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

КВАЛІФІКАЦІЙНА РОБОТА

бакалавр
Освітній рівень

Апаратно-програмний комплекс дистанційного керування залізничним макетом на мікроконтролерах з мультиплатформною програмною підтримкою
Назва теми

КвРКІ .240122.22.05 ПЗ
Шифр

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

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

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

Виконав: студент III курсу, група КІ2с-22-1
Підпис Дем`ян ГЕТЬМАН
Ініціали, прізвище

Керівник Підпис, дата Дмитро МЕДЗАТИЙ
Ініціали, прізвище

Нормоконтролер Підпис, дата Тетяна КИСІЛЬ
Ініціали, прізвище

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

«16» червня 2025 р.

Хмельницький 2025

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет Інформаційних технологій

Кафедра Комп'ютерної інженерії та інформаційних систем

Освітній рівень бакалавр

Галузь знань 12 Інформаційні технології

Спеціальність 123 Комп'ютерна інженерія

Освітня програма «Комп'ютерна інженерія та програмування»

ЗАТВЕРДЖУЮ

Зав. кафедри Ольга ПАВЛОВА

" 10 " 01 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ БАКАЛАВРА

Дем`яну ГЕТЬМАНУ

Прізвище, ім'я, по батькові студента

1. Тема проекту (роботи) Апаратно-програмний комплекс дистанційного керування залізничним макетом на мікроконтролерах з мультиплатформною програмною підтримкою

Керівник проекту (роботи) Дмитро МЕДЗАТИЙ, к.т.н., доцент

Прізвище, ім'я, по батькові, науковий ступінь, вчене звання

Затверджена наказом ректора університету від 01.03.2025 р. № 5

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

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

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

Дослідження предметної області та постановка задачі; Аналіз сучасного стану розвитку залізничного моделювання в Україні та світі; Аналіз ситуації в світових наукових колах; Наукові публікації і технічні дослідження, що стосуються предметної області.
Апаратна реалізація комплексу; Апаратна реалізація стандарту DCC (Digital Command Control).
Програмна реалізація комплексу; Проєктування архітектури, структур даних, інтерфейсу ПЗ; Розробка та опис програмної реалізації та тестування задачі; Тестування та експлуатація апаратно-програмного комплексу.

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслень)

Структура апаратно-програмного комплексу
Командна станція. Схема електрична структурна
Діаграма варіантів використання програми
Статичне представлення структури моделі програми

6. Консультанти розділів дипломного проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Тетяна КИСІЛЬ, к.ф-м.н., доцент кафедри КІІС		
Антиплагіат	Андрій НІЧЕПОРУК, доцент кафедри КІІС		

7. Дата видачі завдання «_10_»_____01_____2025 р.

КАЛЕНДАРНИЙ ПЛАН

№з/п	Назва етапів (розділів) дипломного проекту (роботи)	Термін виконання етапів проекту (роботи)	Примітка
1	Вибір напряму дослідження та узгодження тематики кваліфікаційної роботи з керівником	10.01.2025	виконано
2	Ознайомлення з предметною областю; формулювання мети та задач дослідження; визначення об'єкта та предмета дослідження	01.02.2025	виконано
3	Робота над розділом 1 – дослідження предметної області та постановка задачі	01.03.2025	виконано
4	Робота над розділом 2 – Апаратна реалізація комплексу на основі огляду існуючих рішень, реалізація інтерфейсів Bluetooth та WiFi для звязку із застосунком	01.04.2025	виконано
5	Робота над розділами 3 – Програмна реалізація комплексу, проєктування архітектури, структур даних, інтерфейсу ПЗ, інструкція з інсталяції та експлуатації застосунку. Тестування та експлуатація апаратно-програмного комплексу	29.04.2025	виконано
6	Оформлення пояснювальної записки згідно вимог	25.05.2025	виконано
7	Попередній захист ВКР	26.05.2025	виконано
8	Захист ВКР на засіданні ЕК	Червень 2025 року	

Студент _____ Дем`ян ГЕТЬМАН
Підпис Ініціали, прізвище

Керівник роботи _____ Дмитро МЕДЗАТИЙ
Підпис Ініціали, прізвище

№ рядка	формат	Позначення	Найменування	Кіл. листів	№ екз	Примітка
			Текстові документи			
1	А4	КвРКІ .240122.22.05 ПЗ	Пояснювальна записка	87		
			Графічні матеріали			
2	А1	КвРКІ .240122.22.05 Е8	Структура апаратно-програмного комплексу	1		
3	А1	КвРКІ .240122.22.05 Е1	Командна станція. Схема електрична струкурна	1		
4		КвРКІ .240122.22.05 Е8	Діаграма варіантів використання програми	1		
5		КвРКІ .240122.22.05 Е8	Статичне представлення структури моделі програми	1		

Зм	Арк	№ докум	Підпис	Дата	КвРКІ .240122.22.05 ВП		
Розробив		Гетьман		3.06.25	Відомість проекту	Літера	Аркуш 1 / Аркушів 1
Перевір.		Медзатий		16.06.25		У	
Н. контр.		Кисіль		16.06.25	ХНУ, КІ2с-22-1		
Затв.		Павлова		16.06.25			

АНОТАЦІЯ

Тема кваліфікаційної роботи: «Апаратно-програмний комплекс дистанційного керування залізничним макетом на мікроконтролерах з мультиплатформною програмною підтримкою».

Автор роботи: Дем`ян ГЕТЬМАН

Керівник роботи: Медзатий Дмитро Миколайович

Пояснювальна записка: 87 с., 27 рис., 5 табл., 6 дод., 36 джерел.

Графічна частина: 2 креслення формату А1

ЗАЛІЗНИЧНИЙ МАКЕТ, КОМАНДНА СТАНЦІЯ, МІКРОКОНТРОЛЕР, DCC, XPRESSNET, FLUTTER.

Об'єктом розробки в даному дипломному проєкті є апаратно-програмні засоби керування залізничним макетом, основою якого є командна станція на мікроконтролерах та мультиплатформний додаток для дистанційного контролю.

Мета проєкту — вибір апаратних засобів та реалізація додатку з використанням Flutter, адаптивне макетуванні інтерфейсу, реалізація підтримки усіх популярних платформ.

Метод розробки — низькорівнева мова програмування Arudino, програмний каскад Flutter на мові програмування Dart, JSON серіалізація і десеріалізація.

Результат роботи — отримання готового до впровадження продукту для дистанційного керування залізничим макетом.

Підпис студента

30.05.2025

Дата

ЗМІСТ

ВСТУП

Всім відомо, яке враження у людей викликають макети – зменшені в масштабі копії. У дорослих привертають пильну увагу та дають можливість відчути себе Гуллівером, у дітей вони викликають справжній захват. Особливе місце в створенні макетів займає саме залізничне моделювання.

Залізничне моделювання сприяє популяризації технічної творчості серед учнівської та студентської молоді, закріпленню та конкретизації знань і вмінь з фізики, електроніки та програмування, естетичне виховання та прищеплення трудових навичок учням, психологічна реабілітація дітей та дорослих, профорієнтаційна робота.

Часто при створенні такого макету виникає бажання «оцифрувати» його, тобто зробити процес максимально інтерактивним та живим. Наприклад, додати штучне освітлення до вагонів та оточення, озвучити локомотиви, стукіт коліс, автоматизувати шлагбауми та семафори.

Для оцифрування макету використовують спеціалізовані декодери. Вони працюють на прийнятому стандарті DCC – Digital Command Control. Колекціонер може легко придбати їх та встановити на свої моделі, проте способи керування та зв'язку із цими декодерами залишаться для нього консервативними. Зазвичай, це прості малофункціональні дротові пульти керування, що не є зручним способом для частого користування та гри, адже виникає багато труднощів із проведенням дротів, підключенням великої кількості користувачів тощо.

Основні переваги цифрового управління макетом:

– можливість роздільного управління локомотивами на одному шляху. В аналогових системах локомотиви керувалися виключно напругою, і як тільки на шлях подавалося напруга – все локомотиви практично одночасно починали рух відповідно до полярністю і величиною цієї напруги;

– неймовірна плавність руху моделі на дуже малих швидкостях, недосяжна при аналоговому управлінні;

– з використанням цифрового управління стало можливим не тільки управління рухом локомотива, а й світловими приладами, внутрішнім

освітленням, зчіпками моделі, пантографами, парогенератором, і звичайно звуком;

– з'явилася можливість зупиняти локомотиви (склади) НЕ знеструмлюючи рейки, а значить стало можливим в стоячому локомотиві включити світло або активувати будь–яке доп. обладнання, звукової декодер в стоячому локомотиві продовжує озвучувати двигун;

– щоб побудувати цифровий макет не потрібно знань в електроніці, макет будується з готових декодерів і модулів;

– цифровий макет набагато легше автоматизувати і змінювати схеми руху. Для внесення змін до схеми руху готового макету досить поміняти тільки настройки в програмі управління, нічого не змінюючи в електричній схемі макета.

Об’єктом розробки в даному дипломному проєкті є апаратно–програмні засоби керування залізничним макетом, основою якого є мультиплатформний додаток для дистанційного контролю.

Мета проєкту – вибір апаратних засобів та реалізація додатку з використанням Flutter, адаптивне макетуванні інтерфейсу, реалізація підтримки усіх популярних платформ.

Метод розробки – низькорівнева мова програмування Arudino, програмний каскад Flutter на мові програмування Dart, JSON серіалізація і десеріалізація.

Результат роботи – отримання готового до використання продукту для дистанційного керування залізничим макетом.

Актуальність дослідження. Кіберфізичні системи заповнюють практично всі сфери нашого життя. Це і розумні будинки, розумні виробництва та мережі, безпілотний транспорт та транспортні мережі і навіть розумні міста та ін. У даному дослідженні ми розглянемо один із типів кіберфізичних систем.

1 ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Змістовний аналіз предметної області, її структурних та функціональних особливостей

Залізничний моделізм – це хобі, що полягає в колекціонуванні масштабних залізничних моделей та їх аксесуарів (колії, будівлі, семафори тощо); відтворення в мініатюрі історичних станцій, залізничних ліній; створення моделей локомотивів і вагонів. Особливо популярним це хобі є в Німеччині, Японії та США, адже ці країни одні з найперших, хто розпочав серійне виробництво моделей для створення залізничного макету, рисунок 1.1.

У 1954 році міжнародна організація любителів-моделістів залізниць Modelleisenbahn Verband визначила єдині масштаби для моделювання. І якщо раніше не було загальних стандартів, то з цього часу вводиться суворе дотримання норм: ширина колії, параметри залізничних рейок і стрілочних переводів, габарити рухомого складу. Сьогодні звід правил і норм при будівництві модельної залізниці, лише трохи поступається за складністю технічної документації справжніх потягів [1].

Керування залізничиними макетами поділяється на аналогове та цифрове.

Аналогове керування передбачає зміну напруги на колії, яка визначає швидкість руху локомотива. Чим більша напруга, тим швидше він рухається. Швидкість зменшується шляхом зниження напруги. Зміна полярності призводить до зміни напрямку руху. Аналогове керування називається постійним струмом або DC, оскільки більшість модельних залізниць використовували постійний струм [2].

Рисунок 1.1 – Приклад зовнішнього вигляду залізничного макета

Головна перевага цифрового керування в порівнянні з аналоговою системою управління – це незалежне індивідуальне управління кожним локомотивом незалежно від його місцезнаходження на макеті. В аналогових системах для незалежного керування кількома локомотивами шлях розбивається на електрично ізольовані ділянки, що керуються з різних пультів.

Також командна система, що управляє всіма моделями та пристроями макета по одній шині, дозволяє суттєво спростити схему електричної проводки макета.

Коріння цифрового керування залізничними макетами сягає сорокових років 20 століття, коли компанія Lionel Trains (США) представила на ринок двохканальну систему з управлінням по частоті. На жаль, ця система, незважаючи на її очевидну новизну та технологічний пріоритет, не була настільки надійною, як це хотілося б.

У ранніх 60-х роках минулого століття General Electric (GE) представила 5-канальну систему ASTRAC для керування більш ніж одним поїздом на одній ізольованій ділянці.

У 1979 році системи Dynatrols Carrier Control та CTC-16 були представлені журналом Model Railroader. Обидві були першими системами, що дозволяли керувати окремим поїздом з достатнім ступенем надійності. Проблеми, пов'язані з цими двома системами, мали інше джерело, ніж якість управління – обидві були сумісні друг з одним і з іншими системами.

Початок 80-х ознаменувався проривними рішеннями в мікроелектроніці, що різко знизило собівартість застосування цифрових технологій та розширило на порядок сферу їх застосування. Незабаром з'являються перші цифрові системи DCC.

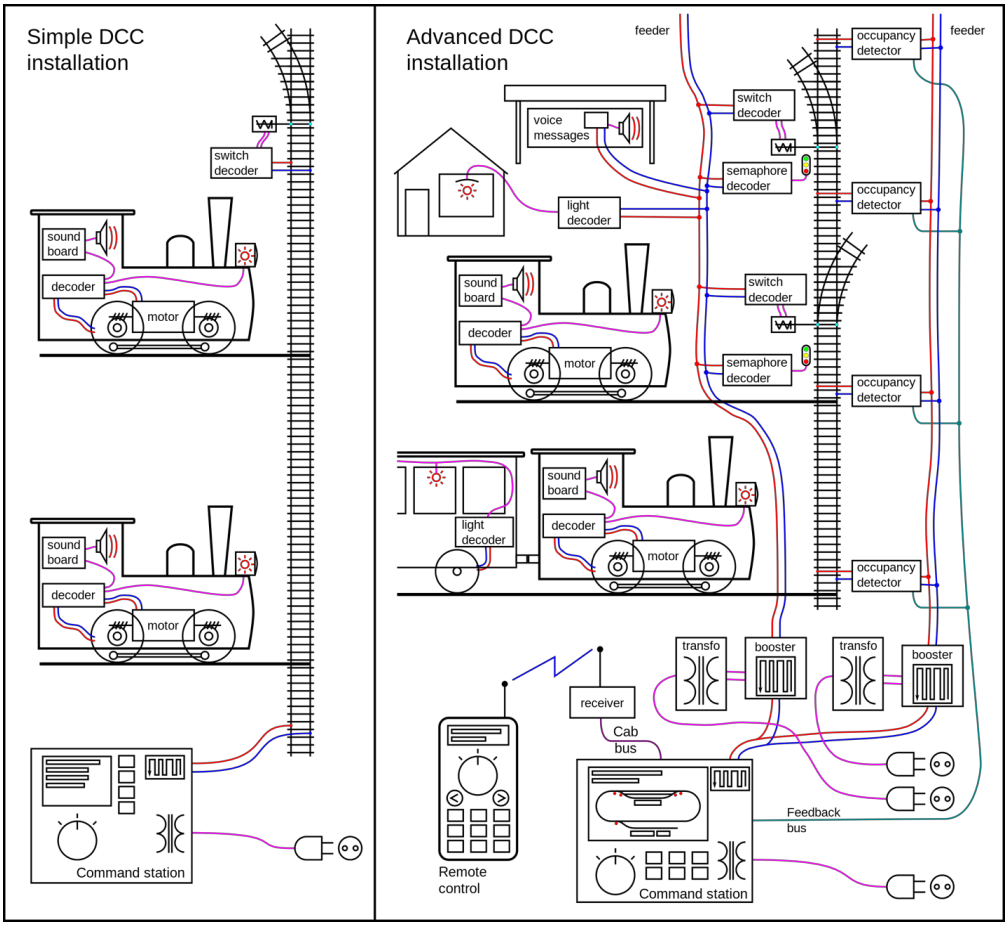


Рисунок 1.2 – Схематична реалізація цифрового керування через протокол DCC

Системи "Digital Command Control" (DCC) були спроєктовані німецькою компанією "Lenz Elektronik GmbH" у 1980-х роках для двох фабрик, що випускали моделі локомотивів, "Märklin" та "Arnold". Перший цифровий декодер, вироблений «Lenz», вийшов на ринок у 1989 для моделей Arnold (типорозміру N) і в середині 1990 року для «Märklin» (типорозмірів Z, H0 і N) [3].

Протокол DCC складається з двох стандартів, опублікованих Національною Асоціацією Моделів Залізниць (NMRA), яка запропонувала стандарт сигналу цифрового управління та об'єднала процес розробки подальших стандартів у рамках власної організації, рисунок 1.2. З 1994 року, коли перший бюлетень про DCC було опубліковано, почалася нова ера у житті домашніх магістралей. Робоча група DCC у рамках NMRA сьогодні інтенсивно працює над покращенням існуючих стандартів та розробкою нових [4].

1.2 Аналіз сучасного стану розвитку залізничного моделювання в Україні та світі

1.2.1 Ситуація в Україні

В Україні залізничне моделювання залишається нішевим хобі, але має активне ядро ентузіастів. Кількість моделістів відносно невелика порівняно із Західною Європою чи США, проте спостерігається згуртована спільнота. У великих містах діють клуби залізничних моделістів. Наприклад, у Києві з 2007 року існує «Український клуб залізничного моделювання» (об’єднання «Київський модуль»), що регулярно організовував зустрічі та виставки Train Hobby Days, демонструючи великі модульні макети. Такі заходи збирали сотні відвідувачів і навіть встановлювали національні рекорди (наприклад, у 2017 році було представлено найбільший діючий макет поїзної мережі довжиною понад 50 м). Окрім столиці, осередки хобі є в Одесі, Львові, Харкові та інших містах – ентузіасти об’єднуються через інтернет-форуми, Facebook-групи (зокрема спільноти за масштабами N (1:160) чи H0 (1:87)) і проводять локальні виставки.

Загалом популярність хобі в Україні невисока, проте зацікавлення зберігається серед тих, хто захоплюється залізницею, історією техніки або моделізмом в цілому [5].

Українські моделістичні магазини пропонують продукцію світових брендів (PIKO, Roco, Märklin, Bachmann тощо), включаючи цифрові DCC-системи. Спеціалізовані інтернет-магазини (наприклад, MelnykModels, ModelKits) забезпечують доступ до локомотивів, вагонів, рейкового матеріалу та електроніки для DCC. Однак вартість імпортних моделей і цифрових контролерів є досить високою для середнього споживача, що обмежує поширення хобі. Деякі ентузіасти в Україні використовують DIY-рішення: наприклад, встановлюють самостійно DCC-декодери, будують контролери на базі Arduino, освоюють відкриті протоколи (такі як XpressNet від Lenz або Loconet від Digitrax) для керування макетами [6-8].

1.2.2 Ситуація на західних ринках (Європа, США, Канада)

У Західній Європі та Північній Америці залізничний моделізм має давні традиції, але галузь стикається зі змішаними трендами. Хобі залишається популярним серед значної частини старшого покоління: у клубах Німеччини, Великої Британії, США середній вік учасників часто перевищує 50–60 років. Зменшення кількості молоді в хобі викликає занепокоєння щодо поступового скорочення аудиторії. У Великій Британії, наприклад, у 2023 році було оголошено про завершення діяльності знаменитої щорічної виставки Warley National Model Railway Exhibition – організатори послалися на старіння членів клубу та складність залучення молодих волонтерів. Також закриваються деякі легендарні модельні магазини (як-от Hattons Model Railways у 2024 р.), що пояснюється зміною структури споживачів, падінням відвідуваності та конкуренцією інтернет-продажів.

Попри виклики, ринок на Заході залишається активним: провідні виробники продовжують випуск нових моделей і інновацій. Особливо сильні позиції в

Європі, де є велика спільнота колекціонерів та моделістів. Німеччина – один з центрів хобі – забезпечує понад половину європейського ринку; тамтешня компанія Märklin відзначила у 2022–2023 рр. зростання продажів до рекордних ~127 млн євро, завдяки лояльності давніх фанатів та появі нових захоплених «kidults» (дорослих, що купують моделі для себе).

Виробники активно впроваджують мобільні додатки та бездротові контролери для керування моделями. Наприклад, система Z21 (Roco) дозволяє керувати поїздами через смартфон/планшет по Wi-Fi з інтерфейсом, схожим на геймпад. Lionel у США пропонує для початківців набори LionChief з Bluetooth-контролем (дитина чи дорослий може керувати поїздом з телефону без складної установки). Такі спрощені цифрові рішення допомагають залучити новачків, які звикли до гаджетів. Окрім того, цифрові технології допомагають поєднати реальне хобі з віртуальним досвідом: багато моделістів фотографують та знімають відео своїх мініатюр і діляться онлайн (хештег #modeltrains має сотні мільйонів переглядів у TikTok). Це створює онлайн-спільноти, де досвідчені майстри і молоді фанати обмінюються порадами, що також підтримує інтерес до хобі.

Таким чином, цифрові технології – від DCC-протоколів до соціальних медіа – стали ключовим фактором збереження і оновлення залізничного моделізму в сучасному світі.

1.2.3 Аналіз ситуації в світових наукових колах

Залізничний моделізм – це не лише хобі, але й предмет наукових досліджень, особливо в контексті сучасних цифрових технологій, таких як цифрове керування DCC та мікроконтролери. У науковому та освітньому середовищі DCC розглядається як приклад:

- розподіленого керування (distributed control systems);
- промислової автоматизації (особливо в мініатюрному масштабі);
- вбудованих систем (embedded systems);
- цифрової модуляції сигналів (маніпуляція, протоколізація, декодування).

Університетські лабораторії іноді використовують DCC для викладання основ цифрової обробки сигналів, розробки ПЗ для мікроконтролерів та розумних транспортних систем.

У сучасному моделізмі широко застосовують мікроконтролери – AVR, PIC, Arduino, STM32, ESP32 тощо. Мікроконтролери застосовуються для:

- керування аксесуарами (світлофори, стрілки, бар'єри, будівлі);
- телеметрія: зчитування швидкості, положення, струму;
- бездротове керування через Wi-Fi, Bluetooth (наприклад, через MQTT або web-інтерфейси);
- інтеграція з ПК: автоматизація макета через програми типу Rocrail, JMRI.

Дослідження також охоплюють:

- симуляцію руху поїздів у масштабі – для тестування алгоритмів керування;
- оптимізацію розкладів руху та логістики;
- штучний інтелект у моделюванні трафіку на макетах (наприклад, для автоматичного розведення маршрутів).

Наукові статті про залізничний моделізм з DCC та мікроконтролерами можна знайти в галузі:

- Embedded Systems Design;
- Automation and Control Engineering;
- Educational Robotics and Simulation.

1.2.4 Наукові публікації і технічні дослідження, що стосуються предметної області

У статті [9] описано інтеграцію системи цифрового керування DCC з промисловим програмним комплексом SIMATIC S7-1200 для управління моделлю

залізниці. Система забезпечує збір даних у реальному часі, контроль стрілок, сигналів та переїздів, а також взаємодію з диспетчерською системою ILTIS-N через REST API.

В роботі [10] дослідження присвячене розробці системи управління моделлю поїзда за допомогою мікрокомп'ютера, зокрема Arduino та Raspberry Pi. Розглядаються аспекти цифрового керування, включаючи порівняння аналогових та цифрових методів управління, а також впровадження інтелектуальних підсилювачів для DCC-мереж.

Використання бездротових команд DCC, яке розглянуте в [11], демонструється передача DCC-команд бездротовим шляхом за допомогою Arduino та модулів nRF24L01. Такий підхід дозволяє керувати локомотивами без необхідності живлення через рейки, що відкриває нові можливості для моделювання.

Патент на DCC-декодер для моделей залізниць [12] описує простий DCC-декодер, який використовує мікроконтролер PIC16CE625 для обробки цифрових сигналів та управління моделлю поїзда.

Верифікація систем залізничної сигналізації, приведено в [13] реалізується за допомогою інструменту NuSMV. Хоча дослідження зосереджене на реальних залізничних системах, методи можуть бути адаптовані для моделювання та тестування моделей залізниць.

Колії в Лабораторії керування залізничними транспортними засобами факультету бізнесу та економіки Університету Менделя в Брно можна класифікувати як великі; тому потрібна спеціальна система.

Цифрове керування модельною залізничною колією вимагає керування рухом транспортних засобів та їх функціями (звуки, освітлення), а також керування елементами колії (стрілки, сигнали, датчики зайнятості тощо). Хоча керування транспортними засобами забезпечується стандартом, ункціональність якого цілком адекватна для нормальної експлуатації, керування елементами колії може здійснюватися кількома способами.

Зокрема в роботі [14] розроблено новий протокол для шини RS485, розроблено апаратне та прошивне забезпечення нового GPIO-ведомого модуля MTB-UNI v4, розроблено новий головний модуль RS485 MTB-USB та розроблено два інших нових апаратних модулі, включаючи прошивки для їхніх мікроконтролерів. Було розроблено комп'ютерний додаток MTB Daemon та бібліотеку для доступа до шини. Результатом є система, яка відповідає відносно високим вимогам і дозволяє подальший розвиток і використання в лабораторії.

Що стосується створення спеціалізованих застосунків для керування залізничним макетом, то, наприклад стаття [15] пропонує поглиблений аналіз Flutter, популярного кросплатформного фреймворку, розробленого Google для розробки мобільних додатків.

Flutter – це набір інструментів Google для створення інтерфейсу користувача для створення власно компільованих додатків для мобільних пристроїв, веб-сайтів та настільних комп'ютерів з єдиної кодової бази. У цій статті досліджується архітектура, функції, філософія дизайну Flutter та те, як він спрощує розробку додатків. Розглядаються кросплатформні можливості Flutter, його переваги та обмеження, а також його майбутнє в галузі розробки програмного забезпечення.

В епоху, коли смартфони стали звичайним явищем, і, здається, кожен має їх, питання, яке ставить перед собою більшість компаній, звучить так: як нам створювати дивовижні додатки з невеликими витратами та без особливих клопотів. У статті [16] досліджується Flutter та його рішення для цієї проблеми розробки від Google. За допомогою практичних прикладів, а також контекстуального розуміння, демонструється, як розробники використовували Flutter для створення додатків для iOS, Android і навіть настільних комп'ютерів – все з єдиною кодовою базою. Дослідження виходить за рамки стандартних технічних нагород і розглядає вплив Flutter на структуру проектів: як він революціонізує міжкомандну співпрацю, його найкращі якості, а іноді й найбільші недоліки.

Ґрунтуючись на реальних працях розробників та існуючій динаміці на ринку, пропонується досить перспективну точку зору на те, що Flutter означає для майбутнього розробки додатків, особливо в контексті зміни мобільних тенденцій [17, 18].

1.3 Визначення вимог до апаратно-програмного комплексу та розробка технічного завдання

1.3.1 Постановка задачі

Відповідно до завдання на дипломний проєкт необхідно розробити апаратно-програмний комплекс, призначений для дистанційного керування залізничними макетами масштабів H0 (1:87), TT (1:120), N (1:160). Комплекс включає:

- апаратне забезпечення – цифрова командна станція стандарту DCC з інтерфейсами WiFi та Bluetooth;
- програмне забезпечення – мультиплатформний застосунок для операційних систем Windows, Linux, Mac OS, iOS, Android для дистанційного керування залізничним макетом через командну станцію.

1.3.2 Вихідні дані до апаратної частини комплексу – командної станції

Апаратна частина комплексу складається з командної станції та декодерів. Командна станція повинна мати такі технічні та функціональні можливості:

- підтримка стандартів та протоколів– DCC, Xpressnet, X-bus, S88;
- підтримка інтерфейсів – Bluetooth, WiFi; Lokmaus, Multimaus;
- максимальний вихідний струм бустера – 3 А, захист від коротких замикань бустера;
- одночасна підтримка 16 локомотивів в адресах від 1 до 9999;
- підтримка 1024 аксесуари (стрілки та сигнали); можливість підключення пристроїв XpressNet типу Lokmaus, Multimaus або аналогічних;

- програмування та читання DCC-декодерів у режимах Direct, Paged, Register і PoM;
- підтримка 14, 28 та 128 ступенів швидкості локомотивів;
- підтримка функцій керування світлом (FL) і F1 до F12 для кожного локомотива;
- підтримка 128 входів зворотного зв'язку з модулями S88.

