + Нв \quad (4.8)$$

Отже, собівартість дорівнює $С_В$= 65 856 грн.

4.7 Розрахунок ціни розробки програмного забезпечення для мультиканального виконавчого пристрою високої надійності на базі мікроконтролера МК STM32

Розрахунок ціни науково-дослідної роботи включає в себе урахування різноманітних факторів, таких як рівень рентабельності, собівартість та податкова ставка.

Ціну робіт можна визначити за формулою:

$$Ц = Св * (1+Р_{рен}) * (1+ПДВ), \quad (4.9)$$

де: Св – собівартість; $Р_{рен.}$ – рівень рентабельності; ПДВ – ставка податку на додану вартість.

$$Ц = 65\,856*(1+0,3)*(1+0,2)= 102\,735 \text{ грн.}$$

4.8 Визначення економічної ефективності і терміну окупності капітальних вкладень

Ефективність виробництва – це узагальнене і повне відображення кінцевих результатів використання робочої сили, засобів та предметів праці на підприємстві за певний проміжок часу.

Для визначення ефективності продукту розраховують чисту теперішню вартість (ЧТВ) і термін окупності (Ток).

$$ЧТВ = -Св + \sum_{i=1}^{t} \frac{Г_П}{(1+i)^t}, \quad (4.10)$$

де: Св – собівартість розробки; $Г_п$ – грошовий потік за t – ий рік; t – відповідний рік проєкту; і – величина дисконтної ставки (10…15%).

$$ЧТВ = -65\,856 + \frac{36\,879}{(1+0,1)^1} + \frac{36\,879}{(1+0,1)^2} + \frac{36\,879}{(1+0,1)^3} = 25\,857 \text{ грн}$$

Якщо ЧТВ ≥ 0, то проєкт може бути рекомендований до впровадження.

Термін окупності визначається за формулою:

$$T_{ОК} = T_{ПВ} + \frac{Н_В}{Г_{ПР}} \quad (4.11)$$

де: $Т_{ПВ}$ – період до повного відшкодування витрат, років; $Н_В$ – невідшкодовані витрати на початок року, грн.; $Г_{ПР}$ – грошовий потік на початок року, грн.

$$Т_{ок} = 2 + \frac{1\,851}{36\,879} = 2,1 \text{ р.}$$

Всі дані внесемо в зведену таблицю 4.4.

Таблиця 4.4 – Техніко-економічні показники розробки програмного забезпечення для мультиканального виконавчого пристрою високої надійності на базі STM32.

№ п/п	Показник	Значення
1.	Собівартість, грн.	65 856
2.	Плановий прибуток або грошовий потік, грн.	36 879
3.	Ціна, грн.	102 735
4.	Чиста теперішня вартість, грн.	25 857
5.	Термін окупності, рік	2,1

Прибутковість проєкту та термін окупності свідчать про його фінансову ефективність та здатність повернути капітальні вкладення протягом 2,1 року. Отже, на основі отриманих показників можна зробити висновок, що розробка програмного забезпечення для мультиканального виконавчого пристрою високої надійності на базі STM32 є доцільною з економічної точки зору.

5 ОХОРОНА ПРАЦІ, ТЕХНІКА БЕЗПЕКИ ТА ЕКОЛОГІЧНІ ВИМОГИ

5.1 Дії роботодавця при отриманні повідомлення про нещасний випадок

Нещасний випадок на виробництві — це подія, яка сталася з працівником під час виконання ним трудових обов'язків або перебування на робочому місці та призвела до травми, погіршення стану здоров'я або смерті [20]. У разі виникнення такої події важливим є швидке реагування роботодавця, оскільки від правильності його дій залежить своєчасне надання допомоги потерпілому, збереження обставин події та об'єктивність подальшого розслідування.

Після отримання повідомлення про нещасний випадок роботодавець повинен насамперед забезпечити надання потерпілому першої домедичної допомоги, а за необхідності організувати його доставлення до закладу охорони здоров'я [20]. Якщо існує небезпека для інших працівників, необхідно припинити роботу обладнання, відключити небезпечні джерела живлення, обмежити доступ до місця події та вжити заходів для недопущення повторного травмування.

Важливою дією є збереження обстановки на місці події до прибуття комісії з розслідування. Стан робочого місця, обладнання, інструментів, кабелів, електроживлення та інших елементів повинен залишатися таким, яким він був на момент нещасного випадку. Винятком є ситуації, коли збереження обстановки загрожує життю чи здоров'ю інших працівників або може спричинити тяжчі наслідки. У такому випадку необхідно зафіксувати місце події за допомогою фото, відео, схеми або пояснень свідків.

Роботодавець зобов'язаний протягом однієї доби повідомити про нещасний випадок відповідні органи та установи [20]. Повідомлення надсилається територіальному органу Держпраці, територіальному органу Пенсійного фонду України, керівнику підприємства, на території якого стався випадок, якщо потерпілий є працівником іншого підприємства, а також профспілковій організації

або уповноваженій особі з питань охорони праці. Якщо нещасний випадок стався внаслідок пожежі, додатково повідомляють орган ДСНС.

Не пізніше наступного робочого дня роботодавець повинен оформити письмове повідомлення про нещасний випадок за встановленою формою [20]. Після цього наказом по підприємству створюється комісія з розслідування нещасного випадку. До її складу включають представника служби охорони праці, представників роботодавця, профспілки або уповноважену особу трудового колективу. Безпосередній керівник потерпілого до складу комісії не включається, оскільки він може бути зацікавленою особою.

Роботодавець повинен створити належні умови для роботи комісії: надати необхідні документи, пояснення, технічну документацію, результати інструктажів, відомості про умови праці, дані щодо обладнання та інші матеріали, які допомагають встановити причини події. За потреби організовують фотографування місця події, проведення експертиз, лабораторних досліджень або технічних розрахунків.

Після завершення розслідування роботодавець розглядає матеріали комісії, затверджує акт за формою Н-1, організовує оформлення і розсилання матеріалів розслідування відповідним органам та потерпілому або його представникам [20]. Також роботодавець видає наказ про заходи, які необхідно виконати для запобігання подібним випадкам у майбутньому. До таких заходів можуть належати додаткові інструктажі, перевірка електрообладнання, ремонт робочого місця, оновлення інструкцій з охорони праці або посилення контролю за дотриманням правил безпеки.

Отже, основні дії роботодавця при отриманні повідомлення про нещасний випадок полягають у наданні допомоги потерпілому, повідомленні відповідних органів, збереженні місця події, створенні комісії з розслідування, забезпеченні її роботи та впровадженні заходів для недопущення повторення подібних ситуацій [20].

5.2 Чинники електричного характеру, що впливають на тяжкість ураження людини електричним струмом

Електричний струм є одним із небезпечних виробничих чинників, оскільки його дія на організм людини може спричинити місцеві ушкодження тканин, порушення роботи нервової системи, зупинку дихання або серцевої діяльності [21]. Під час розробки та налагодження програмного забезпечення для пристроїв на базі STM32 небезпека ураження електричним струмом може виникати під час роботи з блоками живлення, зарядними пристроями, програматорами, комп'ютерною технікою, макетними платами та вимірювальними приладами.

