НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»
Факультет інформатики та обчислювальної техніки
(повна назва інституту/факультету)

Кафедра інформатики та програмної інженерії
(повна назва кафедри)

«До захисту допущено»
Завідувач кафедри
_____ Едуард ЖАРІКОВ
(підпис) (ім'я прізвище)
"___"_____2023 р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Інженерія програмного забезпечення комп'ютерних систем»

спеціальності «121 Інженерія програмного забезпечення»

на тему: Програмне забезпечення для підтримки діяльності паяльної станції

Виконав студент IV курсу, групи ІТ-81в
(шифр групи)
Власюк Владислав Володимирович _____
(прізвище, ім'я, по батькові) (підпис)

Керівник ст. викл. Халус О.А. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант з графічної документації ст. викл., Головченко М.М. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Рецензент доцент кафедри ІСТ, к.т.н., доц., Шимкович В. М. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ –2023

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 121 Інженерія програмного забезпечення

Освітньо-професійна програма – Інженерія програмного забезпечення комп'ютерних систем

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Едуард ЖАРІКОВ
(підпис) (ім'я прізвище)
"___"_____2023 р.

ЗАВДАННЯ
на дипломний проєкт студенту

Власюку Владиславу Володимировичу
(прізвище, ім'я, по батькові)

1. Тема проєкту Програмне забезпечення для підтримки діяльності паяльної станції

керівник проєкту Халус Олена Андріївна, ст. викл. кафедри ІПІ
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «06» червня 2023 р. №2196с

2. Термін подання студентом проєкту « 16 » червня 2023 року

3. Вихідні дані до проєкту: технічне завдання

4. Зміст пояснювальної записки
1) Аналіз вимог до програмного забезпечення: загальні положення, змістовний опис і аналіз предметної області, аналіз існуючих технологій та успішних ІТ-проєктів, аналіз вимог до програмного забезпечення, постановка задачі.
2) Моделювання та конструювання програмного забезпечення: моделювання та аналіз програмного забезпечення, архітектура програмного забезпечення, конструювання програмного забезпечення.
3) Аналіз якості та тестування програмного забезпечення: аналіз якості ПЗ, опис процесів тестування, опис контрольного прикладу.
4) Впровадження та супровід програмного забезпечення: розгортання програмного забезпечення, підтримка програмного забезпечення.

5. Перелік графічного матеріалу

1) Схема структурна варіантів використань _____

2) Схема структурна класів програмного забезпечення _____

3) Схема бази даних _____

4) Креслення вигляду екранних форм _____

6. Консультанти розділів проєкту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання «10» березня 2023 року _____

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1	Вивчення рекомендованої літератури	10.03.2023	
2	Аналіз існуючих методів розв’язання задачі	14.03.2023	
3	Постановка та формалізація задачі	20.03.2023	
4	Розробка інформаційного забезпечення	25.03.2023	
5	Алгоритмізація задачі	30.03.2023	
6	Обґрунтування вибору використаних технічних засобів	05.04.2023	
7	Розробка програмного забезпечення	11.05.2023	
8	Налагодження програми	25.05.2023	
9	Виконання графічних документів	29.05.2023	
10	Оформлення пояснювальної записки	02.06.2023	
11	Подання ДП на попередній захист	05.06.2023	
12	Подання ДП рецензенту	12.06.2023	
13	Подання ДП на основний захист	19.06.2023	

Студент _____ Владислав ВЛАСЮК

(підпис) (ініціали, прізвище)

Керівник _____ Олена ХАЛУС

(підпис) (ініціали, прізвище)

АНОТАЦІЯ

Пояснювальна записка дипломного проєкту складається з чотирьох розділів, містить 49 таблиць, 31 рисунок та 13 джерел – загалом 78 сторінок.

Дипломний проєкт присвячений розробці програмного забезпечення для підтримки діяльності паяльної станції.

Метою проєкту є забезпечення зручної роботи з паяльною станцією в автоматичному режимі завдяки розробці універсального та покращеного програмного забезпечення.

Об'єкт дослідження: програмне забезпечення для паяльної станції

Предмет дослідження: процеси моделювання, розроблення, модифікації, аналізу, забезпечення якості, впровадження і супроводження програмного забезпечення для паяльної станції.

У першому розділі проаналізовано предметну область існуючі рішення програмного забезпечення для паяльних станцій. За результатами аналізу було розроблено варіанти використання, функціональні та нефункціональні вимоги до застосунку та зроблено постановку задачі.

Розділ моделювання та конструювання програмного забезпечення присвячений моделюванню бізнес-процесів застосунку, побудові архітектури та конструюванню програмного забезпечення. В ньому описано як архітектуру всієї системи в цілому, так і архітектуру кожного компонента окремо.