1.3.3 Вихідні дані до програмної частини комплексу – мультиплатформного додатку

Програмна частина комплексу складається з мультиплатформного застосунку, який здійснює дистанційне керування макетом і може бути встановлений на телефон, планшет, ноутбук, перональний компютер з різними операційними системами. Застосунок повинен мати такий функціонал:

- реалізація функцій командної станції на програмному рівні;
- мультиплатформність (підтримка операційних систем Windows, Linux, Mac OS, iOS, Android);
- реалізація протоколу XpressNet між застосунком та комендною станцією;
- дружній та зручний інтерфейс в Material Design стилі;
- підтримка локалізації;
- можливості користувацької візуальної кастомізації.

1.3.4 Основні принципи функціонування комплексу

Сигнал DCC подається на рейки залізничної колії. Локомотивні декодери, встановлені в рухомому складі, знімають і використовують сигнал DCC для керування швидкістю, напрямком руху та фарами локомотива. Аксесуарні декодери також можуть бути підключені для керування стрілками, світлофорами та іншими аксесуарами.

Користувач використовує телефон, планшет, ноутбук, перональний компютер, а також звичайний пульт, оснащений цифровою клавіатурою та іншими кнопками керування. Інформація з керуючого засобу надсилається на командну станцію DCC, яка перетворює її в керуючу інформацію відповідно до протоколу.

Стандарти DCC визначають рівні напруги, тривалість імпульсів та цифрові формати інформації. Це означає, що декодер від будь-якого виробника відповість на команди та сигнали програмування з систем DCC, вироблених будь-яким іншим виробником.

Кожен локомотив на макеті має унікальну адресу свого мобільного декодера. Використовуючи застосунок (пульт), оператор може вибрати будь-який локомотив на трасі за його адресою та змінити його швидкість, напрямок, освітлення та інші функції. DCC дозволяє декількома пультами одночасно контролювати різні локомотиви. Невибрані локомотиви продовжують працювати за останніми налаштуваннями.

Декодер DCC має схему живлення, яка перетворює сигнал DCC у джерело живлення постійного струму для двигуна та освітлення. Декодер також має цифрову логіку, яка використовує інформацію сигналу живлення для керування локомотивом.

Декодери налаштовуються встановленням так званих конфігураційних змінних або CV (configuration variable). Сьогодні деякі декодери дозволяють запрограмувати значення CV, коли поїзд працює на основній доріжці. Одним з винятків є те, що значення CV, що вказують на унікальну адресу локомотива, не можуть бути змінені під час роботи; адреси можна змінити лише на секції програмування (так званий програмний трек) [4].

Отже, для реалізації поставленого завдання на дипломний проєкт необхідний такий набір складових:

- цифрова командна станція;
- локомотивний декодер в кожному локомотиві;
- аксесуарні декодери для стрілок, світлофорів, шлагбаумів;

- функціональні декодери для вагонів, локомотивів;
- модулі зворотнього звязку S88 для відслідковування положення рухомого складу на макеті;
- додаткові декодери, такі як декодери залізничного переїзду, поворотного круга, розворотної петлі.

Структура комплексу приведена в графічній частині дипломного проєкту.

Основним компонентом системи DCC є командна станція, яка несе відповідальність за створення сигналу DCC, який, в свою чергу, містить команди DCC для керування будь-яким декодером. Підсилювач (бустер) приймає цей сигнал DCC і додає йому "потужність", щоб зробити його зручним для керування рухомим складом та двигунами. Декодер слухає та відповідає на команди DCC, відправлені командною станцією, і керує швидкістю та напрямком локомотива незалежно від інших пристроїв на трасі.

В результаті реалізації такої системи отримуємо можливість:

- кожен поїзд індивідуально контролюється з центру управління;
- ми можемо мати команду для кожного локомотива або змінити локомотив від команди;
- не потрібно ізолювати ділянки рейок, щоб зупинити поїзд, а інші продовжують рух;
- нові елементи керування можна підключені до шини Xpressnet.

2 АПАРАТНА РЕАЛІЗАЦІЯ КОМПЛЕКСУ

2.1 Огляд існуючих рішень промислових командних станцій

Розглянемо існуючі рішення промислових командних станцій світових виробників цифрового обладнання для залізничних макетів.

На рисунку 2.1 зображені одні з найбільш популярних командних станцій провідних світових виробників такого обладнання [19].

Рисунок 2.1 – Командні станції найбільш популярних виробників

У кожній командній станції є пульт управління (один або кілька), за допомогою якого ми можемо управляти макетом. Для того, щоб змінити, наприклад, швидкість одного з локомотивів, необхідно на пульті ввести його

адресу і встановити необхідну швидкість, при цьому швидкість руху інших локомотивів залишиться незмінною.

Цифрова командна станція може керувати кількома десятками локомотивів по одній парі рейок одночасно, тобто весь рухомий склад може знаходиться на одному і тому ж ділянці шляху. Основний вихід станції, до якого підключаються рейки зазвичай називають main track.

Варто відзначити один важливий параметр станції – цифровий протокол передачі, він визначає набір правил посилки команд декодерів. Щоб декодер міг виконувати команди, станція і декодер повинні працювати в одному протоколі. У світі набули поширення такі типи протоколів: DCC, Marklin-motorola (MFX), Selectrix і FMZ. На сьогоднішній момент DCC практично витіснив інші протоколи, завдяки відкритій документації і тій обставині, що багато великих виробників роблять електроніку, що працює саме в цьому форматі [5].

Однією з найпоширеніших командних станцій на європейському ринку є Roco multiMAUS, рисунок 2.2.

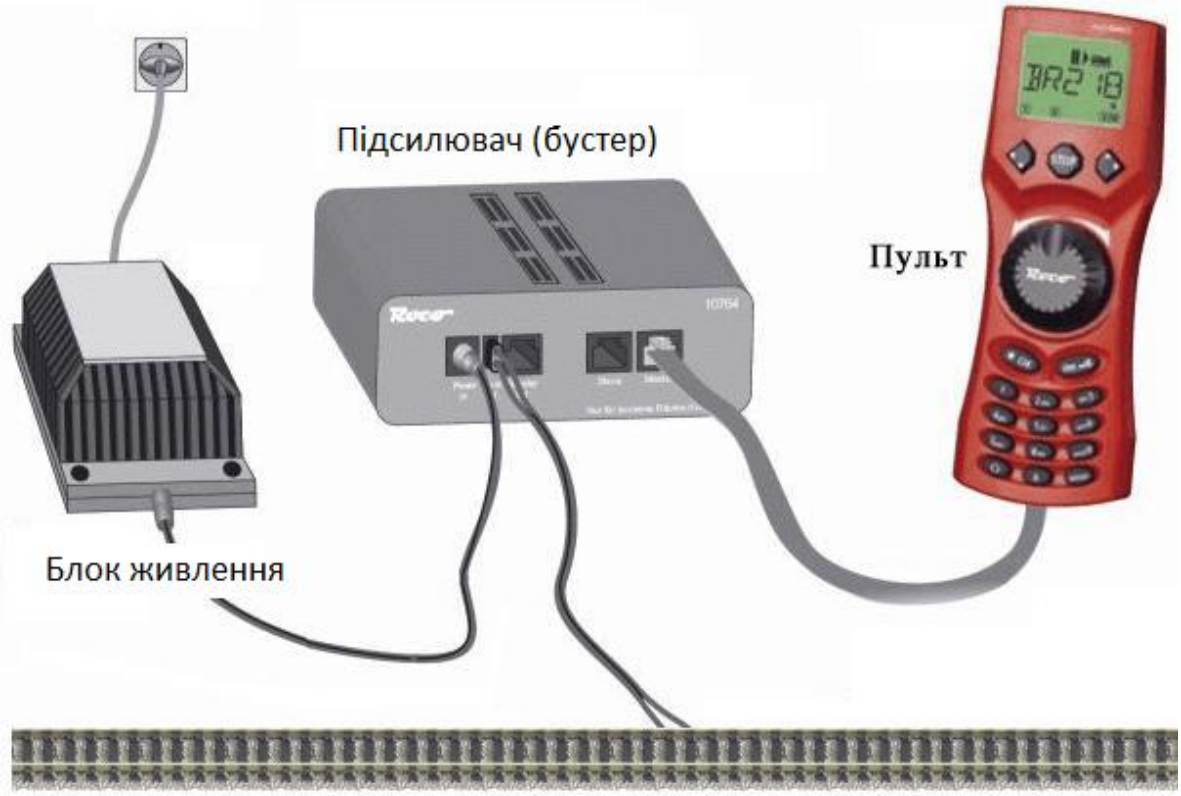


Рисунок 2.2 – Командна станція Roco multiMAUS

Вона набула такого поширення завдяки високій надійності, простоті експлуатації, хорошими можливостями для подальшої автоматизації макета і низькою ціною.

Кожна командна станція має своєму складі підсилювач потужності цифрового сигналу – так званий бустер. Бустер – пристрій, що приймає "пакети" коду від командної станції, і забезпечує електричну потужність, достатню для "донесення" цих "пакетів" через рейки до всіх адресатів на макеті. Одночасно струм від бустера, що містить в собі посилений цифровий сигнал, використовується для живлення двигунів локомотивів, електромагнітних приводів аксесуарів та інших споживачів електрики. Основною характеристикою бустера є його "потужність", тобто сила вихідного струму в амперах, яку він може забезпечити при фіксованому вихідному напрузі. Як правило, цей показник знаходиться в межах від 1,5 до 10 Ампер. Потрібна сила струму залежить від того, якого масштабу залізниця, а також від того, скільки локомотивів одночасно по ній запускати. Якщо виявиться, що потрібна сила струму більше, ніж може забезпечити якийсь конкретний бустер, то більшість систем допускає підключення декількох окремих бустерів.

Незважаючи на те, що багато запропонованих на ринку комплектів апаратури DCC об'єднують в собі командну станцію і бустер в одному корпусі, в комплект не завжди включається джерело живлення, що ставить користувача перед необхідністю нести додаткові витрати. При цьому кожна конкретна система DCC має свої власні вимоги до джерела живлення. Якщо вам доведеться підбирати окреме джерело живлення до вашого нового комплекту апаратури DCC, настійно рекомендується точно дотримуватися всі вказівки виробника апаратури, що стосуються необхідних характеристик такого джерела живлення.

Пульт управління (the cab) або так звана "ручка" (в дослівному перекладі "throttle" – педаль "газу") являє собою інтерфейс, з якого на макет віддаються команди. З його допомогою ви керуєте швидкістю, звуковими та світловими ефектами ваших поїздів. "Ручка" може також управляти аксесуарами, такими як стрілки або сигнали.

В системі Bachmann Dynamis, рисунок 2.3, реалізовано двосторонній бездротовий зв'язок між пультом управління і командної станцією за допомогою інфрачервоного випромінювання. Інфрачервоний сигнал приймається "в межах прямої видимості", тому для стійкого зв'язку необхідно направляти випромінювач бездротового пульта управління в сторону інфрачервоного приймача командної станції з ефективним сектором прийому сигналу в 180 градусів.

Рисунок 2.3 – Комплект Dynamis компанії Bachmann

Пульт управління системи Dynamis нагадує джойстик ігрової відеоприставки. Він скомпонований "горизонтально" (ширина більше глибини) і оснащений функціональними кнопками, а також рукояткою управління швидкістю локомотива. Великий рідкокристалічний екран служить для індикації різних характеристик локомотива, яким в даний момент керує пульт, а також для

виведення різних меню програмування. Цей стартовий комплект розрахований на використання тільки одного пульта управління.

В основу системи EasyDCC, рисунок 2.4, покладена командна станція у виконанні, який передбачає монтаж на макет. У неї інтегровані два пульта управління. Станція оснащена також функціональними кнопками і рідкокристалічним індикатором. Ви можете розширювати можливості системи за допомогою додаткових бустерів, що вмонтовуються на макет панелей з додатковими портами шини периферії управління, додаткових окремих пультів управління, як дротових (модель XR1300), так і радіо. Поряд з комплектом BSS (Basic Starter Set) виробник пропонує також два варіанти "бездротового" стартового комплекту WSS (Wireless Starter Set). У першому варіанті такий комплект включає в себе, крім командної станції, безпровідний радіопульт модели RF1300, а в другому – більш складний "флагманский" радіопульт модели T9000E.

Рисунок 2.4 – Комплект EasyDCC BSS4 компанії CVP Products

2.2 Особливості стандарту DCC (Digital Command Control)

2.2.1 Властивості системи DCC

Система DCC є завадостійкою системою. У разі зникнення надійної передачі сигналу (наприклад, поганий контакт на рейках), декодер виконуватиме останню передану на нього команду, доки не зможе гарантовано отримати нову.

Система DCC є відкритою системою. Тобто її стандарти опубліковані і будь-який пристрій, виконаний за специфікаціями DCC, працюватиме з іншими пристроями інших виробників. Це правило не поширюється на розширену функціональність конкретного виробника, яка може бути реалізована поза стантартом. У такому випадку до неї існують основні вимоги – не перешкоджати функціонуванню пристроїв, виконаних за стандартом і не вносити спотворення сигналу DCC.

Система DCC є системою, що нарощується. Максимальна кількість різних її компонентів, як правило, обмежується конкретним виробником апаратури керування. Однак, у межах обмежень, кількість компонентів (декодерів, станцій команд, підсилювачів) може бути будь-якою.

Система DCC підтримується рядом стандартів, які в даний час розвиваються та розширюються. У той же час ця система гарантує сумісність пристроїв ранніх стандартів з наступними.

Таким чином, DCC є наочним прикладом грамотного перспективного планування. Це й зумовило масову підтримку цього стандарту багатьма виробниками, появу нових виробників, нових продуктів, і, як наслідок, забезпечило доступність DCC для дедалі більшого кола споживачів.

2.2.2 Вимоги до електричного сигналу

Електричний сигнал на коліях макета відповідно до стандарту DCC повинен мати такі основні характеристики:

- електроживлення рухомого складу і додаткових пристроїв, яке повинно забезпечуватися всіма керуючими пристроями і декодерами, реалізується шляхом випрямлення по мостовій схемі;
- амплітуда керуючого сигналу не повинна перевищувати ± 22 В;
- мінімальна амплітуда цифрового сигналу повинна складати ± 7 В;
- декодери для типорозмірів N і менше повинні бути стійкі до постійної напруги не менше 24 В;
- декодери для типорозмірів ТТ і більше повинні бути стійкі до постійної напруги не менше 27 В;

Загальний вигляд електричного сигналу на коліях макета відповідно до стандарту DCC приведений на рисунку 2.5.

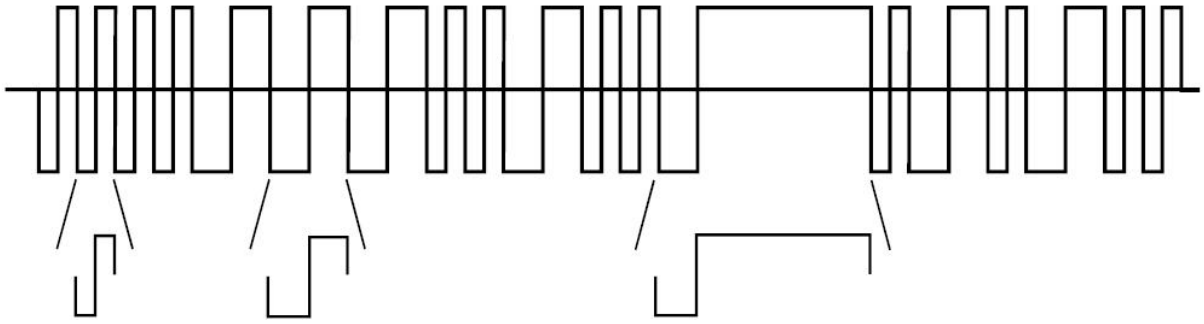


Рисунок 2.5 – Загальний вигляд DCC сигналу

2.2.3 Принцип роботи сигналу DCC

Прямокутний сигнал двох різних тривалостей передає два стани сигналу, які є одиницею або нулем відповідно. Крім того, прямокутна форма сигналу призводить до того, що на рейках завжди є напруга однакової амплітуди, але різної полярності. Цей факт використовується для схем живлення декодерів та

інших компонентів. Тобто, реалізуються в одному ланцюгу функції живлення компонентів та передачі цифрового сигналу, рисунок 2.6.

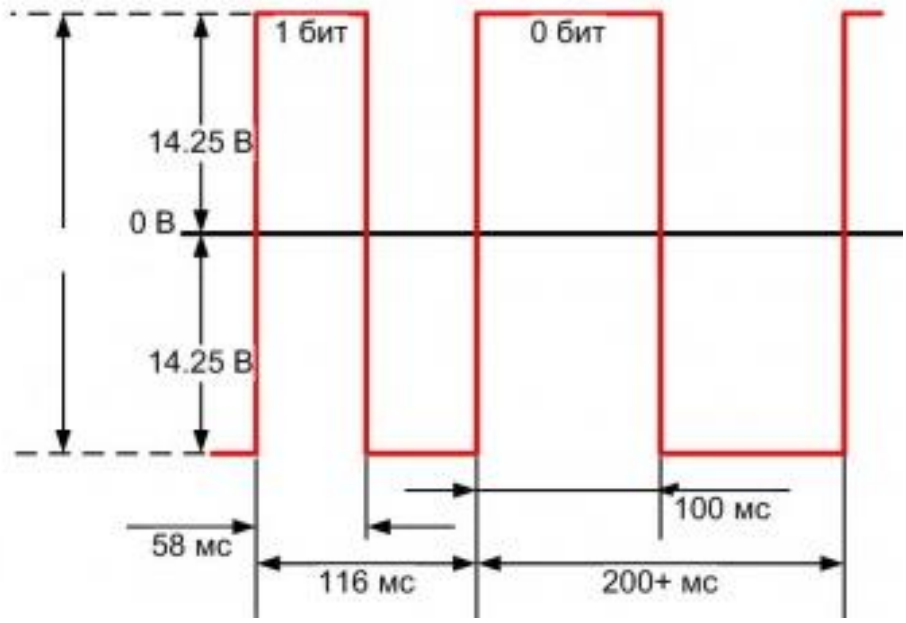


Рисунок 2.6 – Кодування логічних рівнів в DCC

Така організація відмінно підходить до, існуючого багато десятків років, класичного управління моделями через ланцюг з використанням двох рейок і струмознімачів на візках споживачів, таких як локомотиви, вагони з електропроводкою та стаціонарні пристрої з керуванням через ланцюг рейок.

Основною одиницею передачі в DCC служить пакет. Пакет складається з послідовності байтів, що описують поточну команду, обрану адресу та дані самої команди. Порядок проходження інформації жорстко визначено стандартом DCC і називається форматом пакета. Немає жодного спеціального механізму в стандарті DCC, що гарантує підтвердження про доставку пакета. Апаратура DCC "сподівається", що цей пакет буде отримано адресатом. За принципом роботи, цей протокол нагадує ідею низькорівневого транспортного протоколу Інтернет UDP, пакети якого також не мають механізму гарантованої доставки.

З урахуванням специфіки управління пакетами, новітня апаратура DCC може мати параметр кратності відправлення пакетів адресату, що збільшує «шанси» його доставки.

Іншим наслідком факту відсутності гарантій доставки пакета є правило виконання останньої отриманої та успішно декодованої команди будь-яким декодером.

2.3 Вибір схемотехнічного рішення командної станції на мікроконтролерах

В процесі вибору оптимального схемотехнічного рішення реалізації командної станції та іншого цифрового обладнання для побудови макетів було здійснено аналітичний пошук на відповідних ресурсах [20-23].

За основу було взято командну станцію Nano-X, яка відповідає всім вимогам, поставленим у завданні на дипломний проєкт [23].

Таким чином, вибрана схема командної станції включає шину XpressNet і може контролювати:

– 16 локомотивів одночасно в адресах від 1 до 9999;

– 1024 аксесуари – стрілки та сигнали;

– 31 пристрій XpressNet типу Lokmaus, Multimaus або аналогічні;

– програмування та читання DCC-декодерів у режимах Direct, Paged, Register і PoM.

– підтримує 14, 28 та 128 ступенів швидкості, функції FL і F1 до F12 для кожного локомотива;

– підтримує 128 входів зворотного зв'язку з модулями S88;

– містить вбудований підсилювач (бустер) з максимальним струмом до 3А з вибором напруги живлення від 13 В до 22 В;

– має захист від коротких замикань;

– має зв’язок з компютером через Bluetooth;

– дозволяє підключати до трьох провідних пультів керування через шину X-Bus;

– в залежності від комплектації може мати інфрачервоний канал керування (до 10 метрів) або радіопульт (до 30 метрів).

Командна станція побудована на мікроконтролерах PIC16F648 [24] та вихідному каскаді на IMC L6203 [25].

Мікроконтролер PIC16F648, рисунок 2.7 – це 8-розрядний мікроконтролер з RISC архітектурою. Даний мікроконтролер є дешевим, з високою швидкодією та низьким енерго споживанням. Він має вбудоване ЕППЗУ програми, ОЗУ даних і може випускатися в 18 і 28 вивідних корпусах.

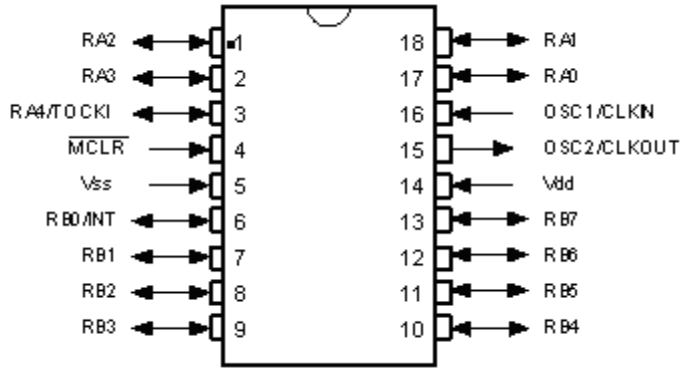


Рисунок 2.7 – УГЗ мікроконтролера PIC16F648

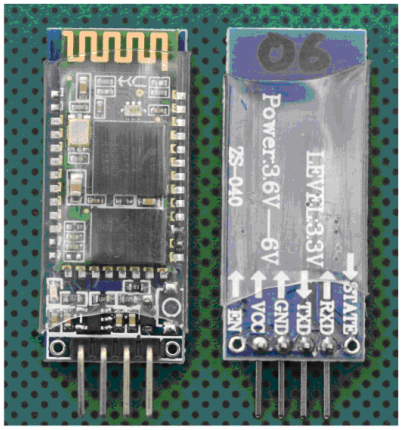
Вихідна напруга бустера може бути встановлена між 12 В і 22 В потенціометром, в залежності від масштабу або потреби користувача. Схема КС містить реле для переключення між основним та програмувальним треком при програмуванні CV.

Зворотній зв'язок підтримується модулями S88 – можна підключити до 8 модулів (128 входів).

Для відображення різних станів командної станції є індикатор, який світиться або блимає відповідно до поточного стану.

2.3.1 Реалізація Bluetooth – з’єднання із застосунком

Для реалізації Bluetooth-з’єднання із застосунком використовується модуль Bluetooth HC-06, зовнішній вигляд якого показаний на рисунку 2.8 [26].



HC-06

Рисунок 2.8 – Зовнішній вигляд модуля BlueTooth HC-06

За замовчуванням у FLASH пам'ять модуля HC-06 записано ПЗ, яке дозволяє зв'язати по радіо Bluetooth телефон, ноутбук тощо будь-яким пристроєм на мікроконтролері, що має TTL-порт UART RS-232. За допомогою пакета CSR CASIRA BLUELAB SDK (в якому є робочі приклади програм Bluetooth) можна самому перепрограмувати модулі і створювати свої власні пристрої Bluetooth. На борту у модуля стоїть чіп пам'яті на 1 мегабайт. Там записано управляюча програма і всі налаштування. На зовнішні 34 контакти модуля виведені:

– апаратний UART, сигнали TXD, RXD, CTS і RTS;

– послідовний порт PCM (для цифрового введення/виводу звуку);

– два аналогових входи/виходи AIO;

– вхід скидання RESET (її можна нікуди не підключати);

– вхід напруги живлення +3.3 В, струм споживання максимум 35 мА;

– інтерфейс USB;

– інтерфейс SPI, через який прошивається ПЗ і відбувається налагодження;

– 12 цифрових порти введення/виведення PIO.

Умовно-графічне зображення модуля BlueTooth HC-06 приведене на рисунку 2.9.

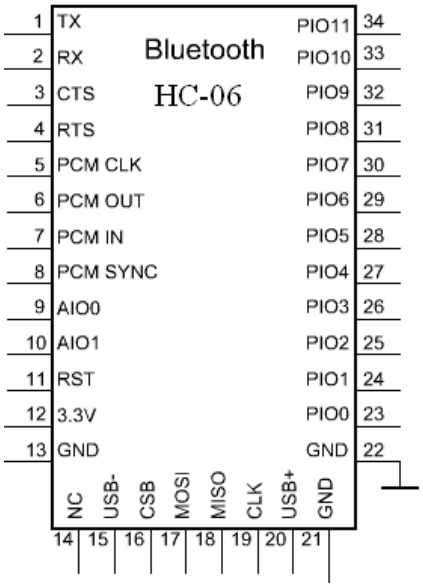


Рисунок 2.9 – Умовно-графічне зображення модуля BlueTooth HC-06

Технічні параметри модуля HC-06:

– використовувані радіочастоти 2.40 .. 2.48 ГГц, діапазон ISM (реєстрація та ліцензування частот не потрібно), рівень потужності Class2 (+6 dBm), чутливість приймача -85 dBm;

– протокол BlueTooth V2.0;

– можливі швидкості передачі даних 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200, 230400, 460800, 921600, 1382400 бод. Швидкість може бути обрана користувачем за допомогою АТ-команд, за замовчуванням швидкість 9600 бод, 8 біт даних, 1 стоп-біт, без біта парності;

– напруга живлення 3.3 В, струм споживання 35 мА в режимі пошуку радіоз'єднання, і близько 8 мА, коли радіоканал встановлений і функціонує, струм в режимі сну - менше 1 мА;

– розміри 28 х 15 х 2.35 мм.

Мініатюрний модуль дозволяє підключитися до телефонів, PDA, смартфонам, ноутбукам, до будь-яких пристроїв, що мають на борту модуль Bluetooth в режимі master, і передавати дані через стандартний RS-232.

2.3.2 Реалізація WiFi – з'єднання із застосунком

Для реалізації WiFi – з'єднання із застосунком застосовується модуль ESP-12F, який є однією з найпопулярніших варіацій таких пристроїв, які виробляються компанією Espressif Systems, рисунок 2.10 [27].

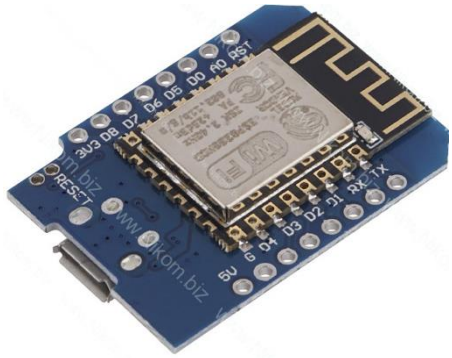


Рисунок 2.10 – Зовнішній вигляд модуля WiFi ESP-12F

Модуль ESP-12F базується на мікроконтролері ESP8266 який, в свою чергу, базується на архітектурі RISC. Ця архітектура характеризується потужним набором інструкцій, більшість з яких виконуються за один тактовий цикл. Це означає, що ESP-12F може забезпечувати високу продуктивність при роботі на низькій тактовій частоті, що дозволяє ефективно використовувати енергію і забезпечувати довготривалу роботу на батарейках.