На тяжкість ураження людини електричним струмом впливають різні чинники, однак до основних чинників електричного характеру належать сила струму, напруга, електричний опір тіла людини, вид струму, частота змінного струму та шлях проходження струму через організм [21].

Найважливішим чинником є сила струму, що проходить через тіло людини. Чим більша сила струму, тим небезпечнішими є наслідки. Невеликі струми можуть викликати поколювання або незначне скорочення м'язів, а більші значення струму здатні спричинити судоми, неможливість самостійно відпустити провідник, порушення дихання, фібриляцію серця або смерть [21]. Особливо небезпечним є струм, який проходить через область серця.

Напруга також істотно впливає на тяжкість ураження [21]. Зі збільшенням напруги зростає ймовірність пробою шкіри та проходження через тіло більшого струму. Навіть відносно невисока напруга може бути небезпечною, якщо людина має вологі руки, пошкоджену шкіру або контактує з металевими частинами обладнання. Тому під час налагодження електронних пристроїв необхідно працювати тільки зі справними джерелами живлення, не торкатися оголених провідників і перед зміною з'єднань вимикати живлення схеми.

Електричний опір тіла людини залежить від стану шкіри, вологості, площі контакту та сили притискання до струмопровідної частини [21]. Суха непошкоджена шкіра має більший опір, тому струм через тіло буде меншим. Якщо

шкіра волога, забруднена або пошкоджена, її опір різко зменшується, а небезпека ураження зростає. Саме тому заборонено працювати з електрообладнанням мокрими руками або в умовах підвищеної вологості без відповідного захисту.

Вид струму також має значення. Змінний струм промислової частоти є більш небезпечним для людини, ніж постійний струм тієї самої величини, оскільки він сильніше впливає на нервову систему та може викликати судомні скорочення м’язів. Постійний струм також є небезпечним, особливо при високих напругах, але його дія зазвичай відрізняється характером впливу на організм.

Частота змінного струму впливає на біологічну дію електричного струму [21]. Найнебезпечнішим для людини вважається струм промислової частоти, оскільки він може викликати порушення серцевого ритму. При збільшенні частоти характер ураження змінюється, але небезпека опіків і теплової дії струму залишається.

Велике значення має шлях проходження струму через тіло людини. Найнебезпечнішими є шляхи «рука — рука», «ліва рука — ноги», «права рука — ноги», оскільки струм може проходити через серце, легені та центральну нервову систему [21]. Менш небезпечними є випадки, коли струм проходить тільки через невелику ділянку тіла, але навіть тоді можливі опіки та місцеві травми.

Для зменшення ризику ураження електричним струмом під час роботи з комп’ютерною технікою та електронними пристроями необхідно використовувати справні блоки живлення, ізольовані провідники, захисне заземлення, запобіжники, автоматичні вимикачі та пристрої захисного вимкнення [21]. Перед підключенням або зміною схеми потрібно вимикати живлення. Не допускається використання пошкоджених кабелів, оголених контактів, несправних розеток і саморобних з’єднань без ізоляції.

Отже, тяжкість ураження людини електричним струмом залежить насамперед від сили струму, напруги, опору тіла, виду і частоти струму та шляху його проходження через організм [21]. Дотримання правил електробезпеки, справність обладнання та правильна організація робочого місця значно зменшують ризик виникнення електротравм.

ВИСНОВКИ

У процесі виконання кваліфікаційної роботи було розроблено програмне забезпечення для мультиканального виконавчого пристрою високої надійності на базі мікроконтролера STM32. Розроблена система призначена для централізованого керування групою периферійних виконавчих вузлів, контролю їхнього стану, обміну службовими даними та відображення діагностичної інформації користувачу.

У загальному розділі було проведено аналітичний огляд існуючих підходів до побудови багатоканальних вбудованих систем керування. Розглянуто особливості використання мікроконтролерів STM32, інтерфейсів UART та RS-485, пакетних протоколів обміну, CRC-контролю, DMA, людино-машинних інтерфейсів і програмних засобів емуляції периферійних пристроїв. На основі проведеного аналізу було обґрунтовано доцільність використання власного спеціалізованого протоколу обміну, локального графічного інтерфейсу та програмного емулятора для стендового тестування.

У межах технічного завдання визначено основні вимоги до програмного забезпечення. Було встановлено, що система повинна забезпечувати стабільний обмін даними з периферійними вузлами, контроль цілісності пакетів, діагностику стану каналів, обробку помилок, контроль втрати зв’язку, підтримку режимів заряджання, одиночної та групової активації, а також відображення інформації на TFT-дисплеї.

У другому розділі було розроблено технічний та робочий проєкт програмного забезпечення. Описано структуру програмної системи, основні модулі, організацію вхідних і вихідних даних, алгоритми обміну та діагностики, інформаційні зв’язки між частинами програми, а також принципи написання тексту програми. Основна реалізація виконана мовою C у середовищі STM32CubeIDE з використанням бібліотеки HAL, FreeRTOS, UART DMA та графічного дисплея ILI9341.

Для забезпечення стабільної роботи програмне забезпечення було

побудовано за модульним принципом. Окремі задачі відповідають за прикладну логіку, комунікацію з периферійними вузлами, обробку кнопок та оновлення графічного інтерфейсу. Такий підхід дозволив розділити функції системи, зменшити взаємне блокування окремих процесів і забезпечити своєчасну реакцію на події користувача та зміни стану периферійних вузлів.

У роботі реалізовано систему діагностики 32 виконавчих каналів. Кожен вузол має власний стан, який відображається в інтерфейсі користувача. Передбачено відображення неготового стану, готовності до заряджання, процесу заряджання, готовності до активації, активованого стану та стану несправності. Для аналізу окремого вузла реалізовано детальний екран, на якому відображаються службові параметри та інформація про можливі помилки.

Для перевірки роботи основної STM32-програми було розроблено допоміжний програмний емулятор периферійних вузлів. Емулятор реалізовано у середовищі Qt із використанням об’єктно-орієнтованого підходу та модуля роботи з послідовним портом. Він дозволяє імітувати роботу до 32 вузлів, задавати їхні стани, формувати діагностичні відповіді, перевіряти обмін даними та моделювати типові й аварійні ситуації без постійного використання повного апаратного стенда.

У спеціальному розділі було наведено інструкції з інсталяції програмного забезпечення, використання тестових наборів та експлуатації програмного комплексу. Описано порядок підготовки середовища, прошивання STM32-пристрою, запуску допоміжного емулятора, перевірки режимів роботи, використання діагностики, заряджання, одиночної та групової активації, а також дії користувача у разі втрати зв’язку або появи несправностей.

В економічному розділі виконано розрахунок витрат на розробку програмного забезпечення. Було визначено трудомісткість робіт, витрати на оплату праці, електроенергію, амортизацію, накладні витрати, собівартість, ціну розробки, чисту теперішню вартість та термін окупності. Отримані результати підтверджують економічну доцільність виконаної розробки.