ESP-12F забезпечує продуктивність до 16 мільйонів операцій в секунду і підтримує флеш-пам'ять програм різної ємності, від 1 до 256 кілобайт. Цей мікроконтролер оптимізований для програмування мовою високого рівня С, що спрощує розробку програмного забезпечення. Особливістю ESP-12F є наявність 10-бітного аналого-цифрового перетворювача (АЦП) з 8 каналами, який дозволяє зчитувати значення аналогових сигналів. Крім того, він підтримує інтерфейс JTAG або debugWIRE, що дозволяє вбудоване налагодження програм.

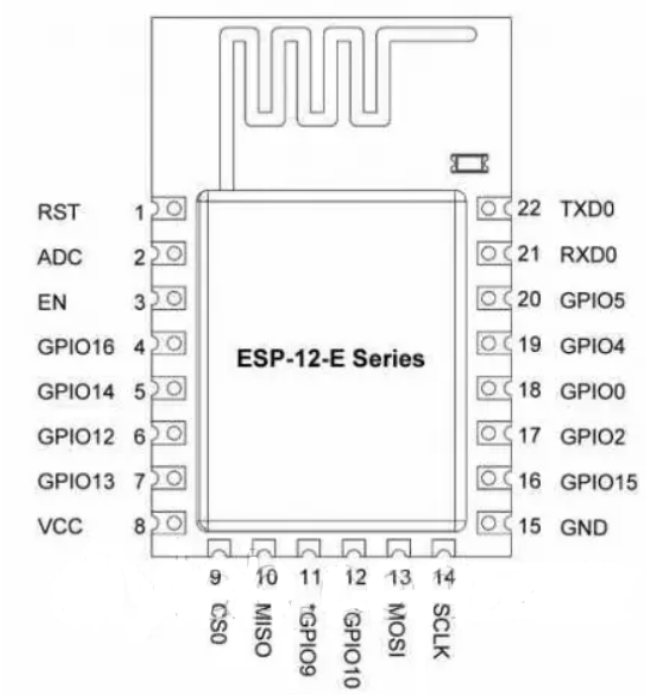


Рисунок 2.11 – Умовно-графічне зображення ESP-12F

2.3.3 Підтримка шини XpressNet

КС може контролювати до трьох пристроїв XpressNet. Кожен пристрій має свою адресу (від 1 до 31). Щоб призначити адресу XpressNet для певного пристрою, дивитися інструкцію з експлуатації для цих пристроїв. Кожен пристрій підключається до КС через роз’єм RJ12.

2.3.4 Підтримка шини S88

Командна станція може керувати модулями зворотного зв'язку S88 (до 128 входів), приєднаними до 6-контактного роз'єму S88 або на базі кабелів RJ45 та CAT5, рисунок 2.12.

Перший модуль S88 в ланцюзі матиме адреси зворотнього зв'язку 65.1-65.8 і 66.1-66.8, другий – 67.1-67.8 і 68.1-68.8 і так далі.

Рисунок 2.12 – Роз'єм шини S88 або на базі кабелів RJ45 та CAT5

2.3.5 Програмування і читання CV

Через обмеження у протоколі XpressNet v.3 від Lenz, лише програмування та читання CV1 до CV256 можна здійснювати в режимах Direct, Paged і Register (сервісний режим). У режимі PoM (програмування на основному треку) можна запрограмувати всі 1024 CV. Щоб прочитати CV в режимі PoM, необхідно активувати RailCom і використовувати зовнішній дисплей (наприклад, Lenz LRC120).

Програмування та читання CV у сервісному режимі можливо лише в PoM режимі. Для того, щоб прочитати CV з декодера, він повинен мати в запасі щонайменше 200 мА енергоспоживання для читання імпульсів.

2.3.6 Сигналізація на платі командної станції

Червоний світлодіод на платі КС використовується для сигналізації однієї з декількох умов, таблиця 2.1.

Таблиця 2.1 – Сигналізація на платі командної станції

Стан світлодіода	Проблема	Вирішення
Світлодіод не горить	Блок живлення не підключений до КС	Переконайтеся, що блок живлення підключений до КС
Світлодіод горить постійно	Нормальний режим роботи командної станції	Все гаразд
Світлодіод горить постійно, але локомотиви не рухаються	З'єднання від КС до трека відсутнє	Перевірте правильність підключення
Світлодіод швидко блимає	Коротке замикання або перевантаження макета	Перевірити макет і усунути КЗ
		Натиснути кнопку аварійної зупинки STOP і перезапустити систему Зняти локомотиви з рейок
Світлодіод блимає повільно	Система знаходиться в сервісному режимі	Закінчене програмування декодера. При виході із сервісного режиму відновиться нормальна робота
Подвійні спалахи світлодіода	Система знаходиться в режимі конфігурації (після встановлення CV7 зі значенням 50 в режимі PoM)	Запрограмуйте CV7 в режимі PoM з відповідним значенням (дивитися таблиці 2,2). Через 15 секунд система повертається до нормальної роботи

2.4 Програмне забезпечення для мікроконтролерів командної станції

2.4.1 Середовище програмування

Для прошивки мікроконтролерів використовувався дуже ефективний безкоштовний програмний пакет для прошивки PIC-мікроконтролерів різних серій WinPic800 [28], рисунок 2.13.

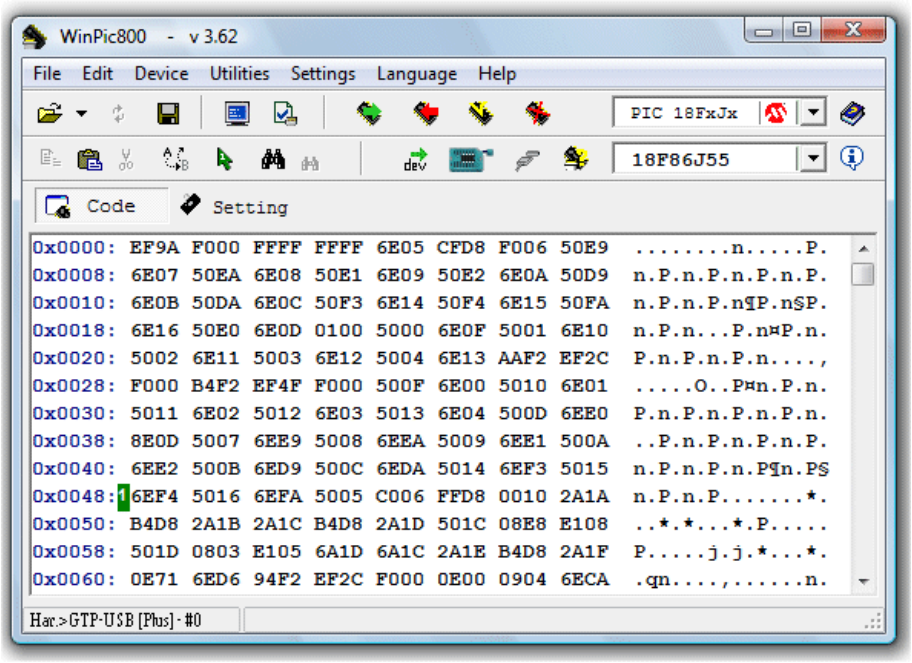


Рисунок 2.13 – Головне вікно WinPic800

Програма WinPic800 працездатна в операційних середовищах Microsoft Windows 98 / NT / 2000 / Me / XP / Vista / 7 / 10.

2.4.2 Програмування мікроконтролерів PIC16F648A

Кроки встановлення та налаштування програми наступні:

- розпаковуємо архів і запускаємо файл WinPic800.exe;
- налаштовуємо програму під адаптер: Установки> Устаткування;
- для обох наведених вище схем вибираємо JDM Programmer.

Треба зауважити, що JDM-програматор — це простий і дешевий програматор для мікроконтролерів, найчастіше PIC (від Microchip), який отримав назву на честь свого автора — Johan de Meersman. Його головна особливість — живлення напряму від COM-порту (RS-232) комп'ютера без додаткового джерела живлення. Також – це простота конструкції, мінімальна кількість компонентів (декілька резисторів, діодів, транзисторів) і, крім того, він не потребує мікроконтролера в самій схемі програматора.

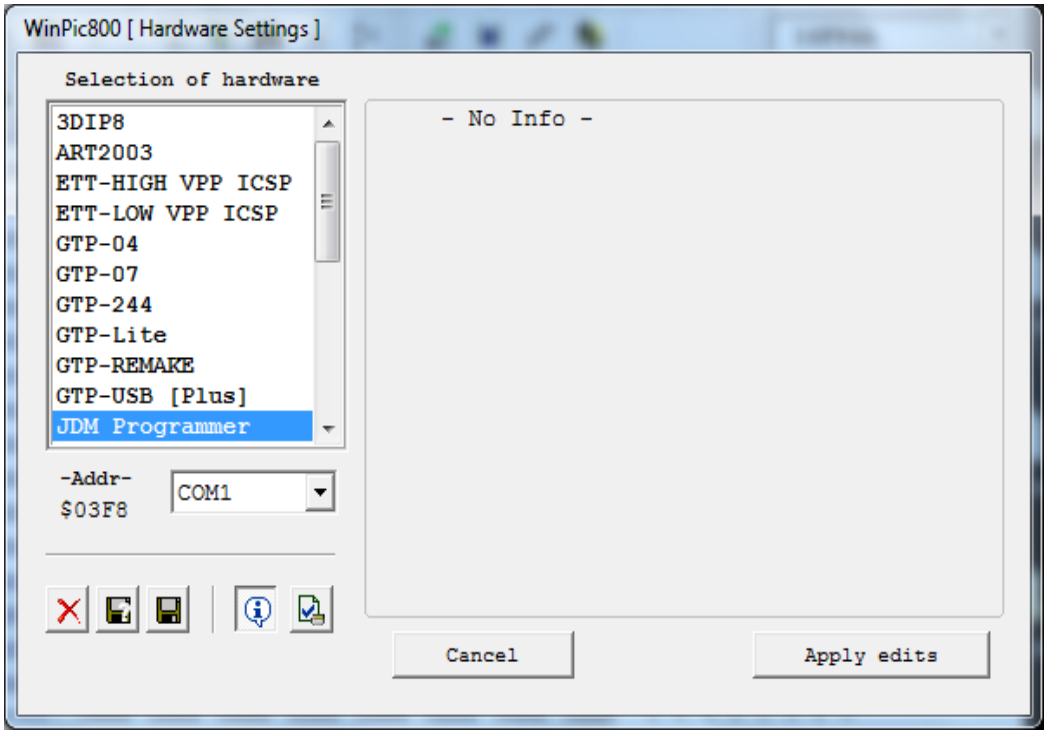


Рисунок 2.14 – Налаштування програматора

- кількома на закладку "Установки" в цьому ж вікні;
- прибираємо галочку "Blockage configuration";
- для вибору програм Compic ставимо галочку Inv навпаки DataIn.

Потім необхідно протестувати програматор на коректність роботи і тільки після цього рухатися далі.

В Установки > Програма > Device ставимо галочку "Використовувати визначення пристрою користувачем".

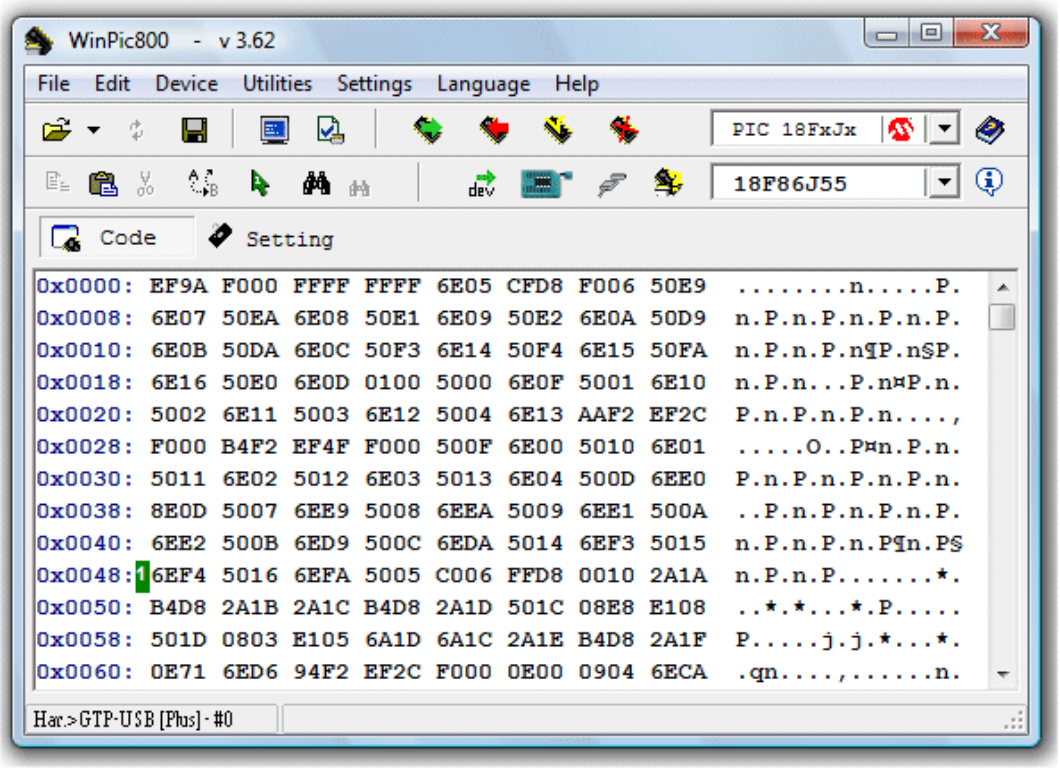


Рисунок 2.15 – Зовнішній вигляд файла прошивки

Програмування:

- відкриваємо файл прошивки *.hex;
- замість символів 3FFF з'являється код програми. Вибираємо сімейство (наприклад, PIC16F);
- вибираємо потрібний пристрій (наприклад, PIC18FXJX);
- встановлюємо біти конфігурації мікроконтролера;
- відкриваємо закладку "Config";
- встановлюємо відповідні галочки. (при використанні кварцу вибираємо XT до 4МГц або HS понад 4МГц);
- натискаємо Ctrl + P і спостерігаємо процес програмування (не забудьте включити живлення для контролера на программаторе, якщо воно передбачено).

3 ПРОГРАМНА РЕАЛІЗАЦІЯ КОМПЛЕКСУ

3.1 Постановка задачі та опис вхідної інфомаціїї

3.1.1 Постановка задачі на розробку застосунку

Метою проєкту є не лише розробити зручний спосіб дистанційного керування, а й адаптувати його на більшість доступних платформ: Windows, Linux, Mac OS, iOS, Android. Для цього підійде не кожна сучасна технологія розробки додатків, тому в якості мультиплаформного програмного інструментажу обрано фреймворк із відкритим кодом Flutter, що можна вважати однією з вимог під час розробки [29-33].

Додаток повинен відповідати таким вимогам:

– приємний та зручний інтерфейс в Material Design стилі;

– мультиплатформенність;

– підтримка локалізації;

– анімації, візуальні UX покращення;

– реалізація протоколу XpressNet;

– можливості користувацької візуальної кастомізації;

– зберігання введених користувачем даних.

Додаток повинен містити такі екрани:

– початковий екран під час ініціалізації і запуску – “splash-screen”

– екран роботи з локомотивами;

– екран роботи з аксесуарами;

– екран програмування командної станції для просунутих користувачів;

– діалогові вікна при виникненні помилки під час підключення та роботи з командною станцією;

– діалогове вікно вибору значка (іконки) із вказаної колекції для швидшого орієнтування в інтерфейсі;

– діалогове вікно керування групами аксесуарів;

– діалогове вікно керування маршрутами аксесуарів;

– діалогове вікно для введення. текстових даних (наприклад, назва маршруту).

Додаток повинен надавати такий логічний функціонал:

– бездротове з'єднання з командною станцією засобами Wi-Fi або Bluetooth;

– керування станом живлення командної станції;

– вибір і керування доступними на залізничному макеті локомотивами (можлива кількість: від 1 до 256);

– регулювання напрямком руху, швидкістю і її градацією;

– доступ до спеціальних функцій локомотива;

– керування аксесуарами на залізничному макеті;

– створення та програвання маршрутів, складених із аксесуарів-стрілок користувачем;

– ручне програмування змінних командної станції для просунутих користувачів.

Додатково, необхідно надати можливість візуальної кастомізації:

– присвоювання локомотивним функціям значків (іконок) для швидшого орієнтування в інтерфейсі;

– групування аксесуарів для швидшого доступу без потреби ручного вводу їх адреси;

– найменування та контроль над маршрутами.

Під час розробки необхідно врахувати такі вимоги:

– використання мультиплатформених плагінів (для коректного виконання команд на усіх платформах);

– модульний, структурований, «чистий» код, що поліпшить процес майбутніх доробок та масштабування;

– використання бібліотек локалізації та інтернаціоналізації (*l10n* та *i18n*);

– використання SharedPreferences для зберігання постійних даних.

3.1.2 Опис вхідної інформації

Більшість вхідних даних додатку отримується способом інтерактивним – потреба обробляти велику кількість даних, вручну введену користувачем, відсутня.

До основних елементів взаємодії відносяться такі компоненти:

– перемикач стану живлення командної станції;
– селектор адреси поточного локомотива;
– регулятор напряму руку локомотива;
– регулятор швидкості руху локомотива;
– селектор градації швидкості локомотива;
– клавіатура локомотивних функцій;
– поле вводу адреси аксесуара;
– перемикач стану поточного аксесуара.

Для візуальної складової компонентами вхідних даних стане:

– селектор значка для локомотивної функції;
– назва та значок групи швидкого доступу до аксесуарів;
– назва та склад маршруту (список зв'язанних стрілок).

Однак, окрім цього, отримана інтерактивнм методом інформація додатково обробляється програмою, та передається командній станції в форматованому вигляді за обраним протоколом спілкування з DCC.

Для комунікацією з командною станцією під час роботи програми використовуються такі змінні:

– значення стану живлення командної станції;
– адреса поточного локомотива на екрані керування локомотивом;
– значення напряму руху вибраного локомотива;
– поточне значення швидкості вибраного локомотива;
– матриця ввімкнених та вимкнених локомотивних функцій;
– адреса поточного аксесура на екрані роботи з аксесуарами.

Таблиця 3.1 – Атрибути об'єкта «Локомотив»

Атрибут об'єкту	Тип даних	Короткий опис
Адреса	Ціле число	Номер локомотива на залізничному

	(0–1023)	макеті
Швидкість	Ціле число (0–128)	Значення поточної швидкості
Градація швидкості	Ціле число (допустимі значення: 14, 28, 128)	Максимально можливе значення швидкості (чим більше, тим плавніше градація
Напрям руху	Ціле число	Використовується для позначення реверсивного руху
Матриця функцій	Список функцій (максимальна кількість: 28)	Список усіх доступних функцій локомотива з їх власним станом

Таблиця 3.2– Атрибути об’єкта «Локомотивна функція»

Атрибут об’єкту	Тип даних	Короткий опис
Адреса	Ціле число (0–1023)	Номер локомотивної функції на залізничному макеті
Стан	Булевий	Ввімкнений або вимкнений
Значок *(візуал)*	*Спеціальний*	Значок функції обраний користувачем для кращого опізнавання

Таблиця 3.3 – Атрибути об’єкта «Аксесуар»

Атрибут об’єкту	Тип даних	Короткий опис
Адреса	Ціле число	Номер аксесуара

Стан	Булевий	Ввімкнений або вимкнений

Таблиця 3.4 – Атрибути об’єкта «Група аксесуарів»

Атрибут об’єкту	Тип даних	Короткий опис
Назва	Стрічка	Назва групи
Значок	*Спеціальний*	Значок групи
Аксесуари	Список із аксесуарів	Відсортований за зростанням список аксесуарів, доданих до групи користувачем, з метою швидкого доступу до частовикористаних аксесуарів залізничного макету

Таблиця 3.5 – Атрибути об’єкта «Маршрути»

Атрибут об’єкту	Тип даних	Короткий опис
Назва	Стрічка	Назва маршруту, введена користувачем
Стрілки	Список із аксесуарів	Впорядкований список аксесуарів, відведених до групи стрілок, кожен містить предзаповнений атрибут *Стан*, що відповідає за необхідний стан стрілки в момент програвання маршруту

Таким чином, ключовими об’єктами при реалізації наведеної частини протоколу XpressNet є такі, що наведені на таблицях 3.1, 3.2, 3.3, 3.4 та 3.5.

Результуючою інформацією у своїй більшості буде інформація, обчислена за обрним протоколом спілкування з DCC. Усі вхідні користувацькі дані, отримані через інтерфейс програми, необхідно нормалізувати, адаптувати під

різні протоколи та, нарешті, відправити на опрацювання командній станції. Це досягається за допомогою простої бітової і байтової арифметики, виведених формул та логічних тверджень.

Цей прихований для користувача (за виключенням роботи в режимі відладки) шар логіки можна вважати результатом її роботи, адже в цьому також полягає ціль створюваного додатку. Результатом, в такому випадку, буде позитивна відповідь від командної станції про виконану дію, який обробляється і перевіряється у фоновому режимі.

Подібному обчисленню підлягають такі атрибути:

– байт швидкості локомотива;
– байт градації швидкості локомотива;
– байт напрямку руху локомотива при обчисленнях швидкості;
– група перемикаючої локомотивної функції;
– бітмаска групи локомотивних функцій;
– байт перемикаючого аксесуара;
– конфігурація командної станції (CV – Configuration Variable);
– визначення групи CV, обчисленя байту значення;
– адаптація під різні версії протоколу при обчисленнях.

З точки зору інтерфейсу варто вказати візуально-необхідні для сприйняття елементи додатку. Наприклад, спідометр для відображення швидкості обраного локомотива – один із центральних віджетів при роботі з програмою, оскільки 90% проведеного із залізничим макетом часу приділяється керуванню локомотивом. Результатами буде виведені на екран, попередньо супроводжені анімаціями: швидкість локомотива, градація швидкості, клавіатура локомотивних функцій, групи аксесуарів, список маршрутів.

3.2 Проєктування архітектури, структури та інтерфейсу програми

3.2.1 Етапи проєктування програми

Враховуючи вимоги, описані в підрозділі 3.1, розробка додатку послідовно поділяється на такі етапи:

– розробка прошивки для Wi-Fi модуля зв'язку з командною станцією;
– реалізація протоколу на мові програмування Dart;
– проєктування, макетуваня, розробка графічного інтерфейсу програми;
– поєднання користувацького інтерфейсу з логікою протоколу;
– тестування додатку та його функціоналу.

Для першого етапу буде використано низькорівневу мову програмування, призначену для програмування мікроконтролерів. Необхідно реалізувати простий TCP проксі-сервер, який приймає вхідні пакети, та транслює їх командній станції.

Другий етап полягає у повній або частковій реалізації протоколу в рамках обраної мови програмування. Вимогами до написаного API буде простота інтеграції та подальшого використання.

На третьому етапі необхідно визначитись з багатьма дизайнерськими моментами створення юзер-інтерфейсу з нуля. Наприклад: палітра кольорів, розміщення віджетів, анімації, навігація по екранам, дотримання одного стилю тощо. Усі можливості взаємодії з додатоком детально зображені на діаграмі use-case (Додаток Б), які треба врахувати при макетуванні інтерфейсу додатку.

Четвертий етап потребує найбільше часу та уваги. Під час його виконання потрібно з'єднати вхідну інформацію користувача з результуючою інформацією командної станції. Цей процес супроводжується нюансами візуальними (оновлення віджетів, анімації) та логічними (асинхронний код, розділення швидкості на реальну та візуальну, проміжні дії із структурами даних). Часто при розробці на фреймворку Flutter виникає необхідність в реалізації власних_віджетів та компонентів інтерфейсу. Цей процес також входить до четвертого етапу злиття візуалу із логікою.

Останній етап – це модульне або ручне тестування усіх вузлів системи.

3.2.2 Проєктування архітектури, структур даних, інтерфейсу

Фреймворк Flutter має багато різних підходів до проєктування архітектури додатку. Одними з найпопулярніших є: Native State, Provider, BloC, Redux. Я обрав перший – простий і частовикористовуваний спосіб керування станом програми.

Все, що використовується у Flutter, складається з віджетів. Вони можуть бути видимими, невидимими, містити дочірні віджети та взаємодіяти між собою. Кожен їх може бути як віджетом без стану (Stateless Widget), і віджетом, що має стан (Stateful Widget). Основна відмінність – можливість повторно “рендерити” віджети під час виконання програми. Stateless Widget буде малюватись лише один раз і є незмінним. Stateful Widget може відмальовуватися багато разів залежно від зміни внутрішнього стану віджету.

Для створення Stateful Widget потрібно створити два класи. Перший клас повинен успадковуватися від Stateful Widget, який у свою чергу успадковується від Widget і є незмінним. Примірник цього класу не перетворюється при кожному малюванні та використовується для зберігання переданих параметрів та ініціалізації стану. Другий – клас стану, який має доступ до Stateful Widget через внутрішню властивість і займається безпосередньо відмальовуванням стану, реагуючи на його зміну.

Перевагами такого підходу є простота та швидкість впровадження у додаток з невеликою кількістю екранів, що виражається у відсутності будь-яких додаткових бібліотек.

Проте, під час розробки я стикнувся і з певними недоліками:

– інтерфейс та бізнес-логіка ніяк не розділені;
– складність у модульному тестуванні;
– складність у масштабуванні та підтримці.

Для об’єктно-орієнтованого програмування у дипломному проєкті використовуються принципи DRY та SOLID. Це дозволяє тримати код в чистоті, простіше орієнтуватись в ньому та масштабуватись з меншим зусиллям.

Принцип DRY (Don’t Repeat Yourself – «не повторюй себе») полягає в уникненні дублювання інформації, тобто програмного коду. Цей принцип було

втілено у дипломному проєкті завдяки розбивки великих елементів екрану на менші.

Наприклад, екран керування локомотивом набуває такої структури:

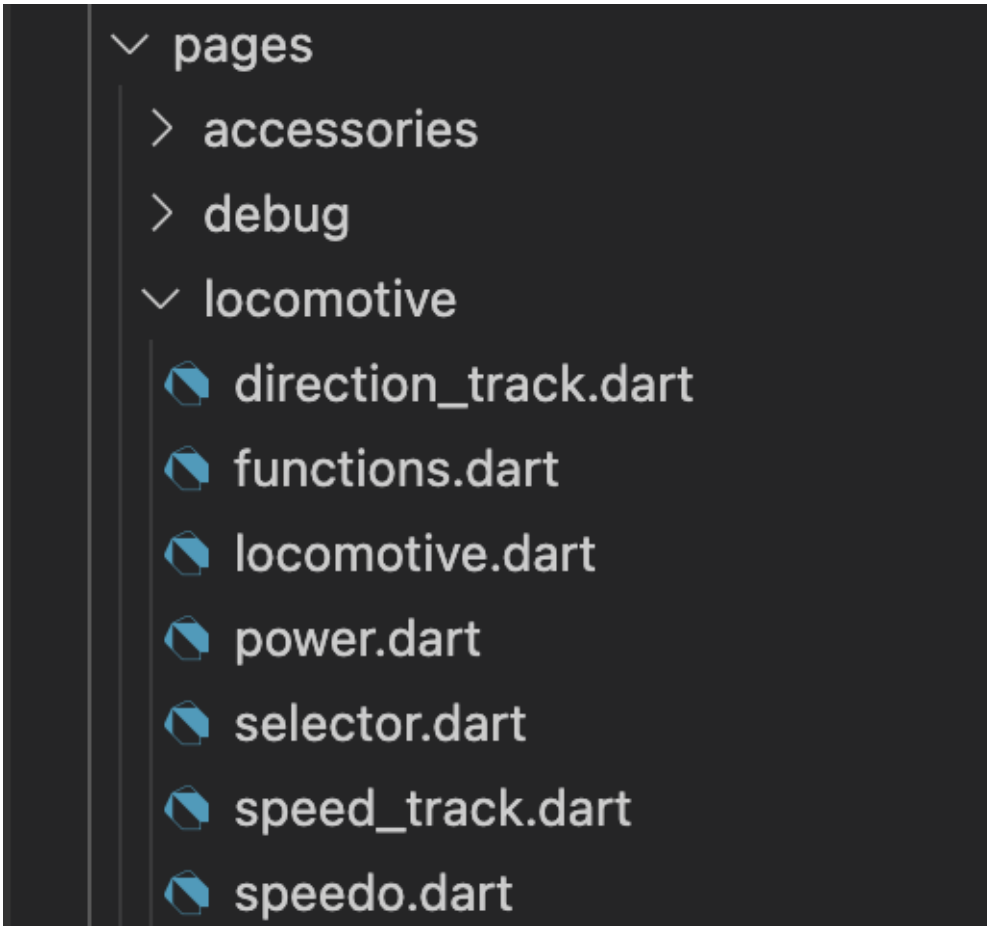


Рисунок 3.1 – Структура пакета pages проєкту

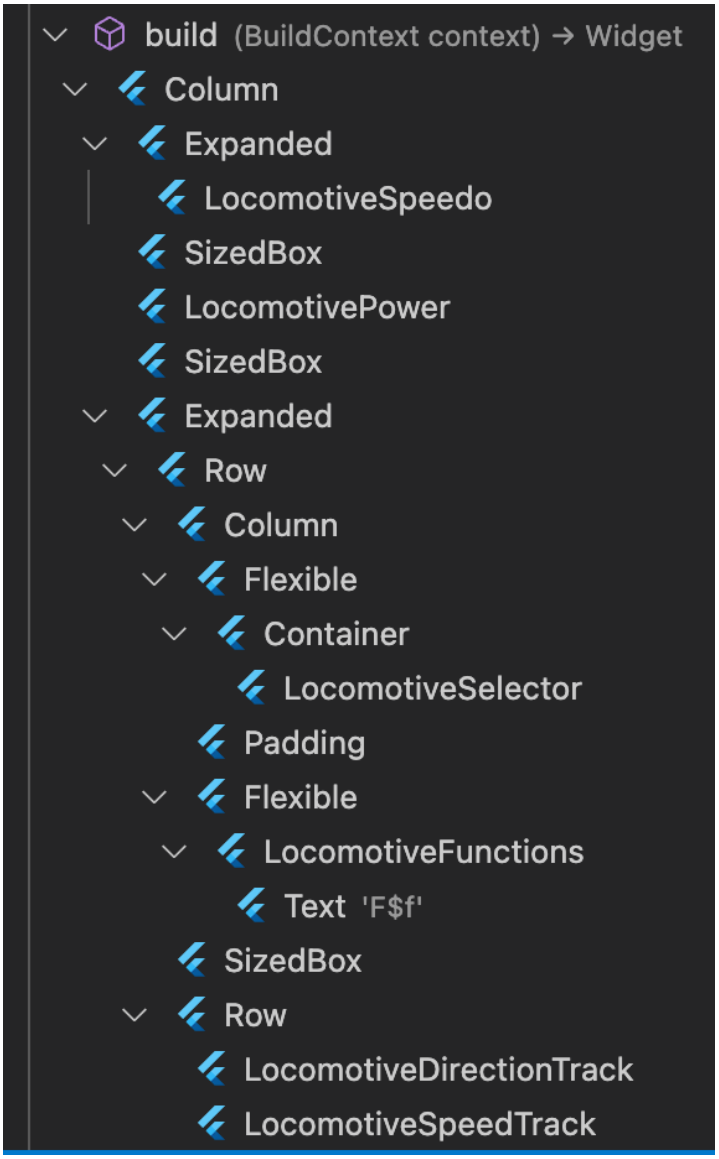


Рисунок 3.2 – Структура Locomotive сторінки проєкту

Інше поняття, що є абревіатурою п’яти базових принципів об’єктно-орієнтованого програмування, має назву SOLID. Аналогічно до принципу

DRY, більшість складових SOLID реалізується у побудові графічного інтерфейсу. До SOLID входять такі принципи: принцип єдиного обов’язку, принцип відкритості/закритості, підстановки Лісков, розділення інтерфейсу, інверсії залежностей.

Компоненти любого з екранів легко можуть бути використані іншими екранами або віджетами при необхідності, достатньо лише імпорувати потрібний віджет та включити його в функцію побудови.

В цілому, структура проєкту Flutter виглядає таким чином:

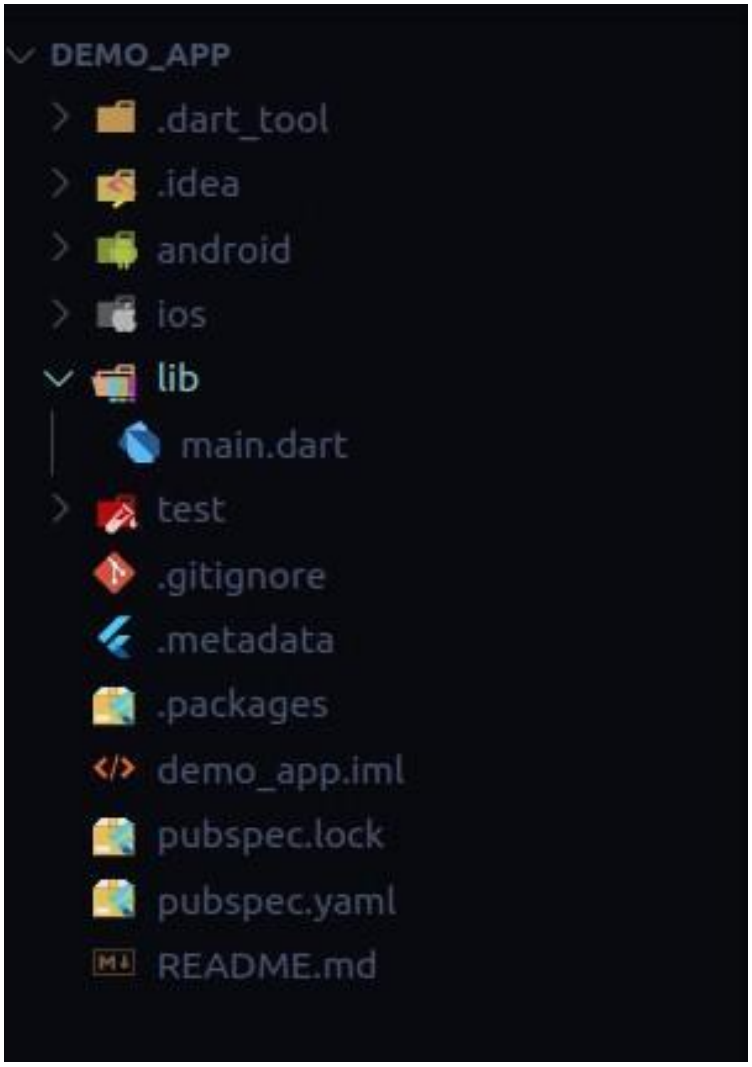


Рисунок 3.3 – Приклад структури проєкту розробленого на Flutter

– `lib` –весь код програми лежить в цій папці;

– `pubspec.yml` – залежності програми необхідні для її запуску;

– `test` – директорія, яка містить тести додатку;

– `ios, android` – папки з налаштуваннями для кожної з платформ, там вказується які права потрібні для запуску програми (доступ до локації, Bluetooth), і все що специфічно для платформи.

Розробнику надається повна свобода у структуризації проєкта – загальноприйнятих поширених стандартів майже не існує, що дозволяє невеликим командам визначитись із зручною для них архітектурою та без жодних недоліків працювати в її рамках.

Структура проєкту має такий вигляд як на рисунку 3.4.

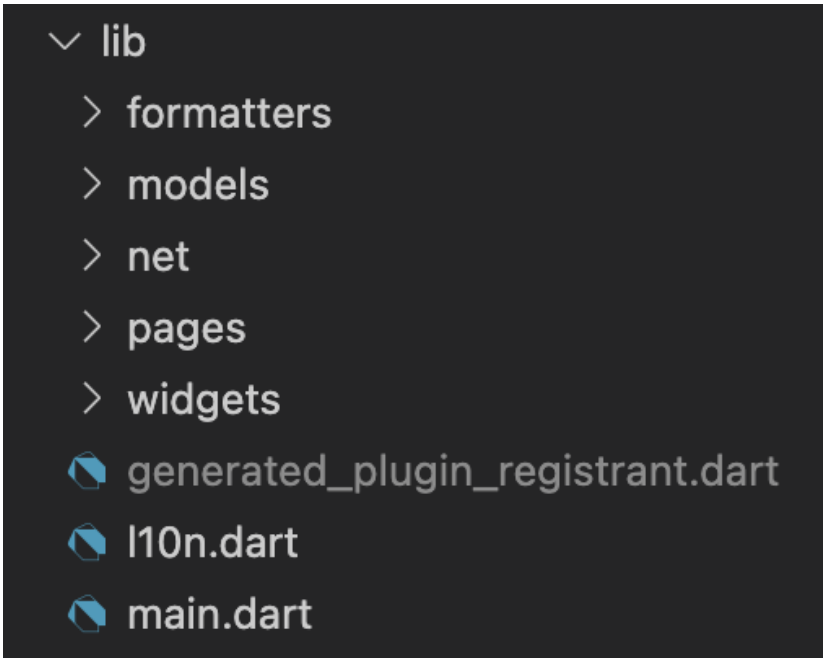


Рисунок 3.4 – Структура проєкту

Згруповано та розміщено файли проєкту, таким чином:

– `formatters` – містить класи, пов’язані із форматуванням даних;
– `models` – програмні високорівневі класи або обгортки над іншими;
– `net` – реалізація протоколу XpressNet;
– `pages` – містить екрани графічного інтерфейсу;
– `widgets` – реалізація власних віджетів.

Код на Flutter має схильність до великого рівня вкладенності через особливості ієрархічного компонування, тому важливим фактором є своєчасний та постійний рефакторинг, розбивка великих віджетів на менші частини.

В рефакторингу часто допомогають створені іншими розробниками плагіни для популярних редакторів коду та IDE. В якості редактора було обрано VS Code, разом із такими розширеннями: автоматичне форматування коду для Dart & Flutter, Flutter Widget Wrap для зручної роботи з деревом віджетів, Flutter Snippets для попередньо створених сніпетів (частин коду) частовикористаних патернів розробки на Flutter.

Інтерфейс додатку розроблявся поетапно, часто з постійними змінами, правками і корекціями, оскільки адаптація під велику кількість платформ з різним розміром екрану потребує максимальної гнучкості.

Такий вигляд має найперший екран програми – екран ініціалізації, рисунок 3.5.

Рисунок 3.5 – Екран ініціалізації додатку (splash screen)

При ввімкнені режиму відладки, логічне призначення екрану – встановлення зв’язку з командною станцією – пропускається. Це дає можливість

спершу розробити інтерфейсу додатку, а потім реалізувати логіку і адаптувати її до віджетів на екрані.

Інакше, відбудеться спроба з'єднання з Wi-Fi модулем командної станції та, у випадку виникнення помилки, для користувача відобразиться подібне діалогове вікно, рисунок 3.6.

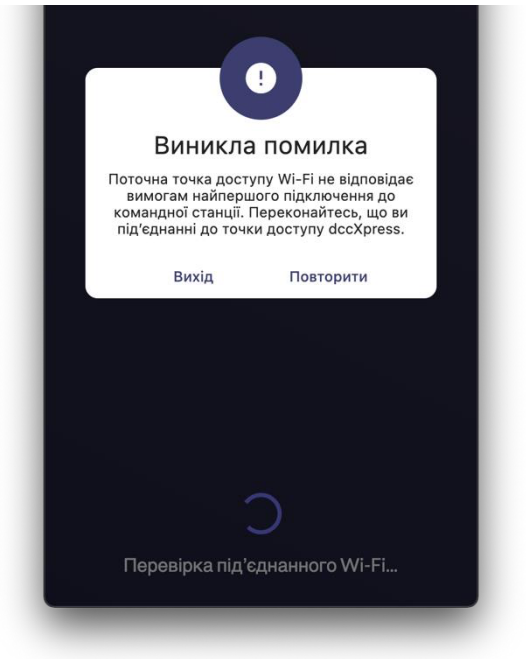


Рисунок 3.6 – Приклад діалогового вікна із повідомленням про помилку

Так виглядає нижнє меню навігації між екранами, рисунок 3.7.

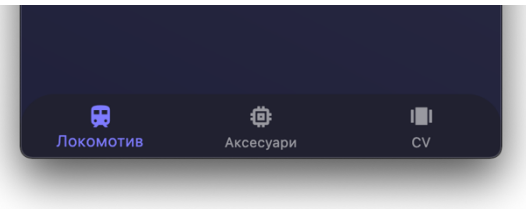


Рисунок 3.7 – Нижнє навігаційне меню додатку

При переході на екран аксесуарів, центральна вкладка "Аксесуари" анімовано перетворюється на кнопку створення маршруту:

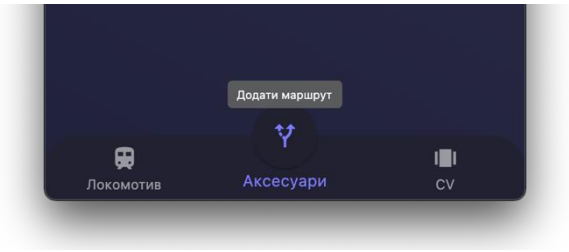


Рисунок 3.8 – Кнопка додавання маршруту як ключовий елемент екрану

До екрану локомотива входять такі компоненти:

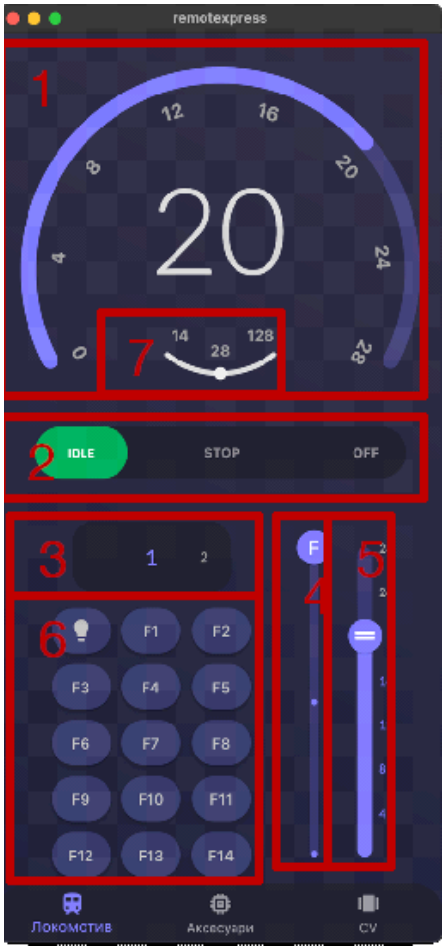


Рисунок 3.9– Елементи екрану керування локомотивами

1. Спідометр локомотива, відображає поточну швидкість;

2. Регулятор подачі живлення (IDLE – робочий режим, STOP – режим екстренного гальмування, OFF – повне відключення живлення);

3. Селектор поточного локомотива;

4. Важіль напрямку руху обраного локомотива (F – вперед, N – нейтраль, R – реверс);

5. Важіль регулювання швидкості руху обраного локомотива;

6. Матриця локомотивних функцій;

7. Селектор градації швидкості обраного локомотива.

До екрану аксесуарів входять такі компоненти:

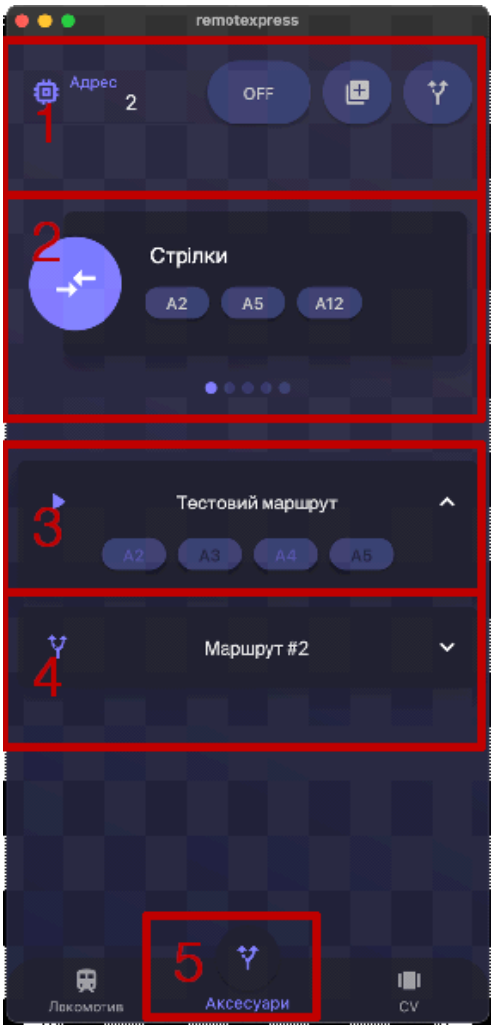


Рисунок 3.10 – Елементи керування екрану аксесуарами

1. Панель керування введеним аксесуаром – перемикання стану, додаваняя і видалення з групи, додавання і видаленн із маршрута;

2. “Свайпер” груп аксесуарів – динамічний віджет перелистування доступних груп;

3. Записаний маршрут у розгорнутому стані – дає можливість переглянути додані стрілка та програти його;

4. Маршут у згорнутом стані;

5. Кнопка додавання нового маршруту.

Вищенаведені рисунки відповідають за приблизні макети діалогових вікон у додатку. Вони виконані у єдиному стилі та можуть містити в собі любий інтерактивний елемент.

3.3 Розробка та опис програмної реалізації та тестування задачі

Відповідно до пункту 2.4.1 в якості Wi-Fi модуля обрано ESP-12F, який базується на мікроконтролері ESP8266. Прошивка до якого розробляється на мові програмування Arduino [34].

Для цієї частини проєкту створено спеціальну директорію *arduino*, в якій, поки що, знаходить один файл *remotexpress.ino* з прошивкою модуля. В результаті роботи над кодом прошивки створено простий проксуючий TCP-сервер, який приймає пакети від доступних клієнтів (максимум 4) та передає їх в COM-порт, що працює на стандартній швидкості 9600 біт/с. Аналогічно, прочитані через Serial порт дані, передаються усім клієнтам сервера. Додатково реалізована можливість конфігурації роботи Wi-Fi модуля таким чином, щоб він підєднувався до Wi-Fi роутера в локальній мережі. Це зроблено для максимальної гнучкості при майбнутному допрацюванню додатку.

Тепер, маючи робочий спосіб комунікації із командною станцією, можна перейти до розробки протоколу XpressNet на мові програмування Dart. Він буде представлений у вигляді самостійної бібліотеки з власним, спрощеним API. При необхідності, її можна буде включити в інші проєкти, іншими розробниками.

Приклад включення:

```
import 'package:remotexpress/net.dart'
```

Детальні зв'язки між класами зображені на діаграмі класів модуля реалізації протоколу.

Реалізація протоколу розташовується в директорії *net* та містить чотири основні класи для роботи із залізничним макетом на даний момент: *Station, Command, Locomotive, Accessory*. Уся взаємодія відбується через перший клас в списку. За допомогою цього API також є можливість викликати нереалізовані бібліотекою команди – їх можна сформувати та відправити самостійно.

Проєктування та макетування інтерфейсу описано у розділі 2.1, тому на данний момент залишається розробити його з використанням фреймворку Flutter. Підготовлена у цьому ж розділі структура проєкту дозволяє одразу перейти до його реалізації. У створеному пакеті *pages* я описав каскад наявних екранів, разом із початковим екраном ініціалізації.

Для виконання вимоги мультиплатформеного гнучкого інтерфейсу було використано спеціальні віджети *Expanded, Flexible*, разом із контейнерами *Column* і *Row*. Це дозволило описати макет інтерфейсу лише раз, а на екранах різної висоти та ширини, розміри елементів автоматично корегувались.

Кожен самостійний компонент любого із екранів знаходиться в окремому файлі – це дозволяє зберегти модульність та не перевантажувати код програми. Зрозуміла структура директорії таким чином дозволяє швидко здійснювати навігацію по файлам проєкту.

Під час розробки екранів графічного інтерфейсу неодноразово доводилось шукати альтернативні віджети та плагіни, поширені іншими розробниками. У виключних випадках доводилось реалізовувати такі самостійно.

До списку використаних плагінів входять:

– *intl* – локалізація проєкту
– *shared_preferences* – зберігання постійних даних;
– *network_info_plus* – робота з мережею;
– *json_annotation* – серіалізація та десеріалізація даних;
– *google_fonts* – шрифти;
– *cupertino_icons* – іконки;

– *syncfusion_flutter_gauges* – віджет спідометра;

– *syncfusion_flutter_sliders* – віджет слайдера;

– *countup* – анімація лічильника;

– *numberpicker* – віджет числового селектора;

– *card_swiper* – віджет групи карток (свайпер);

– *expandable* – віджет розширюваного елемента списку;

– *icon_picker* – віджет вибору іконки.

До списку самостійно створених віджетів входять:

– *animated_toggle* – анімований віджет перемикаючих кнопок (використовується на екрані керування локомотивами для перемикання стану живлення командної станції);

– *custom_dialog* – власний віджет діалогового вікна;

– *toggle_button* – власний віджет перемикаючої кнопки у двох станах (ON/OFF).

Кожен із створених віджетів наслідує stateful реалізацію віджетів, оскільки має властивість змінювати себе в результаті взаємодії. Вони часто використовуються основними компонентами інтерфейсу, враховуючи аспекти додатку. Але, незважаючи на це, віджети можна використати і поза поточним проєктом, оскільки створені максимально незалежними та самостійними.

Під час розробки часто використовувались віджети, які відповідають за анімацію, це дозволило зробити додаток більш живим та інтерактивним. Наприклад: *AnimatedAlign, AnimatedSwitcher, CurvedAnimation, AnimatedBuilder, AnimationController*.

Однією з вимог додатку було зберігання введеної користувачем інформації. Оскільки інтеграція повноцінної бази даних буде надмірним для програми без комплексних структур даних, в якості формату серіалізації було обрано JSON. Дані записувались за допомогою плагіну *shared_preferences*, в якому вже існує підтримка більшості платформ. Проте, для того щоб успішно серіалізувати дані, вони повинні реалізувати відповідний програмний інтерфейс. Це привело до створення спеціальної директорії *models*, яка містить власні класи та обгортки над

іншими. За допомогою ще одного плагіну, я зміг автоматизувати опис структури даних у форматі JSON та згенерувати їх заранння. Такий спосіб розробки називається кодогенерацією.

Для зручності використання бібліотеки локалізації створено тип-обгортку розташовану в файлі *l10n.dart*, яка теж використовує генерацію коду.

Останній етап полягає у повному розгортанні проєкту і його тестуванні. Додаток не має практичного сенсу без ключового компоненту системи – командної станції та, власне, залізничного макету. Проте, для тестування я використовую мініатюрну версію такого – платформа з однією рейкою, локомотивом із світловим та звуковим декодером, набором аксесуарів. До рейки підєднується командна станція з контролером XpressNet версії 3.6, до якої, в свою чергу, прошитий Wi-Fi модуль. Опціонально, до командної станції може бути підключений модуль Bluetooth, що теж надає додаткові можливості для користувачів. Останнім компонентом системи є створюваний додаток *remoteXpress*.

Для тестування в основному використовувся ручний спосіб тестування – повне розгортання системи і перевірка кожної частини функціоналу самостійно. Звісно, це викликає свої труднощі після розширення або зміни функціоналу додатку, але можливість автоматизованого тестування потребує реалізацію емулятора командної станції таким чином, щоб вона повторювала логіку фізичної. Це не є задачею простою, не гарантує правильність реалізації, та потребує додатковий об’єм зусиль на створення дипломного проєкту.

3.4 Інструкція з інсталяції та експлуатації розробленого проєкту

3.4.1 Інструкція з інсталяції

Існує декілька варіантів запуску проєкту:

– запустити версію для відладки, з можливістю “гарячого перезавантаження”;

– збудувати реліз-версію програми та встановити на девайс вручну.

Для запуску додатку необхідно:

– операційну систему сімейства Windows, Linux, MacOS, iOS, Android;

– наявний працюючий Wi-Fi модуль.

Для запуску програми з нуля необхідно:

1. Виконати команду `flutter pub get` для завантаження необхідних залежностей програми;

2. Для запуску додатка в режимі відладки виконується команда:

```
flutter run --no-sound-null-safety
```

3. Для запуску додатка в режимі релізу виконується команда:

```
flutter run --release --no-sound-null-safety
```

4. Обрати потрібний девайс для розгортання програми:

```
Multiple devices found:
macOS (desktop) • macos • darwin-arm64
Chrome (web)  • chrome • web-javascript
```

Під час роботи з кодом програми можуть знадобитись такі команди:

1. `flutter gen-l10n` для генерації файлів локалізації (для цього використовується бібліотеки *flutter_localizations* і *intl*);

2. `flutter pub run build_runner build` для генерації файлів серіалізації (для цього використовується бібліотека *json_annotation*).

Необхідні складові для розгортання зображені на діаграмі розгортання (Додаток 4).

3.4.2 Інструкція з експлуатації

Так як дипломний проєкт полягає у реалізації програмної частини в інструкції упущуенно частину для розгортання апаратної складової. Будемо вважати, що командна станція уже з'єднанна із залізничним макетом та має справний прошитий Wi-Fi модуль.