У розділі охорони праці розглянуто дії роботодавця при отриманні повідомлення про нещасний випадок, а також чинники електричного характеру, що

впливають на тяжкість ураження людини електричним струмом. Описано основні заходи безпеки під час роботи з комп’ютерною технікою, джерелами живлення, електронними пристроями та налагоджувальним обладнанням.

Отже, у результаті виконання кваліфікаційної роботи було створено програмне забезпечення, яке забезпечує керування мультиканальним виконавчим пристроєм, стабільний обмін даними з периферійними вузлами, контроль стану 32 каналів, діагностику несправностей, захист від випадкової активації та зручне відображення інформації користувачу. Розроблена система відповідає поставленим вимогам і може бути використана як основа для подальшого розвитку програмного комплексу, розширення функціональних можливостей та адаптації до різних задач автоматизації й керування.

ПЕРЕЛІК ПОСИЛАНЬ

1. STM32H750VB, STM32H750ZB, STM32H750IB datasheet. STMicroelectronics. URL: https://www.st.com/resource/en/datasheet/stm32h750ib.pdf (дата звернення: 29.05.2026).

2. RM0433 Reference manual. STM32H742, STM32H743/753 and STM32H750 Value line advanced Arm-based 32-bit MCUs. STMicroelectronics. URL: https://www.st.com/resource/en/reference_manual/rm0433-stm32h742-stm32h743753-and-stm32h750-value-line-advanced-armbased-32bit-mcus-stmicroelectronics.pdf (дата звернення: 29.05.2026).

3. RS-422 and RS-485 Standards Overview and System Configurations : Application Report SLLA070. Texas Instruments. URL: https://www.ti.com/lit/pdf/slla070 (дата звернення: 29.05.2026).

4. Qt 6 Documentation. Qt Group. URL: https://doc.qt.io/qt-6/ (дата звернення: 29.05.2026).

5. Qt Widgets Module. Qt 6 Documentation. Qt Group. URL: https://doc.qt.io/qt-6/qtwidgets-index.html (дата звернення: 29.05.2026).

6. MODBUS over Serial Line Specification and Implementation Guide. Version 1.02. Modbus Organization, 2006. URL: https://www.modbus.org/file/secure/modbusoverserial.pdf (дата звернення: 29.05.2026).

7. ILI9341 a-Si TFT LCD Single Chip Driver. Specification. ILI Technology Corp. URL: https://cdn-shop.adafruit.com/datasheets/ILI9341.pdf (дата звернення: 29.05.2026).

8. FreeRTOS Documentation. Kernel overview. FreeRTOS. URL: https://www.freertos.org/Documentation/00-Overview (дата звернення: 29.05.2026).

9. Queue Management. FreeRTOS Kernel API Reference. FreeRTOS. URL: https://www.freertos.org/Documentation/02-Kernel/04-API-references/06-Queues/00-QueueManagement (дата звернення: 29.05.2026).

10. STM32CubeH7 MCU package. STM32H7 HAL drivers and software components. STMicroelectronics. URL: https://www.st.com/en/embedded-software/stm32cubeh7.html (дата звернення: 29.05.2026).

11. Getting started with STM32Cube HAL. STMicroelectronics Wiki. URL: https://wiki.st.com/stm32mcu/wiki/Getting_started_with_STM32_:_STM32Cube_HAL (дата звернення: 29.05.2026).

12. UM2609 STM32CubeIDE user guide. STMicroelectronics. URL: https://www.st.com/resource/en/user_manual/um2609-stm32cubeide-user-guide-stmicroelectronics.pdf (дата звернення: 29.05.2026).

13. Методичні вказівки до виконання кваліфікаційної роботи для здобувачів фахової передвищої освіти спеціальності 122 «Комп'ютерні науки» / уклад.: Г. Я. Марціяш, Р. О. Слободян. Тернопіль : ВСП «ТФК ТНТУ ім. І. Пулюя», 2026. 48 с.

14. ДСТУ 3008:2015. Інформація та документація. Звіти у сфері науки і техніки. Структура та правила оформлювання. Київ : ДП «УкрНДНЦ», 2016.

15. ДСТУ 8302:2015. Інформація та документація. Бібліографічне посилання. Загальні положення та правила складання. Київ : ДП «УкрНДНЦ», 2016. 20 с.

16. Pro Git Book. Getting Started — About Version Control. Git SCM. URL: https://git-scm.com/book/en/v2/Getting-Started-About-Version-Control (дата звернення: 29.05.2026).

17. QSerialPort Class. Qt Serial Port. Qt 6 Documentation. Qt Group. URL: https://doc.qt.io/qt-6/qserialport.html (дата звернення: 29.05.2026).

18. STM32CubeIDE. Integrated development environment for STM32. STMicroelectronics. URL: https://www.st.com/en/development-tools/stm32cubeide.html (дата звернення: 29.05.2026).

19. Терміни амортизації основних засобів у податковому обліку. Державна податкова служба України. URL: https://if.tax.gov.ua/media-ark/news-ark/437911.html (дата звернення: 29.05.2026).

20. Про затвердження Порядку розслідування та обліку нещасних випадків, професійних захворювань та аварій на виробництві : Постанова Кабінету Міністрів України від 17.04.2019 № 337. Верховна Рада України. URL: https://zakon.rada.gov.ua/go/337-2019-%D0%BF (дата звернення: 29.05.2026).

21. НПАОП 0.00-1.28-10. Правила охорони праці під час експлуатації електронно-обчислювальних машин : наказ Державного комітету України з промислової безпеки, охорони праці та гірничого нагляду від 26.03.2010 № 65. Верховна Рада України. URL: https://zakon.rada.gov.ua/go/z0293-10 (дата звернення: 29.05.2026).

ДОДАТКИ

Додаток А. Лістинг програмного коду задач FreeRTOS

```
void StartAppTask(void const *argument)
{
    (void)argument;
    App_InterfaceInit();

    for (;;)
    {
        osEvent event = osMessageGet(appEventQueueHandle, APP_WAIT_MS);

        if (event.status == osEventMessage)
        {
            App_HandleEvent(event.value.v);
        }

        App_Periodic();
    }
}

void StartCommTask(void const *argument)
{
    (void)argument;

    for (;;)
    {
        osEvent event = osMessageGet(commCmdQueueHandle, 0U);

        if (event.status == osEventMessage)
        {
            Comm_HandleCommand(event.value.v);
```

```
        }

        Comm_Periodic();
        osDelay(COMM_TASK_DELAY_MS);
    }
}

void StartUiTask(void const *argument)
{
    (void)argument;

    for (;;)
    {
        Ui_Periodic();
    }
}

void StartInputTask(void const *argument)
{
    (void)argument;

    for (;;)
    {
        uint16_t edges = Input_CollectEdges();

        if (edges != 0U)
        {
            (void)App_PostEvent(APP_EVENT_INPUT, edges);
        }

        osDelay(INPUT_SCAN_DELAY_MS);
    }
}
```