На екрані керування локомотивами, рисунок 3.11, користувач проводить 90% часу роботи з додатком. Тут зосереджено більшість часто-використовуваних функцій при взаємодії із залізничним макетом:

– Спідометр локомотива відображає поточну швидкість та її градацію, програється з анімацією, дозволяє керувати градацією швидкості, розповсюджується лише на обраний локомотив;

– Регулятор подачі живлення. IDLE означає робочий режим, при обранні режиму STOP відбувається екстренне гальмування усіх локомотивів на макеті, при обранні режиму OFF відбувається повне фізичне відключення живлення командної станції;

– Селектор поточного локомотива дозволяє змінити адресу локомотива, з яким працює користувач;

– Важіль напрямку руху обраного локомотива дозволяє вибрати напрям руху локомотива: F – вперед, N – нейтраль, R – реверс. Блокується при обраному режимі командної станції OFF;

– Важіль регулювання швидкості руху обраного локомотива змінює його поточну швидкість. Блокується при обраному режимі командну станції STOP;

– Матриця локомотивних функцій довзоляє контролювати їх стан (ввімкнений/вимкнений). Користувач може задати для кожної функції значок, зажавши її на декілька секунд замість натискання – це довзолить швидше орієнтуватись між адресами функцій;

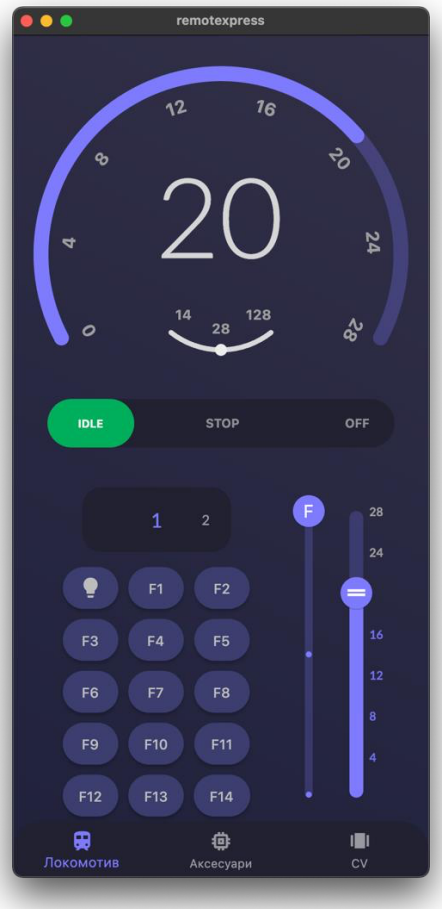


Рисунок 3.11 – Інтерфейс екрану керування локомотивами

На екрані керування аксесуарами зосереджені елементи роботи із аксесуарами станції – це всі додаткові моделі на макеті, які можна привести в рух: стрілки, освітлення, світлофори, шлагбауми, різноманітні декорації. Аксесуар буває, або ввімкненим, або вимкненим. Його можна додати до групи для швидкого доступу без необхідності вводу адреси. Функціонал маршрутів дозволяє згрупувати стрілки на макеті в маршрути та згодом перемикати їх одна за одною. При додаванні стрілки до маршруту, її стан запам’ятовується.

ВИСНОВКИ

В результаті роботи над дипломним проєктом розроблений Апаратно-програмний комплекс дистанційного керування залізничним макетом на мікроконтролерах з мультиплатформною програмною підтримкою.

Проєктування комплексу здійснювалось шляхом вибору необхідних апаратних компонентів з Інтернет-джерел, а також розробку мультиплатформного застосунку для реалізації дистанційного керування залізничним макетом.

Комплекс призначений для дистанційного керування залізничними макетами масштабів H0 (1:87), TT (1:120), N (1:160). Комплекс включає: цифрову командну станцію стандарту DCC з інтерфейсами WiFi та Bluetooth, а також мультиплатформний застосунок для операційних систем Windows, Linux, Mac OS, iOS, Android для дистанційного керування.

Основні характеристики цифрової командної станції:

- підтримка стандартів та протоколів– DCC, Xpressnet, X-bus, S88;
- підтримка інтерфейсів – Bluetooth, WiFi; Lokmaus, Multimaus;
- максимальний вихідний струм бустера – 3 А, захист від коротких замикань бустера;
- одночасна підтримка 16 локомотивів в адресах від 1 до 9999;
- підтримка 1024 аксесуари (стрілки та сигнали); можливість підключення пристроїв XpressNet типу Lokmaus, Multimaus або аналогічних;
- програмування та читання DCC-декодерів у режимах Direct, Paged, Register і PoM;
- підтримка 14, 28 та 128 ступенів швидкості локомотивів;
- підтримка функцій керування світлом (FL) і F1 до F12 для кожного локомотива;
- підтримка 128 входів зворотного зв'язку з модулями S88.

Основні функціональні можливості застосунку:

- реалізація функцій командної станції на програмному рівні;

- мультиплатформність;
- реалізація протоколу XpressNet;
- дружній та зручний інтерфейс в Material Design стилі;
- підтримка локалізації;
- можливості користувацької візуальної кастомізації.

Зв’язок з макетом здійснюється через інтерфейси Bluetooth або Wi-Fi.

Використання інтерфейсу Flutter та мови програмування Dart дало змогу ефективно розробити одну версію коду, яка працює на всіх платформах. Окрім того, було реалізовано спеціальну прошивку для Wi-Fi модуля в якості проміжного мосту. Застосунок надає зручний спосіб дистанційно керувати залізничним макетом та його елементами: змінювати швидкість та налашутвання усіх локомотивів на макеті, керувати їх функціями; перемикати аксесуари та автоматизувати побудову маршрутів.

Отже, до переваг розробленого проєкту можна віднести:

– гнучкість у розробці і універсальність написаного коду;
– унікальність проєкту в обраній ніші;
– зручний та приємний інтерфейс;
– можливості користувацької кастомізації;
– автоматизація ключових моментів при роботі з макетом.

Тим не менш, проєкт залишає багато можливостей для покращення та розширення функціоналу:

– адаптація інтерфейсу для широкоформатних дисплеїв;
– голосовий ввід команд;
– відстуність зворотного зв’яку з командною станцією на даному етапі;
– відсутність синхронізації користувацьких даних між клієнтами;
– розширений режим відладки;
– перегляд вихідної інформації від командної станції.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Основні нормативи для побудови моделей залізниць – URL: https://ferrum.at.ua/publ/modeli_zaliznic/osnovni_normativi_dlja_pobudovi_modelej_zaliznic/13-1-0-811 (дата звернення: 25.04.25)

2. Terry Flynn. Manually switched DC control systems – URL: https://www.angelfire.com/clone/rail/control.html (дата звернення: 25.04.25)

3. History of Digital Command Control – URL: https://dccwiki.com/DCC_History (дата звернення: 25.04.25)

4. The Beginner's Guide to Digital Command Control – URL: https://dccwiki.com/Introduction_to_DCC (дата звернення: 25.04.25)

5. Український клуб залізничного моделювання «Київський модуль» – URL: https://moemisto.ua/kiev/ukrayinskiy-klub-zaliznichnogo-modelyuvannya-kiyivskiy-modul-organization (дата звернення: 25.04.25)

6. Інтернет магазин моделей дитячих залізниць– URL: https://mmodels.com.ua/ua/ (дата звернення: 25.04.25)

7. Інтернет магазин моделей – URL: https://sv-techno.com.ua/index.php?categoryID=448 (дата звернення: 25.04.25)

8. Інтернет магазин моделей – URL:https://modelkits.com.ua/masshtab-1-87-ho/ (дата звернення: 25.04.25)

9. Macko, D., Hrmo, Ľ., Hrbček, J., & Nagy, P. (2025). Real-Time Control System for Model Railway Based on SIMIS W Interlocking System. Applied Sciences, 15(1), 180. URL: https://www.mdpi.com/2076-3417/15/1/180 (дата звернення: 25.04.25)

10. Advances in Intelligent Systems and Computing – URL: https://link.springer.com/chapter/10.1007/978-3-031-09070-7_14 (дата звернення: 25.04.25)

11. Model Railway DCC Arduino Wireless Commands on a Dead Rail System – URL:https://www.instructables.com/Model-Railway-DCC-Wireless-Commands-on-a-Dead-Rail (дата звернення: 25.04.25)

12. DCC decoder for model railway – URL: https://patents.google.com/patent/US6320346B1/en (дата звернення: 25.04.25)

13. Verification of railway interlocking systems – URL: https://arxiv.org/abs/1506.03554 (дата звернення: 25.04.25)

14. Horáček, J., Rybička, J., & Čížek, R. (2024). New Method of Electronic Control of Model Railroad Track. WSEAS Transactions on Information Science and Applications, 21, 1–10. https://wseas.com/journals/isa/2024/a025109-001(2024).pdf (дата звернення: 25.04.25)

15. Kumar, A., & Samyal, V. K. (2024). An In-Depth Analysis of Flutter for Cross-Platform Mobile App Development. International Research Journal of Modernization in Engineering, Technology and Science, 6(11), 6196–6205. https://www.irjmets.com/uploadedfiles/paper//issue_11_november_2024/64069/final/fin_irjmets1733761184.pdf (дата звернення: 25.04.25)

16. Santhosh, A., Tiji, A., Anuraj, T. R., Thomas, N., Anoop, A., & Varghese, T. (2024). Cross-Platform Innovation: The Rise and Impact of Flutter in Modern App Development. International Research Journal on Advanced Engineering and Management, 2(12), 3560–3569.

https://www.researchgate.net/publication/387018839_Cross-Platform_Innovation_The_Rise_and_Impact_of_Flutter_in_Modern_App_Development (дата звернення: 25.04.25)

17. Ursu, M.P. Digital electronic modules for command and control of miniature railway systems. In Proceedings of the 2021 16th International Conference on Engineering of Modern Electric Systems (EMES), Oradea, Romania, 10–11 June 2021; pp. 1–4.

18. Hei, X.; Zhao, K.; Ma, W.; Xie, G.; Wang, L. A real-time model of railway interlocking system based on UML extension mechanism. In Proceedings of the 2013 4th International Conference on Software Engineering and Service Science, Beijing, China, 23–25 May 2013; pp. 19–22.

19. Command Station – URL: https://dccwiki.com/Command_Station (дата звернення: 25.04.25)

20. OpenDCC – A Command Station for DCC – URL:https://www.opendcc.de/elektronik/opendcc/opendcc_e.html (дата звернення: 25.04.25)

21. Model Railway – DCC Command Station Using Arduino – URL:https://www.instructables.com/DCC-Command-Station-Introduction/ (дата звернення: 25.04.25)

22. DCC Command Stations – URL: https://rowland.id.au/dcc-command-stations/ (дата звернення: 25.04.25)

23. NanoX-S88 XpressNet Command Station – URL: http://usuaris.tinet.cat/fmco/home_en.htm. (дата звернення: 25.04.25)

24. PIC16F627A/628A/648A Data Sheet Flash-Based, 8-Bit CMOS Microcontrollers with nanoWatt Technology – URL: https://ww1.microchip.com/downloads/en/devicedoc/40044e.pdf (дата звернення: 25.04.25)

25. Electronic Components Datasheet Search – URL: https://www.alldatasheet.com/datasheet–pdf/pdf/555706/STMICROELECTRONICS/L6203.html (дата звернення: 25.04.25)

26. HC-06 Bluetooth Module – URL: https://components101.com/wireless/hc-06-bluetooth-module-pinout-datasheet (дата звернення: 25.04.25)

27. ESP-12 Datasheet – URL: https://www.alldatasheet.com/view.jsp?Searchword=ESP-12&sField=2 (дата звернення: 25.04.25)

28. Free Software Microcontroller Programmer – URL: https://www.winpic800.com/?lang=en (дата звернення: 25.04.25)

29. Створення додатку на Flutter: перші кроки – URL: https://dou.ua/lenta/articles/flutter-first-steps (дата звернення: 25.04.25)

30. Про Flutter, коротко: основи – URL: https://devzone.org.ua/post/pro-flutter-korotko-osnovi (дата звернення: 25.04.25)

31. Документація Flutter – URL: https://docs.flutter.dev (дата звернення: 25.04.25)

32. A tour of the Dart language – URL: https://dart.dev/guides/language/language-tour (дата звернення: 25.04.25)

33. Effective Dart – URL: https://dart.dev/guides/language/effective-dart (дата звернення: 25.04.25)

34. Downloads. Arduino IDE 2.3.6 – URL: https://www.arduino.cc/en/software (дата звернення: 25.04.25)

35. Офіційна вікіпедія протоколу XpressNet – URL: https://dccwiki.com/xpressnet (дата звернення: 25.04.25)

36. Специфікація протоколу XpressNet – URL: https://wiki.rocrail.net/lib/exe/fetch.php?media=xpressnet:xpressnet-v2.pdf (дата звернення: 25.04.25)

Додаток А

(інформаційний)

Зовнішній вигляд зібраної командної станції приведений на рисунку А.1

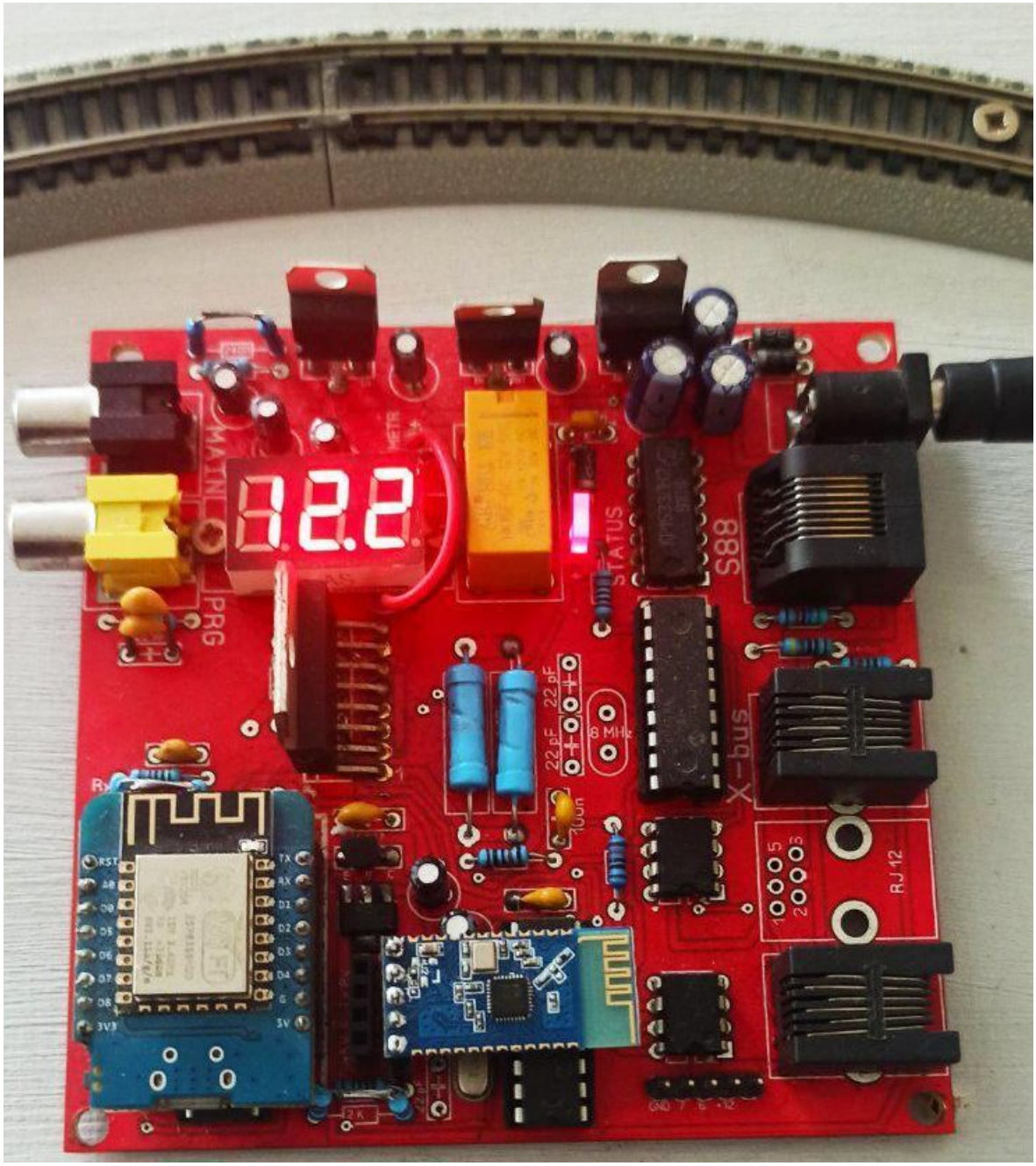
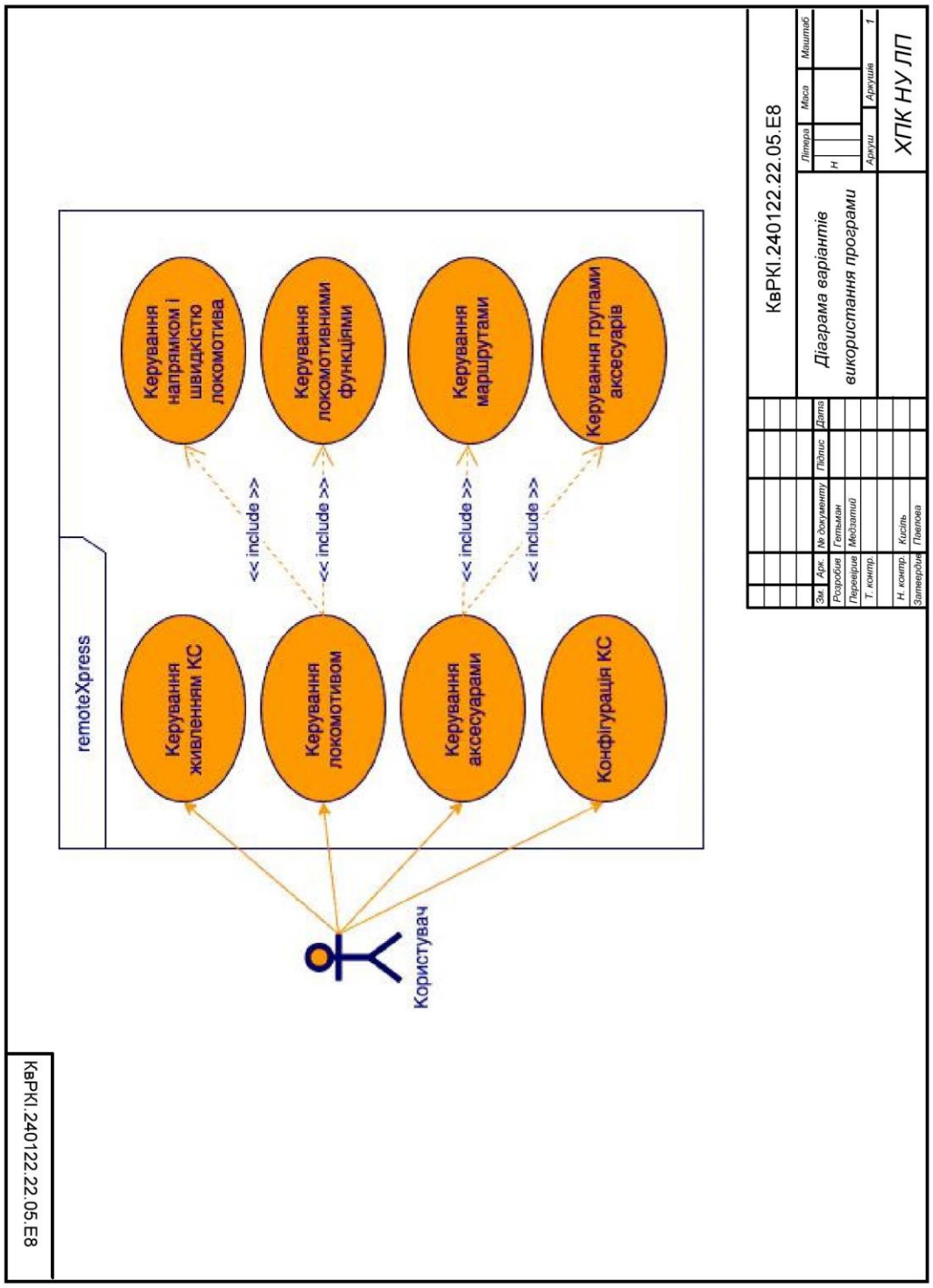


Рисунок А.1 – Зовнішній вигляд зібраної командної станції

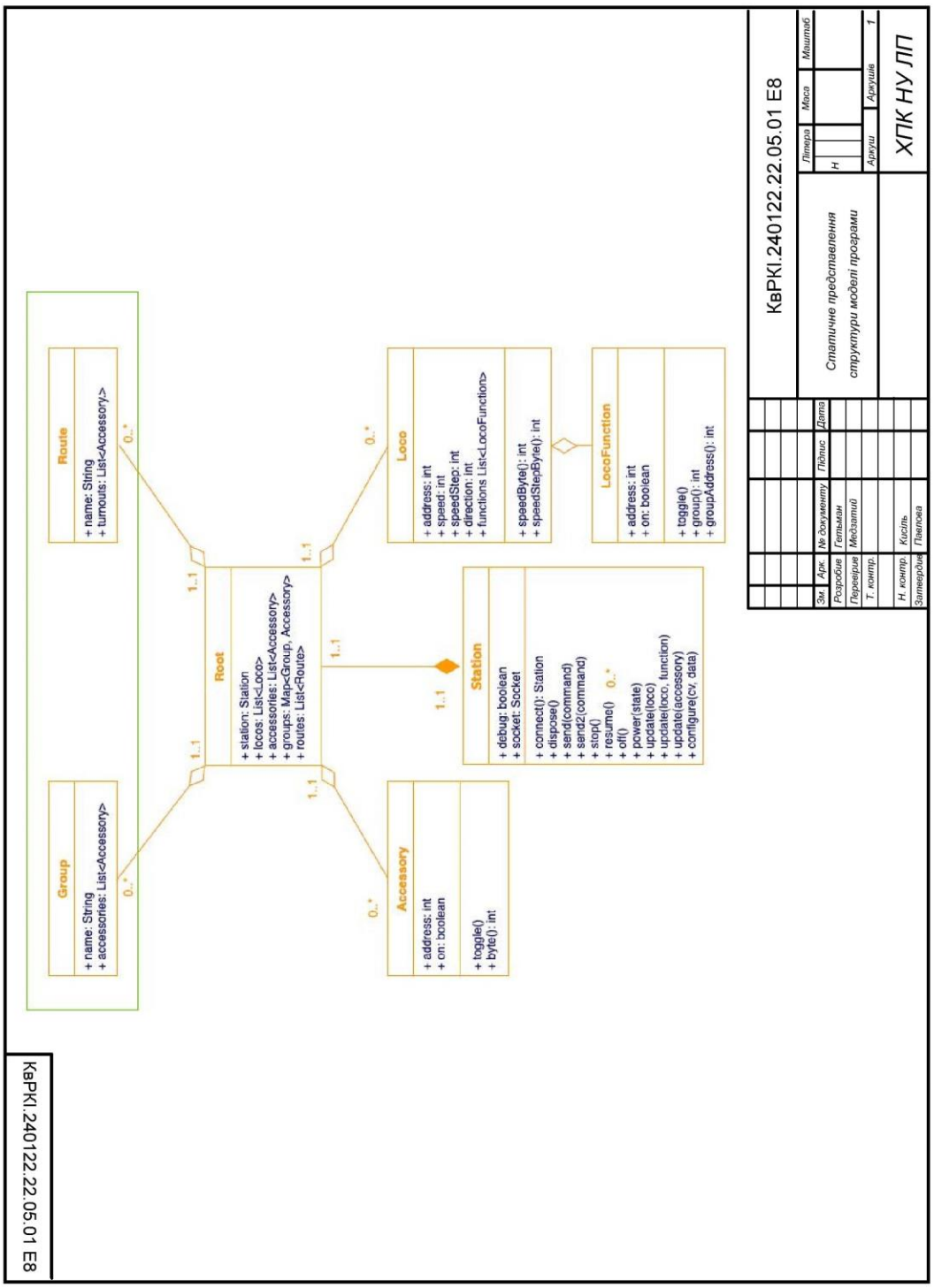
Додаток Б
(обов'язковий)

КОПІЯ КРЕСЛЕННЯ «ДІАГРАМА ВАРІАНТІВ ВИКОРИСТАННЯ ПРОГРАМИ»



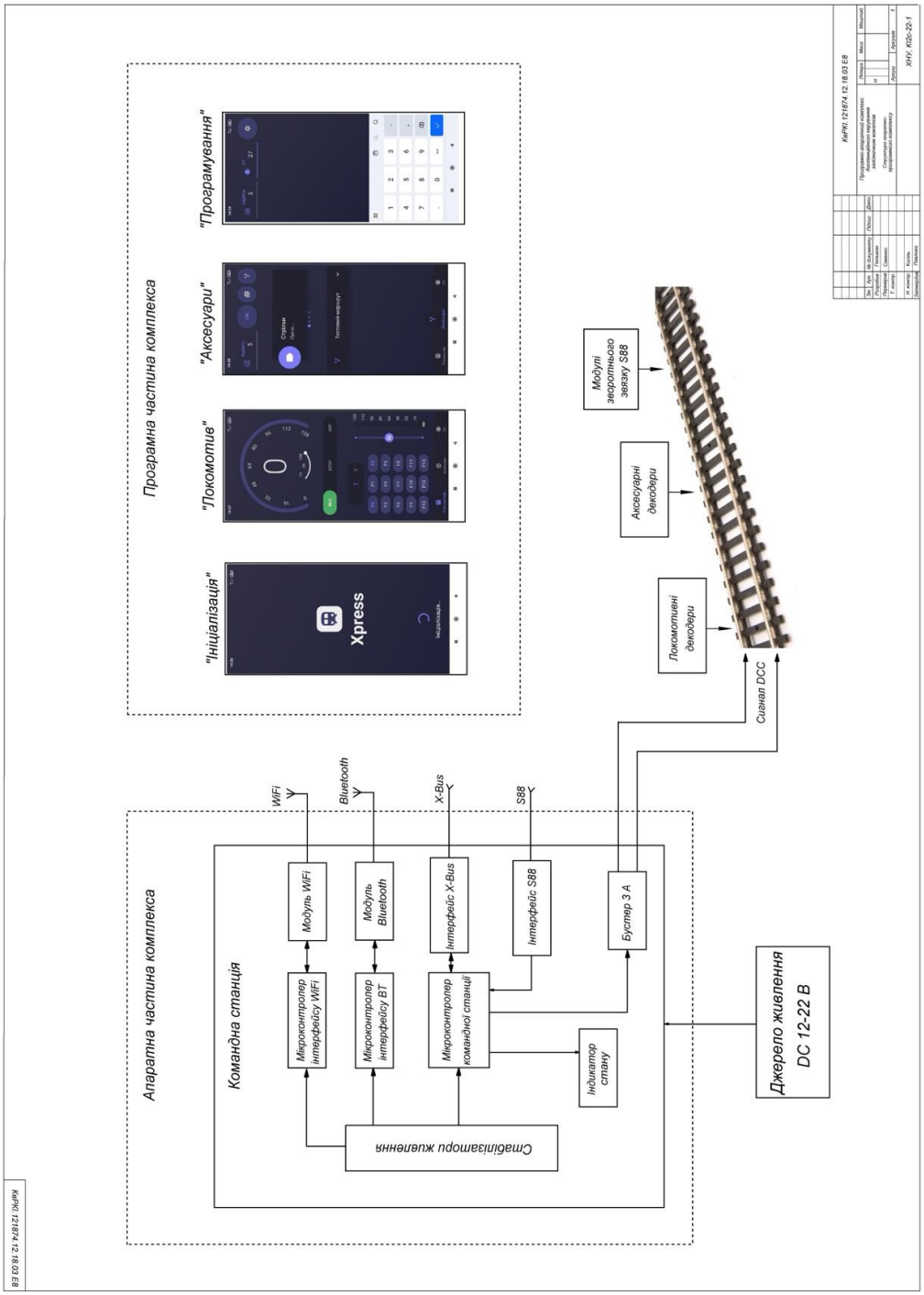
Додаток В
(обов'язковий)

КОПІЯ КРЕСЛЕННЯ «СТАТИЧНЕ ПРЕДСТАВЛЕННЯ СТРУКТУРИ МОДЕЛІ ПРОГРАМИ»