Додаток Б. Лістинг програмного коду черг подій між задачами

```
osStatus App_PostEvent(app_event_type_t type, uint16_t arg)
{
uint32_t message = App_PackMessage((uint8_t)type, arg);
if (appEventQueueHandle == NULL)
{
    return osErrorOS;
}

return Queue_Send(appEventQueueHandle, &message);
}
osStatus App_PostCommCommand(app_comm_cmd_t cmd, uint16_t arg)
{
uint32_t message = App_PackMessage((uint8_t)cmd, arg);
if (commCmdQueueHandle == NULL)
{
    return osErrorOS;
}

return Queue_Send(commCmdQueueHandle, &message);
}
osStatus App_PostUiRender(ui_render_cmd_t cmd, uint16_t arg)
{
uint32_t message = App_PackMessage((uint8_t)cmd, arg);
if (uiRenderQueueHandle == NULL || cmd == UI_RENDER_NONE)
{
    return osOK;
}

return Queue_Send(uiRenderQueueHandle, &message);
}
```

Додаток В. Лістинг структури даних периферійного вузла

```
#define NODE_MAX_COUNT NODE_COUNT_LIMIT
typedef struct
{
uint8_t response_index; /* позиція вузла у відповіді /
uint8_t state; / узагальнений стан вузла /
uint8_t command_status; / результат обробки службової команди */
uint16_t supply_mv;          /* службова напруга вузла */
uint16_t capacitor_mv;       /* напруга накопичувача */

uint8_t  error_code;         /* узагальнений код помилки */
uint8_t  error_param;        /* додатковий параметр діагностики */

uint32_t last_update_ms;     /* час останнього оновлення */
uint8_t  packet_num;         /* номер останнього службового пакета */
} device_info_t;
typedef struct
{
device_info_t devices[NODE_MAX_COUNT];
uint8_t device_count;
uint8_t active_count;
uint8_t cycle_num;
} app_diag_snapshot_t;
```

Додаток Г. Лістинг програмного коду приймання даних через UART DMA

```
void HAL_UARTEx_RxEventCallback(UART_HandleTypeDef *huart, uint16_t size)
{
if (huart->Instance == DEVICE_UART_INSTANCE)
{
uint8_t end_of_burst = Uart_IsIdleEvent(huart);
    if (size > 0U)
    {
        uint16_t copy_len = LimitSize(size, UART_DMA_RX_BUF_SIZE);
        Uart_InvalidateDmaCache(g_uart_rx_dma_buf, copy_len);

        if (g_uart_rx_queue_count < UART_RX_QUEUE_DEPTH)
        {
            uint8_t head = g_uart_rx_queue_head;
            memcpy(g_uart_rx_queue[head], g_uart_rx_dma_buf, copy_len);
            g_uart_rx_queue_len[head] = copy_len;
            g_uart_rx_queue_event[head] = end_of_burst;
            g_uart_rx_queue_head = NextQueueIndex(head);
            g_uart_rx_queue_count++;
        }
        else
        {
            g_comm_rx_drop_count++;
        }
    }
    (void)osSignalSet(commTaskHandle, APP_COMM_SIGNAL_RX);

    HAL_UARTEx_ReceiveToIdle_DMA(huart,
                                 g_uart_rx_dma_buf,
                                 sizeof(g_uart_rx_dma_buf));
}
}
```

Додаток Ґ. Лістинг функції обробки прийнятих пакетів

```
uint8_t DeviceParser_Feed(device_parser_t *parser,
const uint8_t *buf,
uint16_t size,
uint8_t end_of_burst,
uint32_t now_ms,
device_response_frame_t *out_frame)
{
if (parser == NULL || buf == NULL || out_frame == NULL || size == 0U)
{
return 0U;
}
Parser_AppendBytes(parser, buf, size);

while (Parser_HasPacketCandidate(parser))
{
    packet_header_t header = Parser_ReadHeader(parser);

    if (header == HEADER_SERVICE_PACKET)
    {
        Parser_ConsumePacket(parser, FRAME_SERVICE_SIZE);
        continue;
    }

    if (header == HEADER_DEVICE_PACKET)
    {
        if (Parser_HasCompleteDevicePacket(parser) &&
            Packet_CheckStructure(parser) &&
            Packet_CheckCrcShort(parser))
        {
            device_info_t *node = Parser_NextNode(parser);

            node->response_index = Parser_GetNodeIndex(parser);
```

```
            node->state = Packet_GetState(parser);
            node->command_status = Packet_GetCommandStatus(parser);
            node->supply_mv = Packet_GetSupplyMv(parser);
            node->capacitor_mv = Packet_GetCapacitorMv(parser);
            node->error_code = Packet_GetErrorCode(parser);
            node->last_update_ms = now_ms;
        }

        Parser_ConsumePacket(parser, FRAME_DEVICE_SIZE);
        continue;
    }

    Parser_SkipByte(parser);
}

if (end_of_burst != 0U)
{
    *out_frame = parser->frame;
    Parser_ResetFrame(parser);
    return 1U;
}

return 0U;
```

Додаток Д. Лістинг функції оновлення діагностичної моделі

```
static void PublishDeviceFrame(const device_response_frame_t *frame)
{
app_diag_snapshot_t *snapshot;
uint8_t publish_idx;
uint8_t count;
if (frame == NULL)
{
    return;
}

publish_idx = GetNextSnapshotIndex();
snapshot = &g_diag_snapshots[publish_idx];

memset(snapshot, 0, sizeof(*snapshot));

count = LimitNodeCount(frame->device_count);
for (uint8_t i = 0U; i < count; ++i)
{
    snapshot->devices[i] = frame->devices[i];
}

snapshot->device_count = count;
snapshot->active_count = count;
snapshot->cycle_num = frame->packet_num;

taskENTER_CRITICAL();
g_diag_snapshot_published_idx = publish_idx;
taskEXIT_CRITICAL();

(void)App_PostEvent(APP_EVENT_DIAG_DIRTY, UI_RENDER_DIAG);
}
```

Додаток Е. Лістинг функції оновлення графічного інтерфейсу на TFT-дисплеї

```
static void DrawDiagnostics(void)
{
LoadLatestSnapshotToRuntimeCache();
if (!g_diag_screen_inited)
{
    TFT_FillScreen(COLOR_BACKGROUND);
    TFT_WriteText(UI_X_TITLE, UI_Y_TITLE,
                  "Diagnostics",
                  COLOR_TEXT,
                  FONT_TITLE);

    InvalidateDiagCellCache(last_drawn_states);
    g_diag_screen_inited = 1U;
}

if (g_connection_lost)
{
    DrawDiagnosticMessage("Connection status unavailable");
    DrawDiagFooter(0U);
    return;
}

for (uint16_t idx = 0U; idx < NODE_MAX_COUNT; ++idx)
{
    uint8_t state = GetDiagStateByIndex(idx);

    if (state == DIAG_STATE_INACTIVE)
    {
        ClearDiagCell(idx);
        continue;
    }
```

```
    if (CellStateChanged(idx, state))
    {
        uint32_t color = DeviceStateToColor(state);
        DrawDiagCell(idx, color);
        DrawDiagCellNumber(idx);
    }
}

DrawDiagFooter(g_active_node_count);
}
```