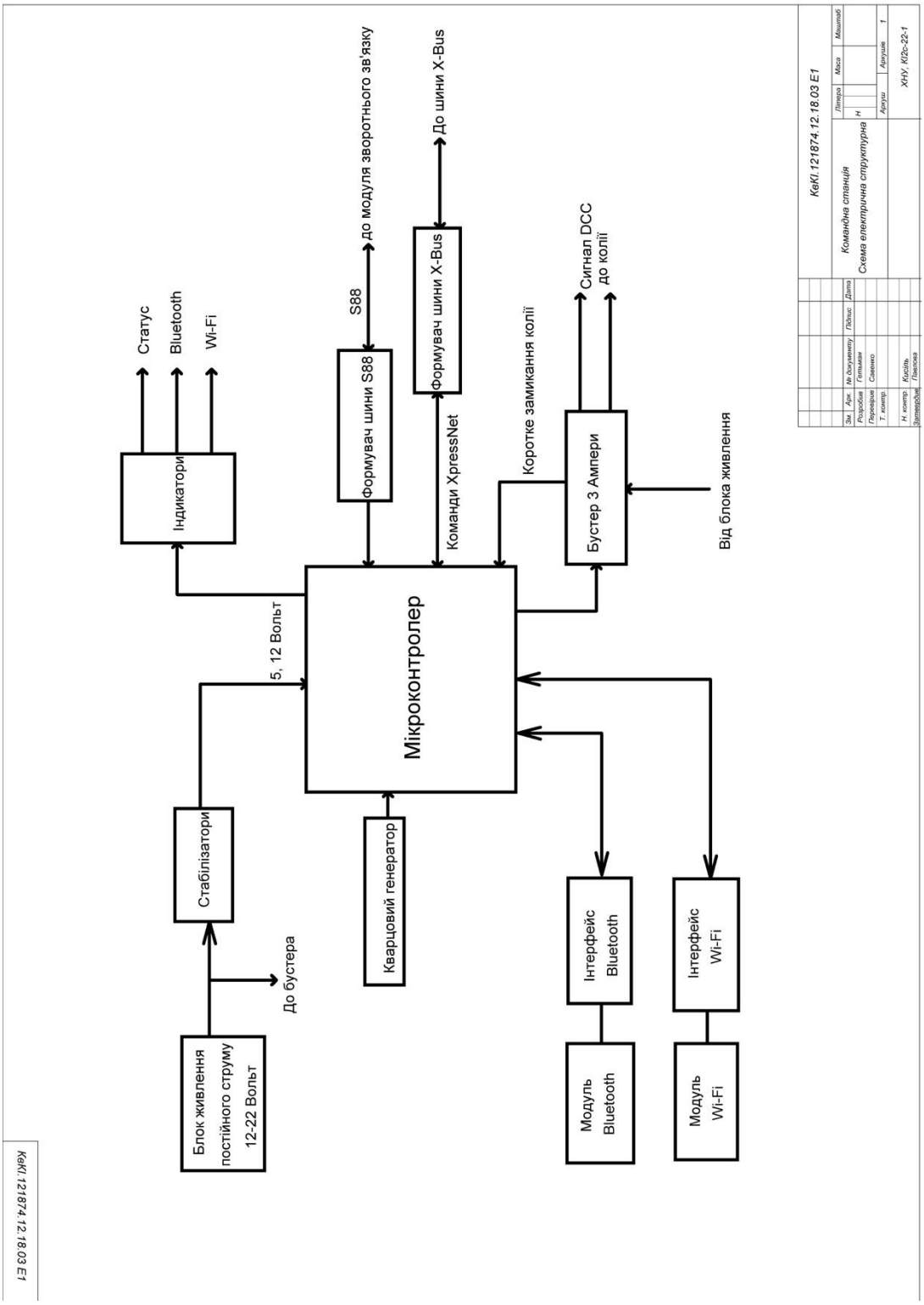
Додаток Г
(обов'язковий)

КОПІЯ КРЕСЛЕННЯ «СТРУКТУРА АПАРАТНО– ПРОГРАМНОГО КОМПЛЕКСУ»



Додаток Д
(обов`язковий)

КОПІЯ КРЕСЛЕННЯ «КОМАНДНА СТАНЦІЯ. СХЕМА ЕЛЕКТРИЧНА СТРУКУРНА»



Додаток Е
(обов`язковий)
Код програми дистанційного керування макетом

```
==> ./lib/net <==

==> ./lib/net/command.dart <==
abstract class Command {
  List<int> bytes();
}

class XorCommand implements Command {
  List<int> _bytes;

  XorCommand(this._bytes);

  List<int> bytes() {
    int xor = 0;
    _bytes.forEach((byte) => xor ^= byte);
    _bytes.add(xor);
    return _bytes;
  }
}

==> ./lib/net/loco.dart <==
class LocoDirections {
  static const int forward = 1;
  static const int neutral = 0;
  static const int reverse = -1;
}

class LocoFunction {
  final int f;
  bool on;

  LocoFunction(this.f, [this.on = false]);

  void toggle() {
    on = !on;
  }

  int group() {
    return f > 20 ? 0x28 : (f > 16 ? 0x23 : 0x20 + (f - 1) ~/ 4);
  }

  int groupAddress() {
    if (f == 0) return 3;
    return 8 - (f > 4 ? (f - 4 * (f - 1) ~/ 4) : f);
  }
}

class Loco {
  static const speedSteps = [14, 28, 128];

  final int address;

  int speed = 0;
  int speedStep = speedSteps[1];
  int direction = LocoDirections.neutral;
  List<LocoFunction> functions = List.generate(28, (i) => LocoFunction(i));

  Loco(this.address);

  void toggleFunction(int f) {
    functions[f].toggle();
  }

  int speedByte() {
    if (speed == 0) return 0;

    bool forward = directi

on == LocoDirections.forward;
    int byte = forward ? 0
x81 : 0x01;
    if (speed % 2 == 0) {


  byte += 0x10 + speed ~/ 2;
    } else {
      byte += (speed + 1) ~/ 2;
    }

    return byte;
  }

  int speedStepByte() {
    const bytes = {14: 0x10, 28: 0x12, 128: 0x13};
    return bytes[speedStep] ?? 0;
  }

  int functionsByte(int group) {
    int byte = 1;
    functions.where((f) => f.group() == group).forEach((f) {
      byte = (byte << 1) + (f.on ? 1 : 0);
    });
    return byte;
  }
}

==> ./lib/net/station.dart <==
import 'dart:io';

import 'package:remotexpress/net/accessory.dart';
import 'package:remotexpress/net/command.dart';
import 'package:remotexpress/net/loco.dart';

class StationPower {
  static const idle = 0;
  static const stop = 1;
  static const off = 2;
}

class Station {
  static const defaultIp = '192.168.4.1';
  static const defaultPort = 333;

  bool _debug = false;
  late Socket _socket;

  Station(this._socket);
  Station.todo() {
    _debug = true;
  }

  static Future<Station> connect() async {
    // ignore: close_sinks
    final      socket      =      await
Socket.connect(defaultIp,
defaultPort);
    return Station(socket);
  }

  void send(Command command) {
    if (_debug) return;
    _socket.add(command.bytes());
  }

  void send2(Command command) {
    if (_debug) return;
    final bytes = command.bytes();
    _socket.add(bytes);
    _socket.add(bytes);
  }

  void close() async {
    await _socket.close();
```

```
  }

  void stop() {
    send2(XorCommand([0x80, 0x80]));
  }

  void resume() {
    send2(XorCommand([0x21, 0x81, 0xA0]));
  }

  void off() {
    send(XorCommand([0x21, 0x81, 0xA0]));
    send2(XorCommand([0x21, 0x80, 0xA1]));
  }

  void power(int power) {
    switch (power) {
      case StationPower.idle:
        resume();
        break;
      case StationPower.stop:
        stop();
        break;
      case StationPower.off:
        off();
        break;
    }
  }

  void configure(int cv, int data) {
    // XpressNet 3.6
    if (cv == 1024) cv = 0;

    send(XorCommand([0x23, 0x1C + (cv ~/ 256), cv, data]));
    resume();

    // XpressNet 3.0
    if (cv >= 256) cv = 0;

    send(XorCommand([0x23, 0x16, cv, data]));
    resume();
  }

  void updateLoco(Loco loco, [bool functions = false]) {
    send(XorCommand([
      0xE4,
      loco.speedStepByte(),
      0, // always zero
      loco.address,
      loco.speedByte(),
    ]));
  }

  void updateLocoFunction(Loco loco, int f) {
    int group = loco.functions[f].group();
    send(XorCommand([
      0xE4,
      group,
      0, // always zero
      loco.address,
      loco.functionsByte(group),
    ]));
  }

  void updateAccessory(Accessory accessory) {
    send2(XorCommand([
      0x52,
      (accessory.a - 1) ~/ 4,
      accessory.byte(),
    ]));
  }
}

==> ./lib/net/accessory.dart <==
class Accessory {
  final int a;
  bool on;

  Accessory(this.a, [this.on = false]);

  void toggle() {
    on = !on;
  }

  int byte() {
    return 0x88 + ((a - 1) % 4 * 2 + (on ? 1 : 0));
  }
}

==> ./lib/formatters <==

==> ./lib/formatters/range.dart <==
import 'package:flutter/services.dart';

class RangeTextInputFormatter extends TextInputFormatter {
  int min;
  int max;

  RangeTextInputFormatter({required int min, required int max})
      : this.min = min,
        this.max = max;

  @override
  TextEditingValue formatEditUpdate(
    TextEditingValue oldValue,
    TextEditingValue newValue,
  ) {
    if (newValue.text == '') return newValue;

    final number = int.parse(newValue.text);
    if (number < min) return TextEditingValue().copyWith(text: min.toString());
    if (number > max) return TextEditingValue().copyWith(text: max.toString());

    final text = number.toString();
    final selection = TextSelection.collapsed(offset: text.length);
    return newValue.copyWith(text: text, selection: selection);
  }
}

==> ./lib/models <==

==> ./lib/models/route.g.dart <==
// GENERATED CODE - DO NOT MODIFY BY HAND

part of 'route.dart';

// **************************************************************************
// JsonSerializableGenerator
// **************************************************************************

Route _$RouteFromJson(Map<String, dynamic> json) => Route(
      json['name'] as String,
      (json['turnouts'] as List<dynamic>?)
          ?.map((e) => Accessory.fromJson(e as Map<String, dynamic>))
          .toList(),
    )..expanded = json['expanded'] as bool;

Map<String, dynamic> _$RouteToJson(Route instance) => <String, dynamic>{
      'name': instance.name,
      'turnouts': instance.turnouts,
      'expanded': instance.expanded,
    };

==> ./lib/models/group.dart <==
import 'package:flutter/material.dart';
import 'package:json_annotation/json_annotation.dart';
import 'package:remotexpress/models/accessory.dart';

part 'group.g.dart';

@JsonSerializable()
class Group {
  String name;
  int iconCodePoint;
  List<Accessory> accessories = [];

  Group(
    this.name,
    this.iconCodePoint, [
    List<Accessory>? accessories,
  ]) : this.accessories = accessories ?? [];
```

```
  IconData icon() {
    return IconData(iconCodePoint, fontFamily: 'MaterialIcons');
  }

  factory Group.fromJson(Map<String, dynamic> json) => _$GroupFromJson(json);
  Map<String, dynamic> toJson() => _$GroupToJson(this);
}

==> ./lib/models/accessory.g.dart <==
// GENERATED CODE - DO NOT MODIFY BY HAND

part of 'accessory.dart';

// **************************************************************************
// JsonSerializableGenerator
// **************************************************************************

Accessory _$AccessoryFromJson(Map<String, dynamic> json) => Accessory(
      json['a'] as int,
      json['on'] as bool? ?? false,
    );

Map<String, dynamic> _$AccessoryToJson(Accessory instance) => <String, dynamic>{
      'a': instance.a,
      'on': instance.on,
    };

==> ./lib/models/route.dart <==
import 'package:json_annotation/json_annotation.dart';
import 'package:remotexpress/models/accessory.dart';

part 'route.g.dart';

@JsonSerializable()
class Route {
  final String name;
  List<Accessory> turnouts = [];
  bool expanded = false;

  Route(this.name, [List<Accessory>? turnouts])
      : this.turnouts = turnouts ?? [];

  factory Route.fromJson(Map<String, dynamic> json) => _$RouteFromJson(json);
  Map<String, dynamic> toJson() => _$RouteToJson(this);
}

==> ./lib/models/group.g.dart <==
// GENERATED CODE - DO NOT MODIFY BY HAND

part of 'group.dart';

// **************************************************************************
// JsonSerializableGenerator
// **************************************************************************

Group _$GroupFromJson(Map<String, dynamic> json) => Group(
      json['name'] as String,
      json['iconCodePoint'] as int,
      (json['accessories'] as List<dynamic>?)
          ?.map((e) => Accessory.fromJson(e as Map<String, dynamic>))
          .toList(),
    );

Map<String, dynamic> _$GroupToJson(Group instance) => <String, dynamic>{
      'name': instance.name,
      'iconCodePoint': instance.iconCodePoint,
      'accessories': instance.accessories,
    };

==> ./lib/models/accessory.dart <==
import 'package:json_annotation/json_annotation.dart';
import 'package:remotexpress/net/accessory.dart' as net;

part 'accessory.g.dart';

@JsonSerializable()
class Accessory extends net.Accessory {
  bool played = false;

  Accessory(int a, [bool on = false]) : super(a, on);

  factory Accessory.fromJson(Map<String, dynamic> json) =>
      _$AccessoryFromJson(json);
  Map<String, dynamic> toJson() => _$AccessoryToJson(this);
}

==> ./lib/main.dart <==
import 'dart:io';
import 'dart:ui';

import 'package:clippy_flutter/arc.dart';
import 'package:window_size/window_size.dart' as window;
import 'package:flutter/material.dart';
import 'package:flutter/services.dart';
import 'package:remotexpress/l10n.dart';

import 'package:remotexpress/pages/accessories/accessories.dart';
import 'package:remotexpress/pages/debug/debug.dart';
import 'package:remotexpress/pages/launch.dart';
import 'package:remotexpress/pages/locomotive/locomotive.dart';

void main() {
  WidgetsFlutterBinding.ensureInitialized();

  if (Platform.isWindows || Platform.isLinux || Platform.isMacOS) {
    const size = Size(420, 740);
    window.setWindowMinSize(size);
  }

  runApp(App());
}

class App extends StatelessWidget {
  static const String title = 'remoteXpress';

  static const Color backgroundColor = Color.fromARGB(0xff, 33, 33, 47);
  static const Color primaryColor = Color.fromARGB(0xff, 125, 123, 250);
  static const Color primaryColorDark = Color.fromARGB(0xff, 60, 62, 107);

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: title,
      themeMode: ThemeMode.light,
      localizationsDelegates: L10n.localizationsDelegates,
      supportedLocales: L10n.supportedLocales,
      theme: ThemeData(
        brightness: Brightness.light,
        canvasColor: Colors.transparent,
        backgroundColor: backgroundColor,
        primaryColor: primaryColor,
        primaryColorDark: primaryColorDark,
        colorScheme: ColorScheme.light(primary: primaryColor),
        bottomNavigationBarTheme: BottomNavigationBarThemeData(
          backgroundColor: backgroundColor,
          selectedItemColor: primaryColor,
          unselectedItemColor: Colors.white54,
        ),
        floatingActionButtonTheme: FloatingActionButtonThemeData(
          elevation: 2,
          focusElevation: 2,
          hoverElevation: 2,
          foregroundColor: primaryColor,
          backgroundColor: backgroundColor,
          splashColor: Colors.transparent,
        ),
      ),
      debugShowCheckedModeBanner: false,
      home: HomePage(title: title),
    );
  }
}

class HomePage extends StatefulWidget {
  final String title;

  HomePage({Key? key, required this.title}) : super(key: key);

  @override
  _HomePageState createState() => _HomePageState();
}
```

```
class _HomePageState extends State<HomePage> {
  bool debug = false;
  bool connected = false;
  late Widget bodyWidget;

  List<Widget> pages = [];
  int selectedPage = 0;

  void onNavigationItem(int index) {
    setState(() {
      selectedPage = index;
    });
  }

  @override
  Widget build(BuildContext context) {
    SystemChrome.setPreferredOrientations([
      DeviceOrientation.portraitUp,
    ]);
    SystemChrome.setSystemUIOverlayStyle(SystemUiOverlayStyle(
      statusBarColor: Colors.transparent,
    ));

    return GestureDetector(
      onTap: () => FocusManager.instance.primaryFocus?.unfocus(),
      child: Scaffold(
        extendBody: true,
        body: Container(
          decoration: BoxDecoration(
            gradient: LinearGradient(
              begin: Alignment.bottomLeft,
              end: Alignment.topRight,
              colors: [
                Color(0xff202139),
                Color(0xff313045),
              ],
            ),
          ),
          child: SafeArea(
            child: AnimatedSwitcher(
              duration: const Duration(seconds: 1),
              child: !connected
                  ? LaunchPage(
                      debug: debug,
                      onLaunched: (station) {
                        pages.add(LocomotivePage(station));
                        pages.add(AccessoriesPage(station));
                        pages.add(DebugPage());
                        setState(() => connected = true);
                      },
                    )
                  : Container(
                      padding: EdgeInsets.only(bottom: 10),
                      child: IndexedStack(
                        index: selectedPage,
                        children: pages,
                      ),
                    ),
            ),
          ),
        ),
        bottomNavigationBar: connected
            ? ClipRRect(
                borderRadius: BorderRadius.only(
                  topLeft: Radius.circular(30.0),
                  topRight: Radius.circular(30.0),
                ),
                child: BottomNavigationBar(
                  elevation: 5,
                  currentIndex: selectedPage,
                  onTap: onNavigationItem,
                  items: [
                    BottomNavigationBarItem(
                      icon: Icon(Icons.train),
                      label: L10n.of(context)!.navigationLocomotive,
                      tooltip: '',
                    ),
                    BottomNavigationBarItem(
                      icon: selectedPage != 1
                          ? Icon(Icons.memory)
                          : Container(width: 20, height: 20),
                      label: L10n.of(context)!.navigationAccessories,
                      tooltip: '',
                    ),
                    BottomNavigationBarItem(
                      icon: Icon(Icons.amp_stories),
                      label: L10n.of(context)!.navigationCV,
                      tooltip: '',
                    ),
                  ],
                ),
              )
            : null,
        floatingActionButtonLocation: FloatingActionButtonLocation.centerDocked,
        floatingActionButton: selectedPage == 1
            ? FloatingActionButton(
                child: Icon(Icons.alt_route),
                tooltip: L10n.of(context)!.addRouteTooltip,
                onPressed: () {
                  final page = pages[selectedPage];
                  if (page is AccessoriesPage) {
                    page.floatingButtonAction();
                  }
                },
              )
            : null,
      ),
    );
  }
}

==> ./lib/generated_plugin_registrant.dart <==
//
// Generated file. Do not edit.
//

// ignore_for_file: directives_ordering
// ignore_for_file: lines_longer_than_80_chars

import 'package:network_info_plus_web/network_info_plus_web.dart';
import 'package:shared_preferences_web/shared_preferences_web.dart';

import 'package:flutter_web_plugins/flutter_web_plugins.dart';

// ignore: public_member_api_docs
void registerPlugins(Registrar registrar) {
  NetworkInfoPlusPlugin.registerWith(registrar);
  SharedPreferencesPlugin.registerWith(registrar);
  registrar.registerMessageHandler();
}

==> ./lib/l10n.dart <==
import 'package:flutter_gen/gen_l10n/l10n.dart';

typedef L10n = AppLocalizations;

==> ./lib/pages <==

==> ./lib/pages/launch.dart <==
import 'package:flutter/material.dart';
import 'package:google_fonts/google_fonts.dart';
import 'package:network_info_plus/network_info_plus.dart';

import 'package:remotexpress/l10n.dart';
import 'package:remotexpress/net/station.dart';
import 'package:remotexpress/widgets/custom_dialog.dart';
import 'package:remotexpress/widgets/logo.dart';

class LaunchPage extends StatefulWidget {
  final bool debug;
  final void Function(Station)? onLaunched;

  LaunchPage({this.debug = false, this.onLaunched});

  @override
  _LaunchPageState createState() => _LaunchPageState();
}

class _LaunchPageState extends State<LaunchPage> {
  static const defaultWifiName = 'dccXpress';
  _LaunchStatus status = _LaunchStatus();
```

```
  Future<Station?> launch() async {
    // 1. Checking WiFi
    setState(() => this.status.next());

    final network = NetworkInfo();
    final wifiName = await network.getWifiName();

    if (wifiName != defaultWifiName) {
      CustomDialog.error(
        context,
        title: L10n.of(context)!.errorBadWiFiTitle,
        content: L10n.of(context)!.errorBadWiFiContent,
        positiveText: L10n.of(context)!.errorBadWiFiPositive,
        onPositivePressed: retryLaunch,
      );
      return null;
    }

    // 2. Connecting to the socket
    setState(() => this.status.next());

    try {
      return await Station.connect();
    } catch (e) {
      CustomDialog.error(
        context,
        title: L10n.of(context)!.errorBadConnectionTitle,
        content: L10n.of(context)!.errorBadConnectionContent,
        positiveText: L10n.of(context)!.errorBadConnectionPositive,
        onPositivePressed: retryLaunch,
      );

      print(e);
      return null;
    }
  }

  Future tryLaunch() async {
    if (widget.debug) {
      widget.onLaunched?.call(Station.todo());
      return;
    }

    final station = await launch();
    if (station != null) {
      station.resume();
      widget.onLaunched?.call(station);
    }
  }

  Future retryLaunch() async {
    Navigator.of(context).pop();
    status.reset();
    await tryLaunch();
  }

  @override
  Widget build(BuildContext context) {
    return Stack(
      fit: StackFit.expand,
      children: [
        Logo.animated(tryLaunch),
        Column(
          mainAxisAlignment: MainAxisAlignment.end,
          children: [
            CircularProgressIndicator(),
            Padding(padding: EdgeInsets.only(top: 20)),
            Text(
              status.text(context) + "...",
              style: GoogleFonts.lato(
                fontSize: 18.0,
                color: Colors.white,
              ),
            ),
            Padding(padding: EdgeInsets.only(top: 30)),
          ],
        ),
      ],
    );
  }
}
```

```
class _LaunchStatus {
  static const statuses = [
    'initializing',
    'checking',
    'connecting',
    'reading',
  ];

  late String current;
  int index = 0;

  _LaunchStatus() {
    next();
  }

  void next() {
    if (index == statuses.length) return;
    current = statuses[index];
    index += 1;
  }

  void reset() {
    index = 0;
  }

  String value() {
    return current;
  }

  String text(BuildContext context) {
    switch (current) {
      case 'initializing':
        return L10n.of(context)!.launchStatusInitializing;
      case 'checking':
        return L10n.of(context)!.launchStatusChecking;
      case 'connecting':
        return L10n.of(context)!.launchStatusConnecting;
      case 'reading':
        return L10n.of(context)!.launchStatusReading;
    }
    return '';
  }
}

==> ./lib/pages/locomotive <==

==> ./lib/pages/locomotive/functions.dart <==
import 'dart:convert';

import 'package:flutter/material.dart';
import 'package:icon_picker/icon_picker.dart';
import 'package:remotexpress/l10n.dart';
import 'package:remotexpress/net/loco.dart';
import 'package:remotexpress/widgets/custom_dialog.dart';
import 'package:remotexpress/widgets/toggle_button.dart';
import 'package:shared_preferences/shared_preferences.dart';

class LocomotiveFunctions extends StatefulWidget {
  final int columns, rows, offset;
  final List<LocoFunction> functions;
  final void Function(int)? onToggle;
  final Widget Function(int) childBuilder;

  LocomotiveFunctions({
    required this.columns,
    required this.rows,
    this.offset = 0,
    required this.functions,
    required this.childBuilder,
    this.onToggle,
  });

  @override
  _LocomotiveFunctionsState createState() => _LocomotiveFunctionsState();
}

class _LocomotiveFunctionsState extends State<LocomotiveFunctions> {
  static const iconCollection = {
    'lightbulb': Icons.lightbulb,
    'volume_up': Icons.volume_up,
```

```
  'air': Icons.air,
  'traffic_rounded': Icons.traffic_rounded,
};

late SharedPreferences prefs;
Map<int, String> icons = {};

void setPrefs() {
  Future.delayed(Duration.zero, () async {
    await prefs.setStringList('functions', icons.values.toList());
  });
}

@override
void initState() {
  Future.delayed(Duration.zero, () async {
    prefs = await SharedPreferences.getInstance();
    if (prefs.containsKey('functions')) {
      List<String> list = prefs.getStringList('functions')!;
      icons = Map.fromIterable(list, key: (v) => list.indexOf(v));
    }
    setState(() {});
  });

  super.initState();
}

void onLongPress(int f) {
  CustomDialog.show(
    context,
    title: L10n.of(context)!.dialogFunctionTitle,
    icon: Icons.info,
    child: _iconPicker(f),
    negativeText: L10n.of(context)!.dialogFunctionNegative,
    onNegativePressed: () {
      setState(() => icons.remove(f));
      setPrefs();
      CustomDialog.pop(context);
    },
  );
}

Widget _iconPicker(int f) {
  return IconPicker(
    initialValue: icons[f],
    title: L10n.of(context)!.dialogFunctionTitle,
    cancelBtn: L10n.of(context)!.dialogPositive,
    decoration: InputDecoration(
      icon: Icon(Icons.apps),
      labelStyle: TextStyle(color: Colors.grey[400]),
      labelText: L10n.of(context)!.dialogFunctionLabel,
      floatingLabelStyle: TextStyle(
        color: Theme.of(context).primaryColorDark,
      ),
      enabledBorder: UnderlineInputBorder(
        borderSide: BorderSide(color: Colors.grey[400]!),
      ),
      focusedBorder: UnderlineInputBorder(
        borderSide: BorderSide(color: Theme.of(context).primaryColorDark),
      ),
    ),
    style: TextStyle(color: Colors.grey[400], fontSize: 15),
    enableSearch: false,
    iconCollection: iconCollection,
    onChanged: (v) {
      setState(() => icons[f] = jsonDecode(v)['iconName']);
      setPrefs();
    },
  );
}

@override
Widget build(BuildContext context) {
  return Row(
    mainAxisAlignment: MainAxisAlignment.center,
    children: List.generate(
      widget.columns,
      (i) => Padding(
        padding: EdgeInsets.only(left: 5, right: 5),
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: List.generate(
            widget.rows,
            (j) {
              final f = i + j * widget.columns + widget.offset;
              final iconKey = icons[f];

              return Expanded(
                child: Padding(
                  padding: EdgeInsets.only(top: 5, bottom: 5),
                  child: ToggleButton(
                    on: widget.functions[f].on,
                    child: iconKey != null
                        ? Icon(iconCollection[iconKey])
                        : widget.childBuilder(f),
                    onPressed: widget.onToggle != null
                        ? () => widget.onToggle!(f)
                        : null,
                    onLongPress: () => onLongPress(f),
                  ),
                ),
              );
            },
          ),
        ),
      ),
    ),
  );
}
```

```
==> ./lib/pages/locomotive/direction_track.dart <==
import 'package:flutter/material.dart';
import 'package:google_fonts/google_fonts.dart';
import 'package:remotexpress/net/loco.dart';
import 'package:syncfusion_flutter_core/theme.dart';
import 'package:syncfusion_flutter_sliders/sliders.dart';

class LocomotiveDirectionTrack extends StatefulWidget {
  final int direction;
  final void Function(int)? onChanged;

  LocomotiveDirectionTrack({
    required this.direction,
    required this.onChanged,
  });

  @override
  _LocomotiveDirectionTrackState createState() =>
      _LocomotiveDirectionTrackState();
}

class _LocomotiveDirectionTrackState extends State<LocomotiveDirectionTrack> {
  static final labelStyle = GoogleFonts.lato(
    color: Colors.grey,
    fontSize: 12,
    fontWeight: FontWeight.bold,
  );

  @override
  Widget build(BuildContext context) {
    final Color dividerColor = Theme.of(context).primaryColor;

    return SfSliderTheme(
      data: SfSliderThemeData(
        activeTrackHeight: 8,
        inactiveTrackHeight: 8,
        trackCornerRadius: 4,
        activeDividerRadius: 3,
        inactiveDividerRadius: 3,
        thumbRadius: 16,
        activeLabelStyle: labelStyle,
        inactiveLabelStyle: labelStyle,
        disabledThumbColor: Theme.of(context).backgroundColor,
        activeDividerColor: dividerColor,
        inactiveDividerColor: dividerColor,
      ),
      child: SfSlider.vertical(
        min: LocoDirections.reverse,
        max: LocoDirections.forward,
        value: widget.direction,
        interval: 1,
```

```
        stepSize: 1,
        showDividers: true,
        onChanged: widget.onChanged != null
            ? (v) => widget.onChanged!(v.toInt())
            : null,
        thumbIcon: Center(
          child: Text(
            ['R', 'N', 'F'][widget.direction + 1],
            style: TextStyle(color: Colors.white, fontSize: 18.0),
          ),
        ),
        trackShape: _SfTrackShape(),
      ),
    );
  }
}

class _SfTrackShape extends SfTrackShape {
  @override
  void paint(
    PaintingContext context,
    Offset offset,
    Offset? thumbCenter,
    Offset? startThumbCenter,
    Offset? endThumbCenter, {
    required RenderBox parentBox,
    required SfSliderThemeData themeData,
    SfRangeValues? currentValues,
    dynamic currentValue,
    required Animation<double> enableAnimation,
    required Paint? inactivePaint,
    required Paint? activePaint,
    required TextDirection textDirection,
  }) {
    final ColorTween inactiveTrackColorTween = ColorTween(
      begin: themeData.disabledInactiveTrackColor,
      end: themeData.inactiveTrackColor,
    );

    inactivePaint = Paint()
      ..color = inactiveTrackColorTween.evaluate(enableAnimation)!;

    super.paint(
      context,
      offset,
      thumbCenter?.translate(0.0, 0.0),
      startThumbCenter,
      endThumbCenter,
      parentBox: parentBox,
      themeData: themeData,
      enableAnimation: enableAnimation,
      inactivePaint: inactivePaint,
      activePaint: inactivePaint,
      textDirection: textDirection,
    );
  }
}

==> ./lib/pages/locomotive/speed_track.dart <==
import 'package:flutter/material.dart';
import 'package:google_fonts/google_fonts.dart';
import 'package:syncfusion_flutter_core/theme.dart';
import 'package:syncfusion_flutter_sliders/sliders.dart';

class LocomotiveSpeedTrack extends StatefulWidget {
  final int speedStep;
  final int speed;
  final void Function(int)? onChanged;
  final void Function() onPointerUp;
  final int interval;

  LocomotiveSpeedTrack({
    required this.speedStep,
    required this.speed,
    required this.onChanged,
    required this.onPointerUp,
    this.interval = 1,
  });

  @override
  _LocomotiveSpeedTrackState createState() => _LocomotiveSpeedTrackState();
}

class _LocomotiveSpeedTrackState extends State<LocomotiveSpeedTrack> {
  static final labelStyle = GoogleFonts.lato(
    color: Colors.grey,
    fontSize: 12,
    fontWeight: FontWeight.bold,
  );

  String formatLabel(dynamic value, String s) {
    int speed = widget.speed, interval = widget.interval - 1;
    bool inRange = speed > value - interval && speed < value + interval;
    return value == 0 || inRange ? '' : s;
  }

  @override
  Widget build(BuildContext context) {
    final Color tickColor = Theme.of(context).primaryColor.withOpacity(0.40);
    final Color activeColor = Theme.of(context).primaryColor.withOpacity(0.88);
    final Color disabledColor = Theme.of(context).backgroundColor;

    return Listener(
      onPointerUp: (_) => widget.onPointerUp(),
      child: SfSliderTheme(
        data: SfSliderThemeData(
          thumbRadius: 16,
          trackCornerRadius: 14,
          activeTrackHeight: 14,
          inactiveTrackHeight: 14,
          labelOffset: Offset(6, 0),
          activeLabelStyle: labelStyle.copyWith(color: activeColor),
          inactiveLabelStyle: labelStyle,
          activeTickColor: tickColor,
          inactiveTickColor: tickColor,
          disabledInactiveTickColor: disabledColor,
          disabledThumbColor: disabledColor,
        ),
        child: SfSlider.vertical(
          min: 0,
          max: widget.speedStep,
          value: widget.speed,
          interval: widget.interval.toDouble(),
          showLabels: true,
          showTicks: false,
          labelFormatterCallback: formatLabel,
          onChanged: widget.onChanged != null
              ? (v) => widget.onChanged!(v.toInt())
              : null,
          thumbIcon: Icon(
            Icons.drag_handle,
            color: Colors.white,
            size: 26,
          ),
        ),
      ),
    );
  }
}

==> ./lib/pages/locomotive/selector.dart <==
import 'package:flutter/material.dart';
import 'package:google_fonts/google_fonts.dart';
// ignore: import_of_legacy_library_into_null_safe
import 'package:numberpicker/numberpicker.dart';

class LocomotiveSelector extends StatefulWidget {
  final int loco;
  final void Function(int) onChanged;

  LocomotiveSelector({
    required this.loco,
    required this.onChanged,
  });

  @override
  _LocomotiveSelectorState createState() => _LocomotiveSelectorState();
}

class _LocomotiveSelectorState extends State<LocomotiveSelector> {
  @override
  Widget build(BuildContext context) {
```

```
    return NumberPicker.horizontal(
      minValue: 1,
      maxValue: 256,
      initialValue: widget.loco,
      onChanged: (v) => widget.onChanged(v.toInt()),
      textStyle: GoogleFonts.lato(color: Colors.grey),
      selectedTextStyle: GoogleFonts.lato(
        color: Theme.of(context).primaryColor,
        fontSize: 20,
      ),
    );
  }
}

==> ./lib/pages/locomotive/speedo.dart <==
import 'package:flutter/material.dart';
import 'package:google_fonts/google_fonts.dart';
import 'package:syncfusion_flutter_gauges/gauges.dart';
// ignore: import_of_legacy_library_into_null_safe
import 'package:countup/countup.dart';

class LocomotiveSpeedo extends StatefulWidget {
  final List<int> speedSteps;
  final void Function(int) onSpeedStepChanged;
  final int speed;
  final int speedStep;
  final int visualSpeed;
  final int previousSpeed;

  LocomotiveSpeedo({
    required this.speedSteps,
    required this.onSpeedStepChanged,
    required this.speedStep,
    this.speed = 0,
    this.visualSpeed = 0,
    this.previousSpeed = 0,
  }) {
    assert(speedSteps.length != 0);
  }

  @override
  _LocomotiveSpeedoState createState() => _LocomotiveSpeedoState();

  static int speedInterval(int speedStep) {
    return speedStep == 128 ? 16 : speedStep ~/ 7;
  }
}

class _LocomotiveSpeedoState extends State<LocomotiveSpeedo> {
  late int previousSpeedStep;

  @override
  void initState() {
    super.initState();
    previousSpeedStep = widget.speedStep;
  }

  double speedStepIndex() {
    return widget.speedSteps.indexOf(widget.speedStep) + 1.0;
  }

  void onSpeedStepLabel(AxisLabelCreatedArgs args) {
    args.text = widget.speedSteps[int.parse(args.text) - 1].toString();
  }

  void onSpeedStepChanged(double index) {
    final speedStep = widget.speedSteps[index.round() - 1];
    if (previousSpeedStep == speedStep) return;
    previousSpeedStep = speedStep;
    widget.onSpeedStepChanged(speedStep);
  }

  double speedInterval() {
    return LocomotiveSpeedo.speedInterval(widget.speedStep).toDouble();
  }

  @override
  Widget build(BuildContext context) {
    return SfRadialGauge(
      axes: [
        RadialAxis(
          startAngle: 50,
          endAngle: 130,
          minimum: 1,
          maximum: 3,
          interval: 1,
          radiusFactor: 0.5,
          showAxisLine: true,
          showTicks: false,
          minorTicksPerInterval: 0,
          centerY: 0.7,
          labelOffset: 20,
          isInversed: true,
          onLabelCreated: onSpeedStepLabel,
          ranges: [
            GaugeRange(
              startValue: 1,
              endValue: 3,
              sizeUnit: GaugeSizeUnit.factor,
              startWidth: 0.03,
              endWidth: 0.03,
              color: Colors.transparent,
            ),
          ],
          pointers: [
            MarkerPointer(
              enableDragging: true,
              value: speedStepIndex(),
              onValueChanged: onSpeedStepChanged,
              enableAnimation: false,
              animationDuration: 300,
              markerType: MarkerType.circle,
              markerWidth: 12,
              markerHeight: 12,
              color: Colors.grey[200],
              offsetUnit: GaugeSizeUnit.factor,
              overlayRadius: 12,
            ),
          ],
          axisLineStyle: AxisLineStyle(
            thickness: 0.06,
            thicknessUnit: GaugeSizeUnit.factor,
            cornerStyle: CornerStyle.bothCurve,
            color: Colors.grey[250],
          ),
          axisLabelStyle: GaugeTextStyle(
            color: Colors.grey,
            fontWeight: FontWeight.bold,
            fontSize: 14,
          ),
        ),
        RadialAxis(
          startAngle: 150,
          endAngle: 30,
          radiusFactor: 1,
          canScaleToFit: true,
          canRotateLabels: true,
          showTicks: false,
          minimum: 0,
          maximum: widget.speedStep + 0.01,
          interval: speedInterval(),
          labelOffset: 20,
          ranges: [
            GaugeRange(
              startValue: 0,
              endValue: widget.speedStep.toDouble(),
              sizeUnit: GaugeSizeUnit.factor,
              startWidth: 0.07,
              endWidth: 0.07,
              color: Colors.transparent,
            ),
          ],
          pointers: [
            RangePointer(
              value: widget.visualSpeed.toDouble(),
              enableAnimation: true,
              animationType: AnimationType.ease,
              animationDuration: 500,
              color: Theme.of(context).primaryColor,
              cornerStyle: CornerStyle.bothCurve,
              width: 15,
            ),
```

```
          ],
          annotations: [
            GaugeAnnotation(
              widget: Padding(
                padding: EdgeInsets.only(bottom: 80),
                child: Countup(
                  begin: widget.previousSpeed.toDouble(),
                  end: widget.visualSpeed.toDouble(),
                  duration: Duration(milliseconds: 500),
                  textScaleFactor: 8,
                  style: GoogleFonts.lato(
                    color: Colors.grey[350],
                    fontWeight: FontWeight.w300,
                  ),
                ),
              ),
              positionFactor: 0.03,
            ),
          ],
          axisLineStyle: AxisLineStyle(
            thickness: 0.08,
            thicknessUnit: GaugeSizeUnit.factor,
            cornerStyle: CornerStyle.bothCurve,
            color: Theme.of(context).primaryColor.withOpacity(0.28),
          ),
          axisLabelStyle: GaugeTextStyle(
            color: Colors.grey,
            fontWeight: FontWeight.bold,
            fontSize: 18,
          ),
          majorTickStyle: MajorTickStyle(
            length: 6,
            thickness: 3,
          ),
          minorTickStyle: MinorTickStyle(
            length: 4,
            thickness: 3,
          ),
        ),
      ],
    );
  }
}
```

```
==> ./lib/pages/locomotive/locomotive.dart <==
import 'package:flutter/material.dart';

import 'package:remotexpress/pages/locomotive/functions.dart';
import 'package:remotexpress/pages/locomotive/power.dart';
import 'package:remotexpress/pages/locomotive/direction_track.dart';
import 'package:remotexpress/pages/locomotive/selector.dart';
import 'package:remotexpress/pages/locomotive/speed_track.dart';
import 'package:remotexpress/pages/locomotive/speedo.dart';

import 'package:remotexpress/net/loco.dart';
import 'package:remotexpress/net/station.dart';

class LocomotivePage extends StatefulWidget {
  final Station station;

  LocomotivePage(this.station);

  @override
  _LocomotivePageState createState() => _LocomotivePageState(station);
}

class _LocomotivePageState extends State<LocomotivePage> {
  final Station station;
  final List<Loco> locos = List.generate(256, (i) => Loco(i + 1));
  late Loco loco;

  _LocomotivePageState(this.station) {
    loco = locos[0];
  }

  int power = 0;
  int visualSpeed = 0;
  int previousSpeed = 0;

  void forceSpeed(int value) {
    previousSpeed = visualSpeed;
    loco.speed = visualSpeed = value;
  }

  void onPowerChanged(int value) {
    setState(() {
      power = value;
      if (power != StationPower.idle) {
        loco.direction = LocoDirections.neutral;
        forceSpeed(0);
      }
    });

    station.power(power);
  }

  void onLocoChanged(int value) {
    setState(() => loco = locos[value - 1]);
    onVisualSpeedChanged();
  }

  void onSpeedChanged(dynamic value) {
    if (value > loco.speedStep) return;
    setState(() => loco.speed = value.round());

    station.updateLoco(loco);
  }

  void onVisualSpeedChanged() {
    setState(() {
      previousSpeed = visualSpeed;
      visualSpeed = loco.speed;
    });
  }

  void onSpeedStepChanged(int value) {
    setState(() {
      loco.speedStep = value;
      forceSpeed(0);
    });

    station.updateLoco(loco);
  }

  void onDirectionChanged(int value) {
    setState(() {
      if (loco.direction == value) return;
      if (value != LocoDirections.neutral) {
        power = StationPower.idle;
      }

      loco.direction = value;
      forceSpeed(0);
    });

    station.updateLoco(loco);
  }

  void onFunctionToggle(int f) {
    setState(() => loco.toggleFunction(f));
    station.updateLocoFunction(loco, f);
  }

  @override
  Widget build(BuildContext context) {
    return Column(
      mainAxisAlignment: MainAxisAlignment.start,
      children: [
        Expanded(
          flex: 1,
          child: LocomotiveSpeedo(
            speedSteps: Loco.speedSteps,
            onSpeedStepChanged: onSpeedStepChanged,
            speedStep: loco.speedStep,
            speed: loco.speed,
            visualSpeed: visualSpeed,
            previousSpeed: previousSpeed,
          ),
        ),
        SizedBox(height: 30),
        LocomotivePower(
          power: power,
```

```
          onPowerChanged: onPowerChanged,
        ),
        SizedBox(height: 40),
        Expanded(
          child: Row(
            mainAxisAlignment: MainAxisAlignment.center,
            children: [
              Column(
                mainAxisAlignment: MainAxisAlignment.center,
                children: [
                  Flexible(
                    flex: 1,
                    child: Container(
                      decoration: BoxDecoration(
                        color: Theme.of(context).backgroundColor,
                        borderRadius: BorderRadius.circular(20),
                      ),
                      child: LocomotiveSelector(
                        loco: loco.address,
                        onChanged: onLocoChanged,
                      ),
                    ),
                  ),
                  Padding(padding: EdgeInsets.only(bottom: 10)),
                  Flexible(
                    flex: 4,
                    child: LocomotiveFunctions(
                      columns: 3,
                      rows: 5,
                      functions: loco.functions,
                      onToggle: onFunctionToggle,
                      childBuilder: (f) => Text('F$f'),
                    ),
                  ),
                ],
              ),
              SizedBox(width: 30),
              Row(
                children: [
                  LocomotiveDirectionTrack(
                    direction: loco.direction,
                    onChanged:
                        power != StationPower.off ? onDirectionChanged : null,
                  ),
                  LocomotiveSpeedTrack(
                    speedStep: loco.speedStep,
                    speed: loco.speed,
                    interval: LocomotiveSpeedo.speedInterval(loco.speedStep),
                    onPointerUp: onVisualSpeedChanged,
                    onChanged: loco.direction != LocoDirections.neutral
                        ? onSpeedChanged
                        : null,
                  ),
                ],
              ),
            ],
          ),
        ),
      ],
    );
  }
}

==> ./lib/pages/locomotive/power.dart <==
import 'package:flutter/material.dart';
import 'package:remotexpress/widgets/animated_toggle.dart';

class LocomotivePower extends StatefulWidget {
  final int power;
  final void Function(int) onPowerChanged;

  LocomotivePower({
    required this.power,
    required this.onPowerChanged,
  });

  @override
  _LocomotivePowerState createState() => _LocomotivePowerState();
}

class _LocomotivePowerState extends State<LocomotivePower> {
  static const List<Color> colors = [
    Color.fromARGB(0xff, 77, 172, 100),
    Color.fromARGB(0xff, 234, 192, 49),
    Color(0xffA81B24),
  ];

  @override
  Widget build(BuildContext context) {
    return AnimatedToggle(
      index: widget.power,
      onToggleCallback: (v) => widget.onPowerChanged(v.toInt()),
      values: ['IDLE', 'STOP', 'OFF'],
      heightScale: 1.5,
      buttonColor: colors[widget.power],
      backgroundColor: Theme.of(context).backgroundColor,
      textColor: Colors.white,
    );
  }
}

==> ./lib/pages/accessories <==

==> ./lib/pages/accessories/control.dart <==
import 'package:flutter/material.dart';
import 'package:flutter/services.dart';
import 'package:remotexpress/l10n.dart';
import 'package:remotexpress/formatters/range.dart';
import 'package:remotexpress/net/accessory.dart';
import 'package:remotexpress/widgets/toggle_button.dart';

class AccessoriesControl extends StatefulWidget {
  final List<Accessory> accessories;
  final void Function(int) onToggle, onAddToGroup, onAddToRoute;

  AccessoriesControl({
    required this.accessories,
    required this.onToggle,
    required this.onAddToGroup,
    required this.onAddToRoute,
  });

  @override
  _AccessoriesControlState createState() => _AccessoriesControlState();
}

class _AccessoriesControlState extends State<AccessoriesControl> {
  TextEditingController controller = TextEditingController();

  int address() {
    return (int.tryParse(controller.value.text) ?? 1) - 1;
  }

  bool isValid() {
    if (controller.value.text.isEmpty) return false;
    int i = address();
    return i >= 0 && i < widget.accessories.length;
  }

  Accessory currentAccessory() {
    if (!isValid()) return Accessory(0);
    return widget.accessories[address()];
  }

  @override
  Widget build(BuildContext context) {
    return IntrinsicHeight(
      child: Row(
        mainAxisAlignment: MainAxisAlignment.spaceBetween,
        crossAxisAlignment: CrossAxisAlignment.stretch,
        children: [
          Expanded(
            flex: 5,
            child: TextFormField(
              controller: controller,
              onChanged: (_) => setState(() {}),
              style: TextStyle(
                fontSize: 18,
                color: Colors.grey[300],
              ),
              textAlign: TextAlign.center,
              keyboardType: TextInputType.number,
```

```
              inputFormatters: [
                FilteringTextInputFormatter.digitsOnly,
                RangeTextInputFormatter(min: 1, max: 1024),
              ],
              decoration: InputDecoration(
                labelText: L10n.of(context)!.accessoryLabel,
                labelStyle: TextStyle(color: Colors.grey[400]),
                floatingLabelStyle: TextStyle(
                  color: Theme.of(context).primaryColor,
                ),
                prefixIcon: Icon(
                  Icons.memory,
                  size: 28,
                  color: Theme.of(context).primaryColor,
                ),
                enabledBorder: UnderlineInputBorder(
                  borderSide: BorderSide(color: Colors.grey[400]!),
                ),
              ),
            ),
          ),
          SizedBox(width: 10),
          Expanded(
            flex: 3,
            child: ToggleButton(
              on: currentAccessory().on,
              child: Text(currentAccessory().on ? 'ON' : 'OFF'),
              onPressed: isValid() ? () => widget.onToggle(address()) : null,
            ),
          ),
          SizedBox(width: 10),
          Expanded(
            flex: 2,
            child: Tooltip(
              message: L10n.of(context)!.addToGroupTooltip,
              child: ToggleButton(
                on: false,
                child: Icon(Icons.add_to_photos),
                onPressed:
                    isValid() ? () => widget.onAddToGroup(address()) : null,
              ),
            ),
          ),
          SizedBox(width: 10),
          Expanded(
            flex: 2,
            child: Tooltip(
              message: L10n.of(context)!.addToRouteTooltip,
              child: ToggleButton(
                on: false,
                child: Icon(Icons.alt_route),
                onPressed:
                    isValid() ? () => widget.onAddToRoute(address()) : null,
              ),
            ),
          ),
        ],
      ),
    );
  }
}

==> ./lib/pages/accessories/routes.dart <==
import 'package:expandable/expandable.dart';
import 'package:flutter/material.dart' hide Route;
import 'package:google_fonts/google_fonts.dart';
import 'package:remotexpress/l10n.dart';
import 'package:remotexpress/models/route.dart';
import 'package:remotexpress/pages/accessories/buttons.dart';

class AccessoryRoutes extends StatefulWidget {
  final List<Route> routes;
  final void Function(Route route) onPlay, onDelete;

  AccessoryRoutes({
    required this.routes,
    required this.onPlay,
    required this.onDelete,
  });

  @override
  _AccessoryRoutesState createState() => _AccessoryRoutesState();
}

class _AccessoryRoutesState extends State<AccessoryRoutes> {
  Widget headerRow(BuildContext context, Route route, [bool expanded = false]) {
    return Row(
      mainAxisAlignment: MainAxisAlignment.spaceBetween,
      children: [
        IconButton(
          icon: Icon(expanded ? Icons.play_arrow : Icons.alt_route),
          color: Theme.of(context).primaryColor,
          onPressed: () =>
              expanded ? widget.onPlay(route) : widget.onDelete(route),
        ),
        Text(
          route.name,
          style: GoogleFonts.lato(
            color: Colors.grey[300],
            fontSize: 16,
          ),
        ),
        ExpandableButton(
          child: Icon(
            expanded ? Icons.expand_less : Icons.expand_more,
            color: Colors.white,
          ),
        ),
      ],
    );
  }

  @override
  Widget build(BuildContext context) {
    return ListView(
      physics: BouncingScrollPhysics(),
      children: [
        ...widget.routes.map<ExpandableNotifier>(
          (route) => ExpandableNotifier(
            child: Container(
              margin: EdgeInsets.symmetric(vertical: 5, horizontal: 10),
              padding: EdgeInsets.all(15),
              decoration: BoxDecoration(
                color: Theme.of(context).backgroundColor,
                shape: BoxShape.rectangle,
                borderRadius: BorderRadius.circular(8),
                boxShadow: [
                  BoxShadow(
                    color: Colors.black12,
                    blurRadius: 10,
                    offset: Offset(0, 10),
                  ),
                ],
              ),
              child: ScrollOnExpand(
                child: Column(
                  children: [
                    Expandable(
                      collapsed: headerRow(context, route),
                      expanded: Column(
                        children: [
                          headerRow(context, route, true),
                          SizedBox(height: 5),
                          route.turnouts.length != 0
                              ? Container(
                                  height: 40,
                                  child: SingleChildScrollView(
                                    scrollDirection: Axis.horizontal,
                                    child: AbsorbPointer(
                                      child: AccessoryButtons(
                                        accessories: route.turnouts,
                                        childBuilder: (a) => Text('A$a'),
                                        onToggle: null,
                                      ),
                                    ),
                                  ),
                                )
                              : Padding(
                                  padding: EdgeInsets.only(bottom: 3, left: 20),
                                  child: Text(
                                    L10n.of(context)!.accessoriesEmpty,
                                    style: TextStyle(
```

```
                                color: Colors.grey[300],
                                fontStyle: FontStyle.italic,
                              ),
                            ),
                          ),
                        ],
                      ),
                    ),
                  ],
                ),
              ),
            ),
          ),
        ),
      ],
    );
  }
}

==> ./lib/pages/accessories/buttons.dart <==
import 'package:flutter/material.dart';
import 'package:remotexpress/net/accessory.dart';
import 'package:remotexpress/widgets/toggle_button.dart';

class AccessoryButtons extends StatefulWidget {
  final List<Accessory> accessories;
  final void Function(int)? onToggle;
  final Widget Function(int) childBuilder;

  AccessoryButtons({
    required this.accessories,
    required this.childBuilder,
    this.onToggle,
  });

  @override
  _AccessoryButtonsState createState() => _AccessoryButtonsState();
}

class _AccessoryButtonsState extends State<AccessoryButtons> {
  @override
  Widget build(BuildContext context) {
    final accessories = List.from(widget.accessories);
    accessories.sort((a, b) => a.a - b.a);

    return Row(
      mainAxisAlignment: MainAxisAlignment.center,
      children: accessories.map<Widget>((accessory) {
        return Padding(
          padding: EdgeInsets.all(5),
          child: ToggleButton(
            on: accessory.on,
            child: widget.childBuilder(accessory.a),
            onPressed: widget.onToggle != null || accessory.played
                ? () => widget.onToggle?.call(accessory.a)
                : null,
          ),
        );
      }).toList(),
    );
  }
}

==> ./lib/pages/accessories/groups.dart <==
import 'package:card_swiper/card_swiper.dart';
import 'package:flutter/material.dart';
import 'package:google_fonts/google_fonts.dart';
import 'package:remotexpress/l10n.dart';
import 'package:remotexpress/models/group.dart';
import 'package:remotexpress/pages/accessories/buttons.dart';

class AccessoryGroups extends StatefulWidget {
  final List<Group> groups;
  final void Function(int)? onToggle;

  AccessoryGroups({
    required this.groups,
    required this.onToggle,
  });

  @override
  _AccessoryGroupsState createState() => _AccessoryGroupsState();
}

class _AccessoryGroupsState extends State<AccessoryGroups> {
  Widget buildItem(BuildContext context, int i) {
    return Container(
      margin: EdgeInsets.all(10),
      child: Stack(
        alignment: Alignment.center,
        children: [
          Container(
            height: 124,
            padding: EdgeInsets.only(left: 35, right: 25),
            margin: EdgeInsets.only(left: 30),
            decoration: BoxDecoration(
              color: Theme.of(context).backgroundColor,
              shape: BoxShape.rectangle,
              borderRadius: BorderRadius.circular(8),
              boxShadow: [
                BoxShadow(
                  color: Colors.black12,
                  blurRadius: 10,
                  offset: Offset(0, 10),
                ),
              ],
            ),
            child: Container(
              margin: EdgeInsets.symmetric(vertical: 16),
              constraints: BoxConstraints.expand(),
              child: Column(
                mainAxisAlignment: MainAxisAlignment.center,
                crossAxisAlignment: CrossAxisAlignment.start,
                children: [
                  Container(height: 4),
                  Padding(
                    padding: EdgeInsets.only(left: 40),
                    child: Text(
                      widget.groups[i].name,
                      style: GoogleFonts.lato(
                        color: Colors.white,
                        fontSize: 18,
                      ),
                    ),
                  ),
                  Container(height: 10),
                  Container(
                    height: 40,
                    child: widget.groups[i].accessories.length != 0
                        ? SingleChildScrollView(
                            scrollDirection: Axis.horizontal,
                            padding: EdgeInsets.symmetric(horizontal: 30),
                            child: AccessoryButtons(
                              accessories: widget.groups[i].accessories,
                              childBuilder: (a) => Text('A$a'),
                              onToggle: widget.onToggle,
                            ),
                          )
                        : Padding(
                            padding: EdgeInsets.only(left: 40),
                            child: Text(
                              L10n.of(context)!.accessoriesEmpty,
                              style: TextStyle(
                                color: Colors.grey[300],
                                fontStyle: FontStyle.italic,
                              ),
                            ),
                          ),
                  ),
                ],
              ),
            ),
          ),
          Container(
            margin: EdgeInsets.symmetric(vertical: 16),
            alignment: FractionalOffset.centerLeft,
            child: CircleAvatar(
              radius: 40,
              backgroundColor: Theme.of(context).primaryColor,
              child: IconButton(
                icon: Icon(widget.groups[i].icon()),
                iconSize: 40,
```

```
              color: Colors.white,
              onPressed: () {},
            ),
          ),
        ),
      ],
    ),
  );
 }

 @override
 Widget build(BuildContext context) {
  return Swiper(
   itemCount: widget.groups.length,
   scale: 0.9,
   viewportFraction: 0.95,
   itemBuilder: buildItem,
   pagination: SwiperPagination(
    alignment: Alignment.topCenter,
    margin: EdgeInsets.only(bottom: 40),
    builder: DotSwiperPaginationBuilder(
     color: Theme.of(context).primaryColorDark,
    ),
   ),
  );
 }
}

==> ./lib/pages/accessories/accessories.dart <==
import 'dart:convert';
import 'dart:io';

import 'package:flutter/material.dart' hide Route;
import 'package:shared_preferences/shared_preferences.dart';
import 'package:remotexpress/widgets/custom_dialog.dart';

import 'package:remotexpress/models/accessory.dart';
import 'package:remotexpress/models/group.dart';
import 'package:remotexpress/models/route.dart';

import 'package:remotexpress/pages/accessories/control.dart';
import 'package:remotexpress/pages/accessories/groups.dart';
import 'package:remotexpress/pages/accessories/routes.dart';

import 'package:remotexpress/net/station.dart';

class AccessoriesPage extends StatefulWidget {
 final Station station;

 AccessoriesPage(this.station);

 late _AccessoriesPageState _state;

 @override
 _AccessoriesPageState createState() {
  _state = _AccessoriesPageState(station);
  return _state;
 }

 void floatingButtonAction() {
  _state.onAddRoute();
 }
}

class _AccessoriesPageState extends State<AccessoriesPage> {
 final Station station;
 late SharedPreferences prefs;

 _AccessoriesPageState(this.station);

 static final defaultAccessories = List.generate(
  1024,
  (i) => Accessory(i + 1),
 );
 static final defaultGroups = [
  Group('Стрілки', Icons.compare_arrows.codePoint),
  Group('Світлофори', Icons.traffic.codePoint),
  Group('Освітлення', Icons.lightbulb.codePoint),
  Group('Шлагбауми', Icons.fence.codePoint),
 ];
 static final defaultRoutes = [
  Route(
   'Тестовий маршрут',
   [],
  ),
 ];

 List<Accessory> accessories = defaultAccessories;
 List<Group> groups = [];
 List<Route> routes = [];

 late String selectedGroup;
 late String selectedRoute;

 void setPrefs() {
  Future set(String key, Object object) async {
   final prefs = await SharedPreferences.getInstance();
   return await prefs.setString(key, jsonEncode(object));
  }

  Future.delayed(Duration.zero, () async {
   await set('groups', groups);
   await set('routes', routes);
  });
 }

 @override
 void initState() {
  Future.delayed(Duration.zero, () async {
   final prefs = await SharedPreferences.getInstance();

   Future setIfNotContains(String key, Object object) async {
    if (!prefs.containsKey(key)) {
     await prefs.setString(key, jsonEncode(object));
    }
   }

   List<T> decodeList<T>(String key, T Function(dynamic) decode) {
    return jsonDecode(prefs.getString(key)!).map<T>(decode).toList();
   }

   // First start
   await setIfNotContains('groups', defaultGroups);
   await setIfNotContains('routes', defaultRoutes);

   groups = decodeList(
    'groups',
    (json) => Group.fromJson(json),
   );
   routes = decodeList(
    'routes',
    (json) => Route.fromJson(json),
   );

   selectedGroup = groups[0].name;
   selectedRoute = routes[0].name;

   setState(() {});
  });

  super.initState();
 }

 void onToggle(int i) {
  final accessory = accessories[i];
  setState(() => accessory.toggle());
  station.updateAccessory(accessory);
 }

 void onAddToGroup(int i) {
  final accessory = accessories[i];

  CustomDialog.show(
   context,
   title: 'Виберіть групу',
   icon: Icons.add_to_photos,
   child: StatefulBuilder(
    builder: (builder, setState) => DropdownButton(
     value: selectedGroup,
     isExpanded: true,
     dropdownColor: Colors.white,
     items: groups
```

```
            .map<DropdownMenuItem<String>>(
              (group) => DropdownMenuItem(
                child: Text(group.name),
                value: group.name,
              ),
            )
            .toList(),
        onChanged: (group) {
          setState(() => selectedGroup = group.toString());
        },
      ),
    ),
    positiveText: 'Додати',
    negativeText: 'Видалити',
    onPositivePressed: () {
      setState(() {
        final group = groups.where((g) => g.name == selectedGroup).first;
        if (!group.accessories.contains(accessory)) {
          group.accessories.add(accessory);
          setPrefs();
        }
      });
      CustomDialog.pop(context);
    },
    onNegativePressed: () {
      setState(() {
        try {
        final group = groups.where((g) => g.name == selectedGroup).first;
        final a = group.accessories.where((a) => a.a == accessory.a).first;
        group.accessories.remove(a);
        setPrefs();
        } catch (e) {}
      });
      CustomDialog.pop(context);
    },
  );
}

void onAddToRoute(int i) {
  final accessory = accessories[i];

  CustomDialog.show(
    context,
    title: 'Виберіть маршрут',
    icon: Icons.alt_route,
    child: StatefulBuilder(
      builder: (builder, setState) => DropdownButton(
        value: selectedRoute,
        isExpanded: true,
        dropdownColor: Colors.white,
        items: routes
            .map<DropdownMenuItem<String>>(
              (route) => DropdownMenuItem(
                child: Text(route.name),
                value: route.name,
              ),
            )
            .toList(),
        onChanged: (route) {
          setState(() => selectedRoute = route.toString());
        },
      ),
    ),
    positiveText: 'Додати',
    negativeText: 'Видалити',
    onPositivePressed: () {
      setState(() {
        final route = routes.where((r) => r.name == selectedRoute).first;
        if (!route.turnouts.contains(accessory)) {
          route.turnouts.add(accessory);
          setPrefs();
        }
      });
      CustomDialog.pop(context);
    },
    onNegativePressed: () {
      setState(() {
        final route = routes.where((r) => r.name == selectedRoute).first;
        final turnout = route.turnouts.where((t) => t.a == accessory.a).first;
        route.turnouts.remove(turnout);
        setPrefs();
      });
      CustomDialog.pop(context);
    },
  );
}

void onPlayRoute(Route route) async {
  await Future.forEach(route.turnouts, (Accessory accessory) async {
    station.updateAccessory(accessory);
    setState(() => accessory.played = true);
    await Future.delayed(Duration(milliseconds: 500));
  });

  await Future.forEach(route.turnouts, (Accessory accessory) async {
    setState(() => accessory.played = false);
    await Future.delayed(Duration(milliseconds: 300));
  });
}

void onAddRoute() {
  final TextEditingController controller = TextEditingController();

  CustomDialog.show(
    context,
    title: 'Введіть назву маршруту',
    icon: Icons.alt_route,
    child: StatefulBuilder(
      builder: (builder, setState) => TextField(
        controller: controller,
        decoration: InputDecoration(hintText: 'Назва'),
        onChanged: (v) {
          setState(() {});
        },
      ),
    ),
    positiveText: 'Додати',
    negativeText: 'Відміна',
    onPositivePressed: () {
      setState(() => routes.add(Route(controller.value.text)));
      setPrefs();
      CustomDialog.pop(context);
    },
    onNegativePressed: () => CustomDialog.pop(context),
  );
}

void onDeleteRoute(Route route) {
  CustomDialog.error(
    context,
    title: 'Видалення маршруту',
    content: 'Ви впевнені, що хочете видалити ${route.name}?',
    positiveText: 'Видалити',
    negativeText: 'Відміна',
    onPositivePressed: () {
      setState(() => routes.remove(route));
      setPrefs();
      CustomDialog.pop(context);
    },
    onNegativePressed: () => CustomDialog.pop(context),
  );
}

@override
Widget build(BuildContext context) {
  return Padding(
    padding: EdgeInsets.symmetric(vertical: 20),
    child: Column(
      mainAxisAlignment: MainAxisAlignment.spaceBetween,
      children: [
        Padding(
          padding: EdgeInsets.symmetric(horizontal: 15),
          child: AccessoriesControl(
            accessories: accessories,
            onToggle: onToggle,
            onAddToGroup: onAddToGroup,
            onAddToRoute: onAddToRoute,
          ),
        ),
        Expanded(
          flex: 2,
          child: AccessoryGroups(
```

```
              groups: groups,
              onToggle: onToggle,
            ),
          ),
          SizedBox(height: 10),
          Expanded(
            flex: 3,
            child: AccessoryRoutes(
              routes: routes,
              onPlay: onPlayRoute,
              onDelete: onDeleteRoute,
            ),
          ),
        ],
      ),
    );
  }
}

==> ./lib/pages/debug <==

==> ./lib/pages/debug/debug.dart <==
import 'package:flutter/material.dart';
import 'package:google_fonts/google_fonts.dart';
import 'package:remotexpress/l10n.dart';
import 'package:remotexpress/widgets/logo.dart';

class DebugPage extends StatefulWidget {
  DebugPage({Key? key}) : super(key: key);

  @override
  _DebugPageState createState() => _DebugPageState();
}

class _DebugPageState extends State<DebugPage> {
  @override
  Widget build(BuildContext context) {
    return Align(
        alignment: Alignment.center,
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            Logo(),
            SizedBox(height: 20),
            Text(
              L10n.of(context)!.cvInProgress,
              style: GoogleFonts.lato(
                fontSize: 18,
                color: Colors.white,
              ),
            )
          ],
        ));

    // Column(
    //   mainAxisAlignment: MainAxisAlignment.start,
    //   children: [
    //     IntrinsicHeight(
    //       child: Row(
    //         mainAxisAlignment: MainAxisAlignment.spaceEvenly,
    //         crossAxisAlignment: CrossAxisAlignment.stretch,
    //         children: [
    //           Expanded(
    //             child: TextField(
    //               style: TextStyle(fontSize: 15),
    //               textAlign: TextAlign.center,
    //               decoration: InputDecoration(labelText: "CV"),
    //               keyboardType: TextInputType.number,
    //               inputFormatters: <TextInputFormatter>[
    //                 FilteringTextInputFormatter.digitsOnly,
    //                 RangeTextInputFormatter(min: 0, max: 1024),
    //               ],
    //             ),
    //           ),
    //           SizedBox(width: 10),
    //           Expanded(
    //             child: TextField(
    //               style: TextStyle(fontSize: 15),
    //               textAlign: TextAlign.center,
    //               decoration: InputDecoration(labelText: "Value"),
    //               keyboardType: TextInputType.number,
    //               inputFormatters: <TextInputFormatter>[
    //                 FilteringTextInputFormatter.digitsOnly,
    //                 RangeTextInputFormatter(min: 0, max: 1024),
    //               ],
    //             ),
    //           ),
    //           SizedBox(width: 10),
    //           Expanded(
    //             child: Padding(
    //               padding: EdgeInsets.only(top: 15, bottom: 0),
    //               child: ElevatedButton(
    //                 onPressed: () {},
    //                 child: Text("PROGRAM!"),
    //               ),
    //             ),
    //           ),
    //           ),
    //         ],
    //       ),
    //     ),
    //   ],
    // );
  }
}

==> ./lib/widgets <==

==> ./lib/widgets/toggle_button.dart <==
import 'package:flutter/material.dart';

class ToggleButton extends StatefulWidget {
  final bool on;
  final Widget child;
  final void Function()? onPressed;
  final void Function()? onLongPress;

  ToggleButton({
    required this.on,
    required this.child,
    this.onPressed,
    this.onLongPress,
  });

  @override
  _ToggleButtonState createState() => _ToggleButtonState();

  bool isDisabled() => onPressed == null;
}

class _ToggleButtonState extends State<ToggleButton> {
  @override
  Widget build(BuildContext context) {
    return OutlinedButton(
      child: widget.child,
      onPressed: widget.onPressed,
      onLongPress: widget.onLongPress,
      style: OutlinedButton.styleFrom(
        primary: widget.on ? Theme.of(context).primaryColor : Colors.grey[400],
        backgroundColor: Theme.of(context).primaryColorDark,
        onSurface: widget.on ? Theme.of(context).primaryColor : null,
        side: BorderSide.none,
        shape: StadiumBorder(),
        elevation: widget.on && !widget.isDisabled() ? 0.5 : 3,
      ),
    );
  }
}

==> ./lib/widgets/logo.dart <==
import 'package:flutter/material.dart';
import 'package:google_fonts/google_fonts.dart';

class Logo extends StatelessWidget {
  static Widget animated(void Function() onCompleted) {
    return _AnimatedLogo(onCompleted: onCompleted);
  }

  @override
  Widget build(BuildContext context) {
    return Column(
      mainAxisAlignment: MainAxisAlignment.center,
      children: [
```

```
        Container(
          padding: EdgeInsets.all(10),
          decoration: BoxDecoration(
            color: Colors.grey[200],
            borderRadius: BorderRadius.circular(20),
          ),
          child: Icon(
            Icons.train,
            size: 80,
            color: Theme.of(context).primaryColorDark,
          ),
        ),
        Padding(
          padding: EdgeInsets.only(top: 20.0),
        ),
        Text(
          "Xpress",
          style: GoogleFonts.lato(
            color: Colors.grey[200],
            fontWeight: FontWeight.bold,
            fontSize: 60,
          ),
        ),
      ],
    );
  }
}

class _AnimatedLogo extends StatefulWidget {
  void Function() onCompleted;

  _AnimatedLogo({required this.onCompleted});

  @override
  __AnimatedLogoState createState() => __AnimatedLogoState();
}

class __AnimatedLogoState extends State<_AnimatedLogo>
    with TickerProviderStateMixin {
  late AnimationController iconAnimation;

  @override
  void initState() {
    iconAnimation = AnimationController(
      vsync: this,
      duration: Duration(seconds: 2),
      lowerBound: 10,
      upperBound: 80,
    )..forward();

    iconAnimation.addStatusListener((status) {
      if (status == AnimationStatus.completed) {
        widget.onCompleted();
      }
    });

    super.initState();
  }

  @override
  void dispose() {
    iconAnimation.dispose();
    super.dispose();
  }

  @override
  Widget build(BuildContext context) {
    return AnimatedBuilder(
      animation: iconAnimation,
      builder: (context, child) {
        return Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            Container(
              padding: EdgeInsets.all(10),
              decoration: BoxDecoration(
                color: Colors.grey[200],
                borderRadius: BorderRadius.circular(20),
              ),
              child: Icon(
                Icons.train,
                size: iconAnimation.value,
                color: Theme.of(context).primaryColorDark,
              ),
            ),
            Padding(
              padding: EdgeInsets.only(top: 20),
            ),
            Text(
              "Xpress",
              style: GoogleFonts.lato(
                color: Colors.grey[200],
                fontWeight: FontWeight.bold,
                fontSize: iconAnimation.value - 20,
              ),
            ),
          ],
        );
      },
    );
  }
}

==> ./lib/widgets/animated_toggle.dart <==
import 'package:flutter/material.dart';

class AnimatedToggle extends StatefulWidget {
  final List<String> values;
  final ValueChanged onToggleCallback;
  final double width;
  final double heightScale;
  final Color backgroundColor;
  final Color buttonColor;
  final Color textColor;

  int index = 0;

  AnimatedToggle({
    required this.values,
    required this.onToggleCallback,
    this.index = 0,
    this.width = 250,
    this.heightScale = 1,
    this.backgroundColor = const Color(0xFFe7e7e8),
    this.buttonColor = const Color(0xFFFFFFFF),
    this.textColor = const Color(0xFF000000),
  }) {
    assert(values.length == 3);
  }

  @override
  _AnimatedToggleState createState() => _AnimatedToggleState();
}

class _AnimatedToggleState extends State<AnimatedToggle> {
  @override
  Widget build(BuildContext context) {
    final int length = widget.values.length - 1;
    final double width = widget.width;
    final double height = width * 0.13 * widget.heightScale;

    return Container(
      width: width * 0.7 * length,
      height: height,
      child: Stack(
        children: [
          Container(
            width: width * 0.7 * length,
            height: height,
            decoration: ShapeDecoration(
              color: widget.backgroundColor,
              shape: RoundedRectangleBorder(
                borderRadius: BorderRadius.circular(width * 0.1),
              ),
            ),
            child: Row(
              mainAxisAlignment: MainAxisAlignment.spaceBetween,
              children: List.generate(
                widget.values.length,
                (i) => MouseRegion(
                  cursor: SystemMouseCursors.click,
                  child: GestureDetector(
```

```
                  onTap: () {
                    widget.index = i;
                    widget.onToggleCallback(i);
                    setState(() {});
                  },
                  child: Container(
                    height: height,
                    alignment: Alignment.center,
                    color: Colors.transparent,
                    padding: EdgeInsets.symmetric(horizontal: width * 0.1),
                    child: Text(
                      widget.values[i],
                      style: TextStyle(
                        fontFamily: 'Rubik',
                        fontSize: width * 0.05,
                        fontWeight: FontWeight.bold,
                        color: const Color(0xFF918f95),
                      ),
                    ),
                  ),
                ),
              ),
            ),
          ),
          AnimatedAlign(
            duration: const Duration(milliseconds: 250),
            curve: Curves.decelerate,
            alignment: [
              Alignment.centerLeft,
              Alignment.center,
              Alignment.centerRight
            ][widget.index],
            child: Container(
              width: width * 0.35,
              height: height,
              decoration: ShapeDecoration(
                color: widget.buttonColor,
                shape: RoundedRectangleBorder(
                  borderRadius: BorderRadius.circular(width * 0.1),
                ),
              ),
              child: Text(
                widget.values[widget.index],
                style: TextStyle(
                  fontFamily: 'Rubik',
                  fontSize: width * 0.045,
                  color: widget.textColor,
                  fontWeight: FontWeight.bold,
                ),
              ),
              alignment: Alignment.center,
            ),
          ),
        ],
      ),
    );
  }
}
```

```
==> ./lib/widgets/custom_dialog.dart <==
import 'dart:io';

import 'package:flutter/material.dart';
import 'package:remotexpress/l10n.dart';

class CustomDialog extends StatefulWidget {
  final String title, positiveText, negativeText;
  final String? content;
  final Widget? child;
  final GestureTapCallback? onPositivePressed, onNegativePressed;
  final Color? circleColor;
  final Icon icon;

  CustomDialog({
    required this.title,
    required this.positiveText,
    required this.negativeText,
    this.content,
    this.child,
    this.onPositivePressed,
    this.onNegativePressed,
    this.circleColor,
    this.icon = const Icon(
      Icons.message,
      size: 35,
      color: Colors.white,
    ),
  });

  @override
  _CustomDialogState createState() => _CustomDialogState();

  static void show(
    BuildContext context, {
    required String title,
    required IconData icon,
    Widget? child,
    String? content,
    String? positiveText,
    String? negativeText,
    GestureTapCallback? onPositivePressed,
    GestureTapCallback? onNegativePressed,
  }) =>
      showGeneralDialog(
        context: context,
        barrierDismissible: true,
        barrierLabel: '',
        barrierColor: Colors.black54,
        transitionDuration: Duration(milliseconds: 400),
        transitionBuilder: (context, a1, a2, _) {
          return ScaleTransition(
            scale: CurvedAnimation(
              parent: a1,
              curve: Curves.elasticOut,
              reverseCurve: Curves.easeOutCubic,
            ),
            child: CustomDialog(
              icon: Icon(icon, size: 35, color: Colors.white),
              title: title,
              content: content,
              child: child,
              positiveText: positiveText ?? L10n.of(context)!.dialogPositive,
              negativeText: negativeText ?? L10n.of(context)!.dialogNegative,
              onPositivePressed: onPositivePressed ?? () => pop(context),
              onNegativePressed: onNegativePressed ?? () => exit(0),
            ),
          );
        },
        pageBuilder: (context, a1, a2) => SizedBox(),
      );

  static void error(
    BuildContext context, {
    required String title,
    required String content,
    String? positiveText,
    String? negativeText,
    GestureTapCallback? onPositivePressed,
    GestureTapCallback? onNegativePressed,
  }) =>
      show(
        context,
        icon: Icons.error,
        title: title,
        content: content,
        positiveText: positiveText,
        negativeText: negativeText,
        onPositivePressed: onPositivePressed,
        onNegativePressed: onNegativePressed,
      );

  static void pop(BuildContext context) {
    Navigator.of(context).pop();
  }
}

class _CustomDialogState extends State<CustomDialog> {
  @override
  Widget build(BuildContext context) {
    return Dialog(
      elevation: 0,
```

```
    backgroundColor: Colors.transparent,
    child: _buildDialogContent(context),
  );
}

Widget _buildButton(
  BuildContext context,
  String text,
  void Function()? onPressed,
) {
  return FlatButton(
    child: Text(text),
    textColor: Theme.of(context).primaryColorDark,
    onPressed: onPressed ?? () => CustomDialog.pop(context),
  );
}

Widget _buildDialogContent(BuildContext context) {
  final positiveButton = _buildButton(
    context,
    widget.positiveText,
    widget.onPositivePressed,
  );
  final negativeButton = _buildButton(
    context,
    widget.negativeText,
    widget.onNegativePressed,
  );

  return Stack(
    alignment: Alignment.topCenter,
    children: [
      Container(
        margin: EdgeInsets.only(top: 30),
        decoration: BoxDecoration(
          color: Colors.white,
          borderRadius: BorderRadius.circular(12),
        ),
        padding: EdgeInsets.only(top: 60, left: 20, right: 20),
        child: Column(
          mainAxisSize: MainAxisSize.min,
          children: [
            Text(
              widget.title,
              style: Theme.of(context).textTheme.headline5,
            ),
            SizedBox(height: 10),
            widget.content != null
                ? Text(
                    widget.content!,
                    style: Theme.of(context).textTheme.bodyText2,
                    textAlign: TextAlign.center,
                  )
                : widget.child!,
            SizedBox(height: 16),
            ButtonBar(
              buttonMinWidth: 100,
              alignment: MainAxisAlignment.spaceEvenly,
              children: widget.negativeText != ''
                  ? [negativeButton, positiveButton]
                  : [positiveButton],
            ),
          ],
        ),
      ),
      CircleAvatar(
        maxRadius: 40,
        child: widget.icon,
        backgroundColor: widget.circleColor,
      ),
    ],
  );
 }
}
```