Додаток Є. Лістинг функції режиму заряджання

```
static void ActionCharging(void)
{
uint16_t ready_count = 0U;
uint16_t not_ready_count = 0U;
LoadLatestSnapshotToRuntimeCache();

for (uint16_t idx = 0U; idx < GetActiveNodeCount(); ++idx)
{
    if (GetDiagStateByIndex(idx) == DEVICE_READY_FOR_CHARGING)
    {
        ready_count++;
    }
    else
    {
        not_ready_count++;
    }
}

if (ready_count == 0U)
{
    DisplayMessage("Charging unavailable", MESSAGE_WARNING);
    return;
}

(void)App_PostCommCommand(COMM_CMD_SET_CHARGING,
                          CMD_ARG_ALL_NODES);

if (not_ready_count > 0U)
{
    DisplayMessage("Some nodes are not ready", MESSAGE_INFO);
}
else
```

```
{
    DisplayMessage("Charging", MESSAGE_INFO);
}
}
```

Додаток Ж. Лістинг програмного коду одиночної та групової активації

```
static void ActionSingleActivation(void)
{
LoadLatestSnapshotToRuntimeCache();
for (uint16_t idx = 0U; idx < GetActiveNodeCount(); ++idx)
{
    if (GetDiagStateByIndex(idx) == DEVICE_READY_FOR_ACTIVATION)
    {
        uint16_t target = GetNodeResponseIndex(idx);

        (void)App_PostCommCommand(COMM_CMD_SET_ACTIVATION, target);
        DisplayMessage("Single activation", MESSAGE_INFO);
        return;
    }
}

DisplayMessage("No nodes ready for activation", MESSAGE_WARNING);
}
static void ActionGroupActivation(void)
{
g_group_target_count = 0U;
g_group_target_pos = 0U;
LoadLatestSnapshotToRuntimeCache();

for (uint16_t idx = 0U; idx < GetActiveNodeCount(); ++idx)
{
    if (GetDiagStateByIndex(idx) == DEVICE_READY_FOR_ACTIVATION)
    {
        g_group_targets[g_group_target_count++]                    =
GetNodeResponseIndex(idx);
    }

    if (g_group_target_count >= NODE_MAX_COUNT)
```

```
    {
        break;
    }
}

if (g_group_target_count == 0U)
{
    DisplayMessage("No nodes ready for activation", MESSAGE_WARNING);
    return;
}

g_group_activation_active = 1U;
g_group_last_ms = 0U;

}
```

Додаток З. Лістинг програмного коду моделі периферійного вузла в емуляторі

Файл: node_model.h, node_model.cpp

```
struct NodeParams
{
uint16_t supplyMv;
uint16_t capacitorMv;
uint8_t diagnosticCode;
uint8_t diagnosticParam;
};
class PeripheralNode
{
private:
uint16_t serviceHeader; // SERVICE_HEADER_PLACEHOLDER
uint8_t packetNumber;
uint8_t responseStatus; // RESPONSE_PLACEHOLDER
uint8_t nodeState; // STATE_PLACEHOLDER
NodeParams params;
uint16_t checksum; // CHECKSUM_PLACEHOLDER
public:
PeripheralNode()
: serviceHeader(SERVICE_HEADER_PLACEHOLDER),
packetNumber(0),
responseStatus(RESPONSE_DEFAULT),
nodeState(STATE_NOT_READY),
params{0, 0, 0, 0},
checksum(0)
{}
uint8_t state() const
{
    return nodeState;
}
```

```
void setState(uint8_t state)
{
    nodeState = state;
}

uint8_t response() const
{
    return responseStatus;
}

void setResponse(uint8_t response)
{
    responseStatus = response;
}

NodeParams nodeParams() const
{
    return params;
}

void setNodeParams(const NodeParams &value)
{
    params = value;
}

uint8_t currentPacketNumber() const
{
    return packetNumber;
}
void setPacketNumber(uint8_t value)
{
    packetNumber = value;
}
};
```

Додаток И. Лістинг програмного коду роботи з COM-портом у програмному емуляторі

```
void MainWindow::refreshAvailablePorts()
{
const QString current = ui->port_cb->currentText();
QStringList ports;
const auto infos = QSerialPortInfo::availablePorts();
for (const QSerialPortInfo &info : infos)
{
    ports << info.portName();
}

ui->port_cb->blockSignals(true);
ui->port_cb->clear();
ui->port_cb->addItems(ports);

const int index = ports.indexOf(current);
if (index >= 0)
{
    ui->port_cb->setCurrentIndex(index);
}

ui->port_cb->blockSignals(false);
}
void MainWindow::on_connect_pb_clicked()
{
if (m_serial.isOpen())
{
m_serial.close();
m_readBuffer.clear();
ui->connect_pb->setText("Connect");
return;
}
```

```
const QString portName = ui->port_cb->currentText();
const int baud = ui->baud_cb->currentData().toInt();

m_serial.setPortName(portName);
m_serial.setBaudRate(baud);
m_serial.setDataBits(QSerialPort::Data8);
m_serial.setParity(QSerialPort::NoParity);
m_serial.setStopBits(QSerialPort::OneStop);
m_serial.setFlowControl(QSerialPort::NoFlowControl);

if (m_serial.open(QIODevice::ReadWrite))
{
    m_readBuffer.clear();
    ui->connect_pb->setText("Disconnect");

    connect(&m_serial,
            &QSerialPort::readyRead,
            this,
            &MainWindow::onSerialReadyRead,
            Qt::UniqueConnection);
}

}
void MainWindow::onSerialReadyRead()
{
const QByteArray chunk = m_serial.readAll();
m_readBuffer.append(chunk);
processIncomingBuffer();

}
```

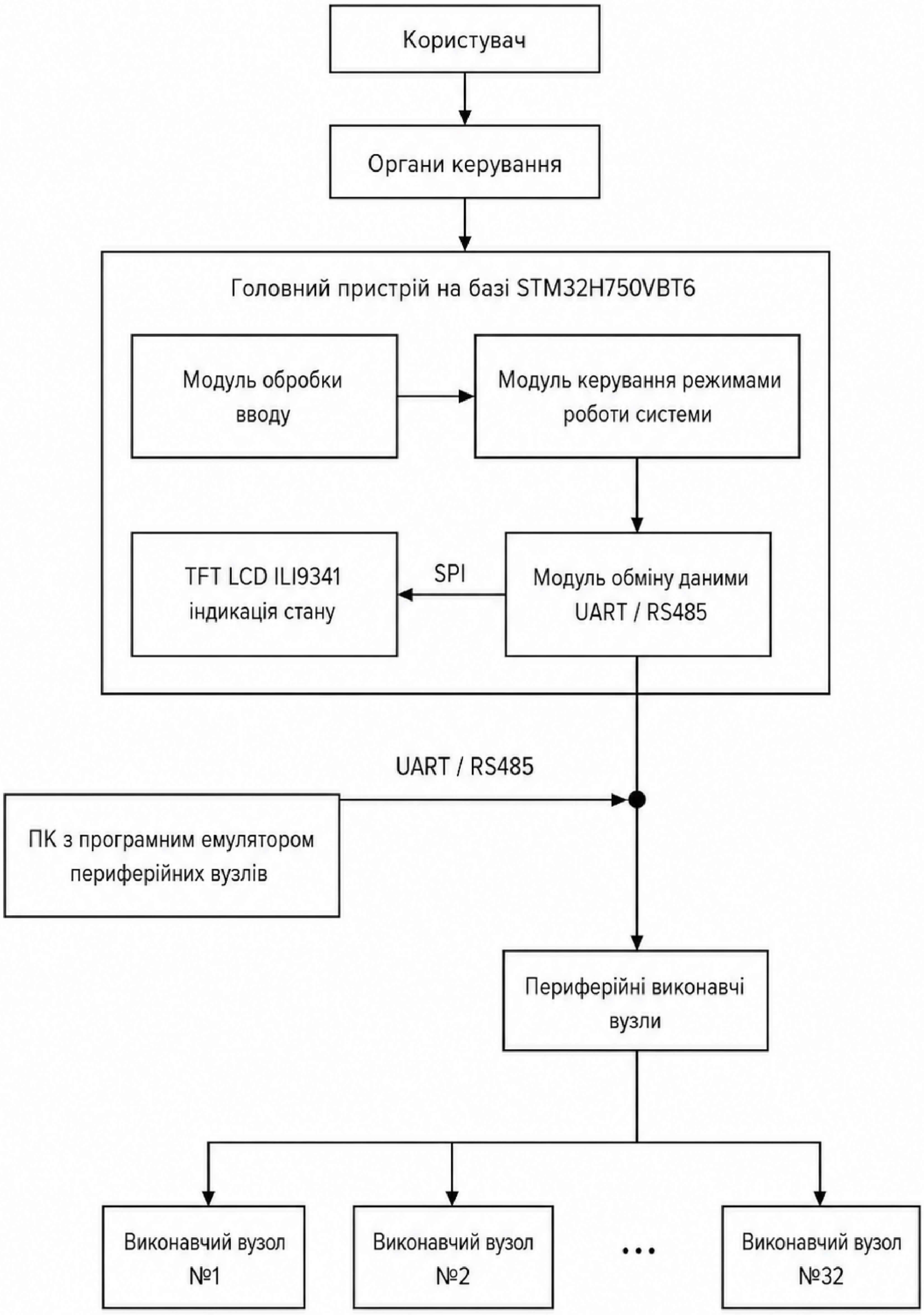
Додаток І. Лістинг програмного коду інтерфейсу керування станами вузлів

```
QWidget* MainWindow::createDeviceWidget(int index)
{
auto *container = new QWidget(this);
container->setObjectName(QString("node_card_%1").arg(index));
auto *layout = new QVBoxLayout(container);
auto *topRow = new QHBoxLayout();
auto *stateRow = new QHBoxLayout();

auto *nameLabel = new QLabel(QString("Node %1").arg(index + 1), container);
auto *stateLabel = new QLabel("state: NOT_READY", container);
stateLabel->setObjectName(QString("state_l_%1").arg(index));

auto *stateCb = new QComboBox(container);
stateCb->setObjectName(QString("state_cb_%1").arg(index));

stateCb->addItem("NOT_READY", STATE_NOT_READY);
stateCb->addItem("READY_FOR_CHARGING", STATE_READY_FOR_CHARGING);
stateCb->addItem("CHARGING", STATE_CHARGING);
stateCb->addItem("READY_FOR_ACTIVATION", STATE_READY_FOR_ACTIVATION);
stateCb->addItem("ACTIVATED", STATE_ACTIVATED);
stateCb->addItem("FAULT", STATE_FAULT);

auto *applyBtn = new QPushButton("Apply", container);

topRow->addWidget(nameLabel);
topRow->addStretch();
topRow->addWidget(stateLabel);

stateRow->addWidget(stateCb);
stateRow->addWidget(applyBtn);
```

```
layout->addLayout(topRow);
layout->addLayout(stateRow);

return container;
}
void MainWindow::updateDeviceFromInputs(int index)
{
if (index < 0 || index >= m_nodes.size())
{
return;
}
QWidget *w = m_nodeWidgets.value(index, nullptr);
if (!w)
{
    return;
}

auto                    *stateCb                    =                    w-
>findChild<QComboBox*>(QString("state_cb_%1").arg(index));
auto *diagnosticFlag =
    w->findChild<QCheckBox*>(QString("diagnostic_flag_%1").arg(index));

uint8_t state = m_nodes[index].state();
uint8_t response = RESPONSE_OK;

if (diagnosticFlag && diagnosticFlag->isChecked())
{
    state = STATE_FAULT;
    response = RESPONSE_DIAGNOSTIC;
}
else if (stateCb)
{
    state = static_cast<uint8_t>(stateCb->currentData().toInt());
}
```

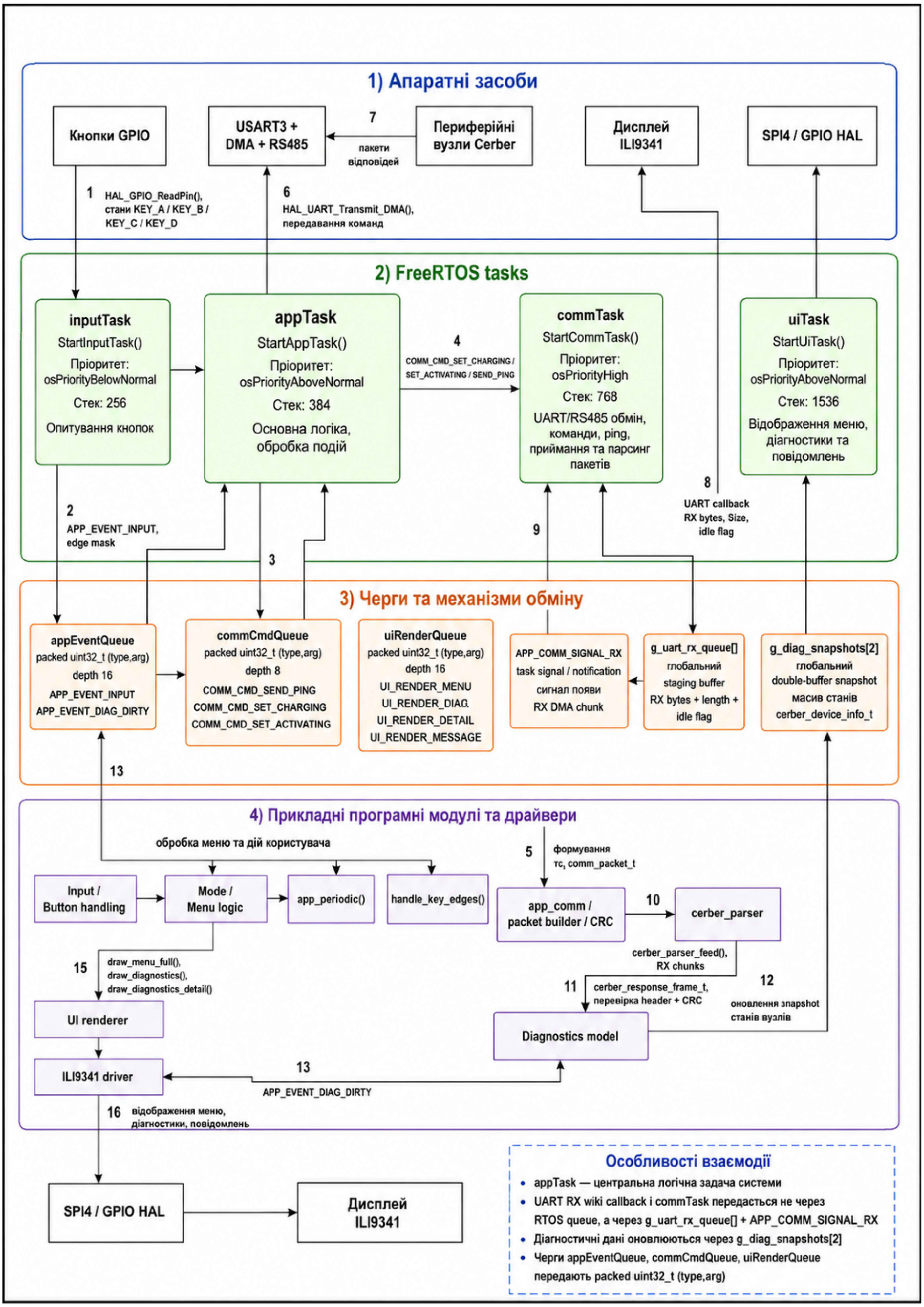
```
NodeParams params = m_nodes[index].nodeParams();

params.diagnosticCode = diagnosticFlag && diagnosticFlag->isChecked()
    ? DIAGNOSTIC_CODE_PLACEHOLDER
    : DIAGNOSTIC_OK;

m_nodes[index].setNodeParams(params);
m_nodes[index].setState(state);
m_nodes[index].setResponse(response);

updateDeviceStateLabel(index);

}
```