Sun Jun 15 07:49:34 EEST 2025, Медзатий Дмитро Миколайович, Хмельницький національний університет, ХНУ

Anti-Plagiarism (UA) v-15.281 Educational

The maximum coincidence with one document 2.0%

Dictionaries check: en_US, ru_RU, ua_UA. **Errors in the documents: 13%**

	Document		Sum coincidence on the DB	
ID: 245903 Title: БКР Апаратно-програмний комплекс дистанційного керування залізничним макетом на мікроконтролерах з мультиплатформною програмною підтримкою Added in a DB: 2025-06-15 Authors: Дем`ян ГЕТЬМАН Heads: Дмитро МЕДЗАТИЙ Consultants: Opponents:	Symbols	Lexemes	Symbols	Lexemes
	71823	686	2337 (3%)	28 (4%)

Plagiarism sources

ID	Description	Plagiarism presence in the document	
		Symbols	Lexemes

Протокол аналізу звіту подібності експертом

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

Автор: Дем`ян ГЕТЬМАН

Співавтор:

Назва: Гетьман_Апаратно-програмний комплекс дистанційного керування залізничним макетом на мікроконтролерах з мультиплатформною програмною підтримкою

Експерт:

Підрозділ: Кафедра комп'ютерної інженерії та інформаційних систем

Коефіцієнт подібності 1:29.5%

Коефіцієнт подібності 2:25.2%

Мікропробіли: 11

Заміна букв: 7

Інтервали: 0

Білі знаки: 0

Дата створення звіту: 2025-06-15 06:51:43.0

Після аналізу Звіту подібності констатую наступне:

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

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

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

Обгрунтування:

2025-06-15 Доцент Андрій Нічепорук

Дата експерт

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

РЕЦЕНЗІЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

Дипломник: Гетьман Дем`ян Богданович

Тема: Апаратно-програмний комплекс дистанційного керування залізничним макетом на мікроконтролерах з мультиплатформною програмною підтримкою

Спеціальність: 123 «Комп'ютерна інженерія»

Обсяг кваліфікаційної роботи:

Кількість листів креслень ___4____ Кількість сторінок записки ___87___

1. Короткий зміст роботи та прийнятих рішень: Метою кваліфікаційної роботи є вибір апаратних засобів та проектування програмного застосунку з використанням середовища Flutter, що реалізує підтримку усіх популярних платформ для керування зазізничним макетом.

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

3. Характеристика виконання кожного розділу, ступінь використання останніх досягнень науки і техніки і передових методів роботи:

В першому розділі кваліфікаційної роботи проведено дослідження предметної області, а саме - аналіз сучасного стану розвитку залізничного моделювання в Україні та світі, аналіз ситуації в світових наукових колах, а також розглянуті наукові публікації і технічні дослідження, що стосуються предметної області.

В другому розділі кваліфікаційної роботи виконана апаратна реалізація комплексу, розглянуто стандарт DCC та розглянуті питання використання мікроконтролерів для даного проекту.

В третьому розділі кваліфікаційної роботи виконана програмна реалізація комплексу, а саме - проєктування архітектури, структур даних, інтерфейсу ПЗ, розробка та опис програмної реалізації та тестування задачі.

4. Позитивні сторони роботи: висока практична цінність роботи.

5. Негативні сторони роботи: доцільно було розглянути інші стандарти цифрового керування макетами окрім стандарту DCC.

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

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

8. Інші зауваження: _____

9. Оцінка дипломної роботи: добре

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

к.т.н., доцент кафедри автоматизації, комп'ютерно-інтегрованих технологій та робототехніки Федуло Микола Васильович

"___"_____2025 р. _____(підпис)

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

Дем`яна ГЕТЬМАНА

ПІБ здобувача вищої освіти

ФІТ, 3 курсу, групи КІ2с-22-1

ЗАЯВА

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

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

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

_____ 2025 року

РІШЕННЯ ЕКСПЕРТНОЇ КОМІСІЇ

КАФЕДРИ КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ ТА ІНФОРМАЦІЙНИХ СИСТЕМ

ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

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

Назва: Апаратно-програмний комплекс дистанційного керування залізничним макетом на мікроконтролерах з мультиплатформною програмною підтримкою

Автор: Дем`ян ГЕТЬМАН

Спеціальність: 123– Комп'ютерна інженерія

Освітня програма: освітньо-професійна

Науковий керівник: Дмитро МЕДЗАТИЙ, к.т.н, доцент

Після аналізу звіту подібності зроблено такий висновок:

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

Підтвердження:

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

1) запозичення розміщені в розділах аналізу існуючих аналогів та прототипів, які не описують безпосередньо авторське дослідження і не стосуються результатів роботи;
2) усі запозичення фрагментарні, або мають належним чином оформленні посилання;
3) окремі виявлені збіги є загальновживаними фразами або виразами, про що свідчить посилання системи на збіг з 10-40 джерелами на один фрагмент речення;
4) в якості запозичень в окремих місцях системою зафіксовано послідовності програмних інструкцій, які є вхідними даними до великої кількості задач і не можуть розглядатися як об'єкт авторських прав і, відповідно, їх порушення;

Сумарний обсяг всіх запозичень, визначений системою виявлення збігів/ідентичності/схожості StrikePlagiarism, складає 29.51% і адресується до 27 першоджерел; та системою Anti-Plagiarism складає 25,2%, що, з урахуванням наведених обґрунтувань, відповідає характеру наукового дослідження і свідчить на користь кваліфікаційної роботи.

Керівник роботи _____ Дмитро МЕДЗАТИЙ

Гарант ОП _____ Андрій Нічепорук

Завідувач кафедри КІІС _____ Ольга ПАВЛОВА