Структурна схема програмно-апаратної системи



					2026. КВР. 122.421.06.00.00 Е1			
Зм.	Арк.	№ документу	Підпис	Дата	Розробка програмного забезпечення для мультиканального виконавчого пристрою високої надійності на базі мікроконтролера МК STM32 Схема електрична структурна програмно-апаратної системи	Літ.	Маса	Масшт.
Розроб.		Гринчишин Я.Ю.						
Перевір.		Марціяш Г.Я.				Аркуш	Аркушів 1	
Т. контр.								
Н. контр.		Приймак В. А.				ВСП ТФК ТНТУ КН-421 м. Тернопіль		
Затв.								

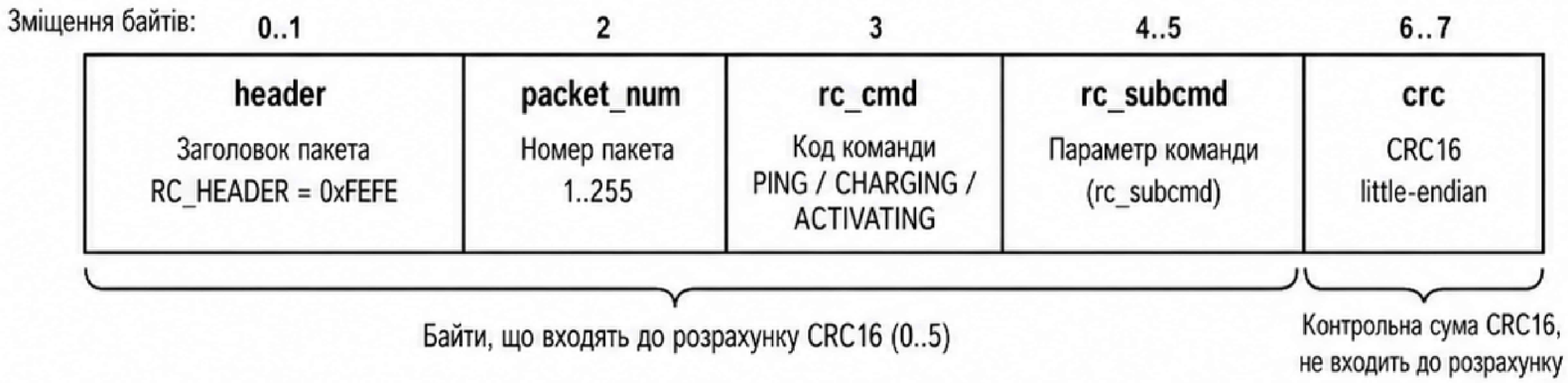
Схема взаємодії програмних модулів



					2026. КВР . 122.421.06.00.00 СВ		
Зм.	Арк.	№ документу	Підпис	Дата	Розробка програмного забезпечення для мультиканального виконавчого пристрою високої надійності на базі мікроконтролера МК STM32 Схема взаємодії програмних модулів	Літ.	Маса / Масшт.
Розроб.		Гринчишин Я.Ю.					
Перевір.		Марціяш Г.Я.				Аркуш	Аркушів 1
Т контр.							
Н. контр.		Приймак В. А.				ВСП ТФК ТНТУ КН-421 м.Тернопіль	
Затв.							

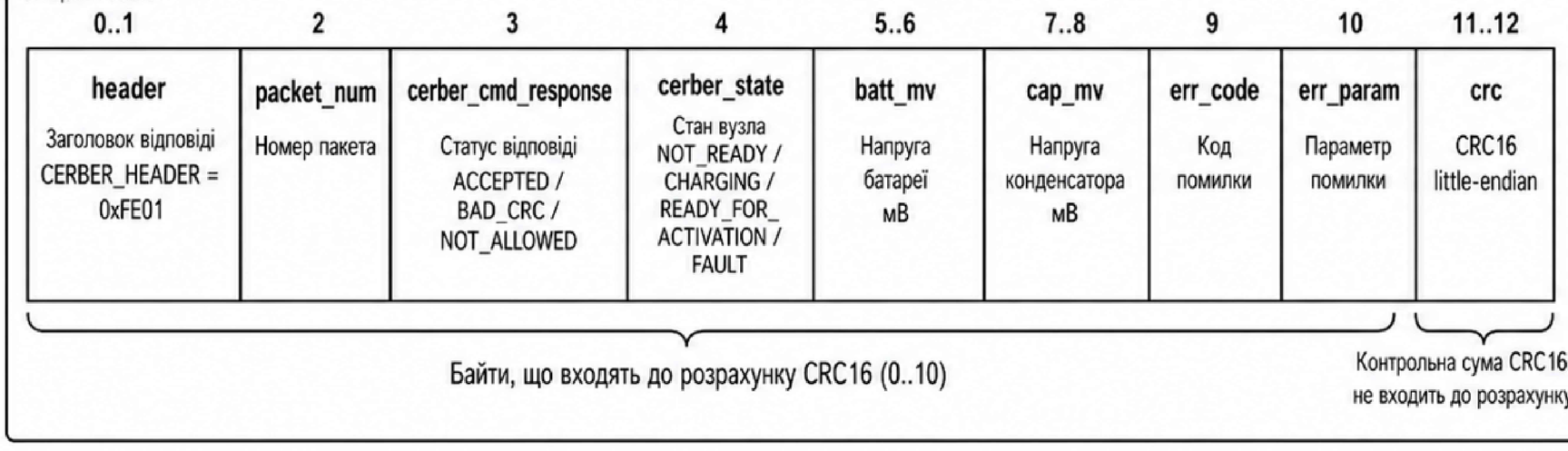
Схема структурни протоколу обміну даними

Пакет команди rc_comm_packet_t (8 байт)



- 0..1 – службовий заголовок пакета команди;
- 2 – номер пакета;
- 3 – код команди;
- 4..5 – параметр команди;
- 6..7 – CRC16.

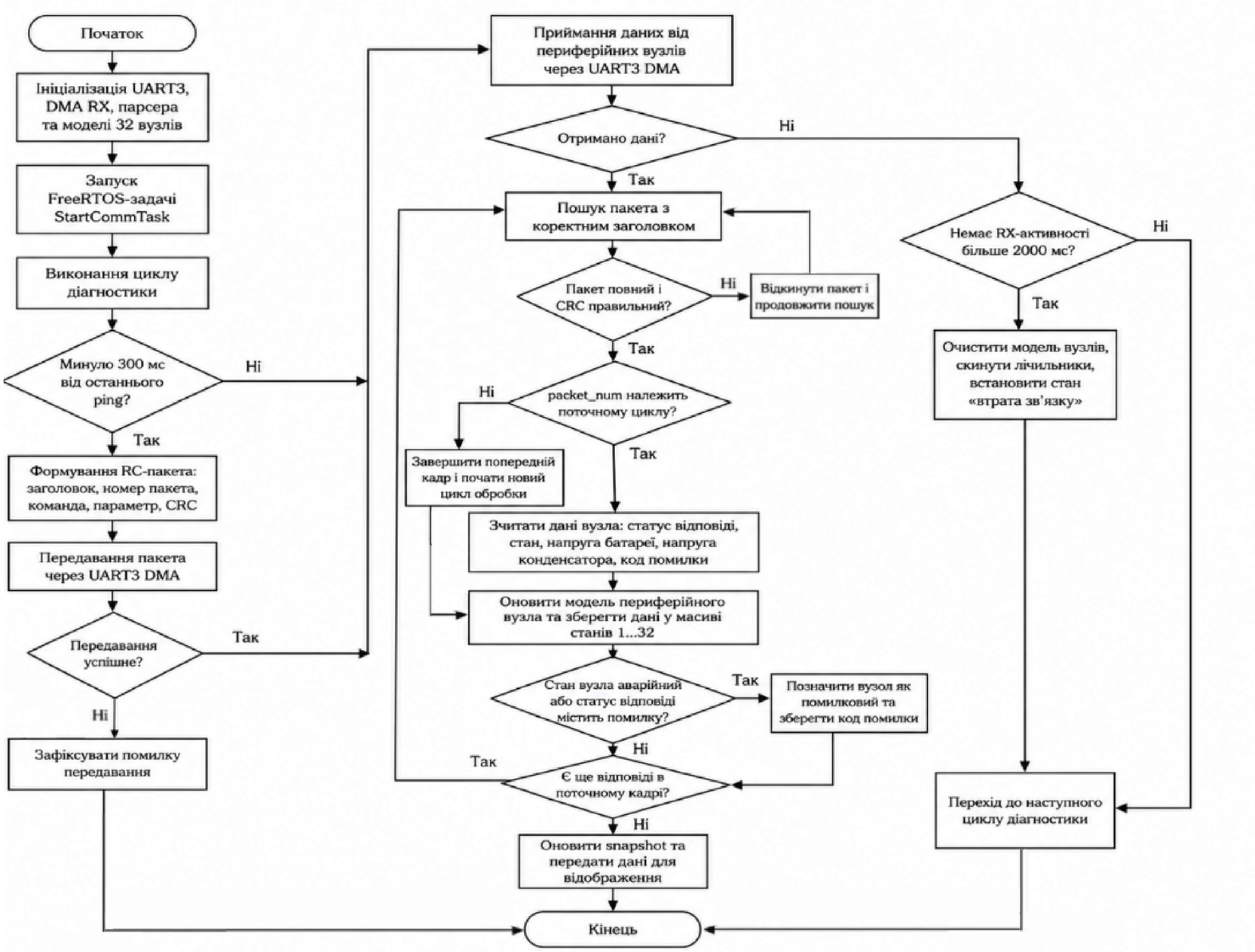
Пакет відповіді cerber_comm_packet_t (13 байт)



- 0..1 – службовий заголовок відповіді;
- 2 – номер пакета;
- 3 – статус виконання команди;
- 4 – стан периферійного вузла;
- 5..6 – напруга батареї;
- 7..8 – напруга конденсатора;
- 9 – код помилки;
- 10 – параметр помилки;
- 11..12 – CRC16.

Зм.	Арк.	№ документу	Підпис	Дата	2026.КВР.122.421.06.00.00 СС	
Розроб.		Гринчишин Я.Ю.			Розробка програмного забезпечення для мультиканального виконавчого пристрою високої надійності на базі мікроконтролера MK STM32 Схема структурна протоколу обміну даними	Літ. / Маса / Масшт.
Перевір.		Мартиш Г.Я.				
Т.контр.						Аркуш / Аркушів 1
Н. контр.		Грайнак В.А.				ВСП ТФК ТНТУ КН-421 м.Тернопіль
Затв.						

Блок-схема алгоритму діагностики периферійних вузлів



					2026. КВР. 122.421.06.00.00 БС		
Зм.	Арк.	№ документу	Підпис	Дата	Розробка програмного забезпечення для мультиканального акваційного пристрою високої надійності на базі мікроконтролера МК STM32	Літ.	Маса	Масшт.
Розроб.		Гринчишин Я.Ю.						
Перевір.		Мартиш І.Я.			Блок-схема алгоритму діагностики периферійних вузлів	Аркуш	Аркушів 1	
Т. контр.						ВСП ТФК ТНТУ КН-421 м.Тернопіль		
Н. контр.		Прийма В.А.						
Затв.								

Таблиця техніко-економічних показників

№	Показник	Одиниці вимірювання	Значення
1	Платформа	–	STM32H750VBT6
2	Призначення	–	Мультиканальний виконавчий пристрій
3	Мова програмування	–	C, C++
4	Середовище розробки	–	STM32CubeIDE, Qt
5	Інтерфейс обміну даними	–	UART / RS485
6	Собівартість	грн	65 856
7	Плановий прибуток або грошовий потік	грн	36 879
8	Ціна	грн	102 735
9	Чиста теперішня вартість	грн	25 857
10	Термін окупності	рік	2,1

					2026. КВР. 122.421.06.00.00 ТБ	
Зм.	Арк.	№ документу	Підпис	Дата	Розробка програмного забезпечення для мультиканального виконавчого пристрою високої надійності на базі мікроконтролера МК STM32. Таблиця техніко-економічних показників	Літ. / Маса / Масшт.
Розроб.		Гринчишин Я. Ю.				
Перевір.		Мерцінш Г. Я.				Аркуш / Аркушів 1
Т. контр.						
Н. контр.		Приймак В. А.				ВСП ТФК ТНТУ КН-421 м. Тернопіль
Затв.						