У третьому розділі було проаналізовано якість програмного забезпечення, розроблено тест-кейси для тестування на основі функціональних вимог з першого розділу, виконано тестування додатка та проаналізовано результати тестування.

У останньому розділі було розглянуто та реалізовано методи розгортання та підтримки програмного забезпечення.

КЛЮЧОВІ СЛОВА: ДЕСКТОПНИЙ ДОДАТОК, КОНТРОЛЬ ТЕМПЕРАТУРИ, ПАЯЛЬНА СТАНЦІЯ, AVALONIA, SQLITE, DOTNET, БАГАТОШАРОВА АРХІТЕКТУРА

ABSTRACT

The explanatory note of the diploma project consists of four sections, contains 49 tables, 31 figures and 13 sources – in total 77 pages.

The diploma project is dedicated to the development of software to support the operation of a soldering station.

The aim of the project is to ensure convenient operation with a soldering station in automatic mode through the development of universal and enhanced software.

The object of study: software for the soldering station.

Research subject: processes of modeling, development, modification, analysis, quality assurance, implementation, and maintenance of software in the context of the field.

In the first chapter, the subject area and existing software solutions for soldering stations are analyzed. Based on the analysis results, application scenarios, functional and non-functional requirements for the application are developed and the problem definition is formulated.

The modeling and software construction chapter focuses on modeling the application's business processes, building the architecture and constructing the software. It describes both the architecture of the overall system and the architecture of the individual components.

In the third chapter, the quality of the software is analyzed. Based on the functional requirements from the first chapter, test cases were developed, and tests of the application were performed. The results of the tests were evaluated.

In the last chapter, the methods of software deployment and support were analyzed and implemented.

KEYWORDS: DESKTOP APPLICATION, TEMPERATURE CONTROL, SOLDERING STATION, AVALONIA, SQLITE, DOTNET, LAYERED ARCHITECTURE

№ з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
1	А4		Завдання на дипломний проєкт	2	
2	А4	КПІ.ІТ-8105в.045490.01.91	Технічне завдання	12	
3	А4	КПІ.ІТ-8105в.045490.02.81	Пояснювальна записка	77	
4	А4	КПІ.ІТ-8105в.045490.03.12	Текст програми	57	
5	А4	КПІ.ІТ-8105в.045490.04.51	Програма та методика тестування	5	
6	А4	КПІ.ІТ-8105в.045490.05.34	Керівництво користувача	10	
7	А3	КПІ.ІТ-8105в.045490.06.99	Графічний матеріал	4	

Змін.	Арк.	№ докум.	Підп.	Дата	КПІ.ІТ-8105в.045490.00.90		
Розроб.		Власюк В.В.			Відомість дипломного проєкту	Літ.	Аркуш 1 / Аркушів
Перевір.		Халус О.А.					
Н.контр.		Головченко М.М				КПІ ім. Ігоря Сікорського ФІОТ каф. ІПІ гр. ІТ-81в	
Затв.		Жаріков Е.В.					

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2023 р.

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ ПІДТРИМКИ ДІЯЛЬНОСТІ ПАЯЛЬНОЇ СТАНЦІЇ

Технічне завдання

КПІ.ІТ-8105в.045490.01.91

“ПОГОДЖЕНО”
Керівник проєкту:
_____ Олена ХАЛУС

Нормоконтроль:
_____ Максим ГОЛОВЧЕНКО

Виконавець:
_____ Владислав Власюк

Київ – 2023

ЗМІСТ

1 НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ

Назва розробки: Програмне забезпечення для підтримки діяльності паяльної станції.

Галузь застосування:

Наведене технічне завдання поширюється на розробку програмного забезпечення для підтримки діяльності паяльної станції, котре використовується для регулювання температури за температурними профілями та призначена для інженерів з ремонту комп’ютерної та іншої техніки.

2 ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки програмного забезпечення для підтримки діяльності паяльної станції є завдання на дипломне проєктування, затверджене кафедрою інформатики та програмної інженерії Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського».

3 ПРИЗНАЧЕННЯ РОЗРОБКИ

Розробка призначена для управління різними контролерами температури паяльної станції в ручному та автоматичному режимі, що забезпечує рівномірний та правильний нагрів материнської плати, яка обслуговується інженерами з ремонту техніки.

Метою розробки є забезпечення зручної роботи з паяльною станцією в автоматичному режимі через універсальне та покращене програмне забезпечення.

4 ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Вимоги до функціональних характеристик

Програмне забезпечення повинно забезпечувати виконання наступних основних функції:

4.1.1 Користувацького інтерфейсу:

– інтерфейс повинен бути адаптивним, з розширенням вікна має розширюватися графік;

– можливість зміни мови, має бути принаймні підтримка англійської та української мови;

– інтерфейс повинен бути зручним з точки зору користувача, не перевантаженим зайвими елементами;

– графічний інтерфейс повинен відповідати прототипу, наведеному на рисунку 4.1.

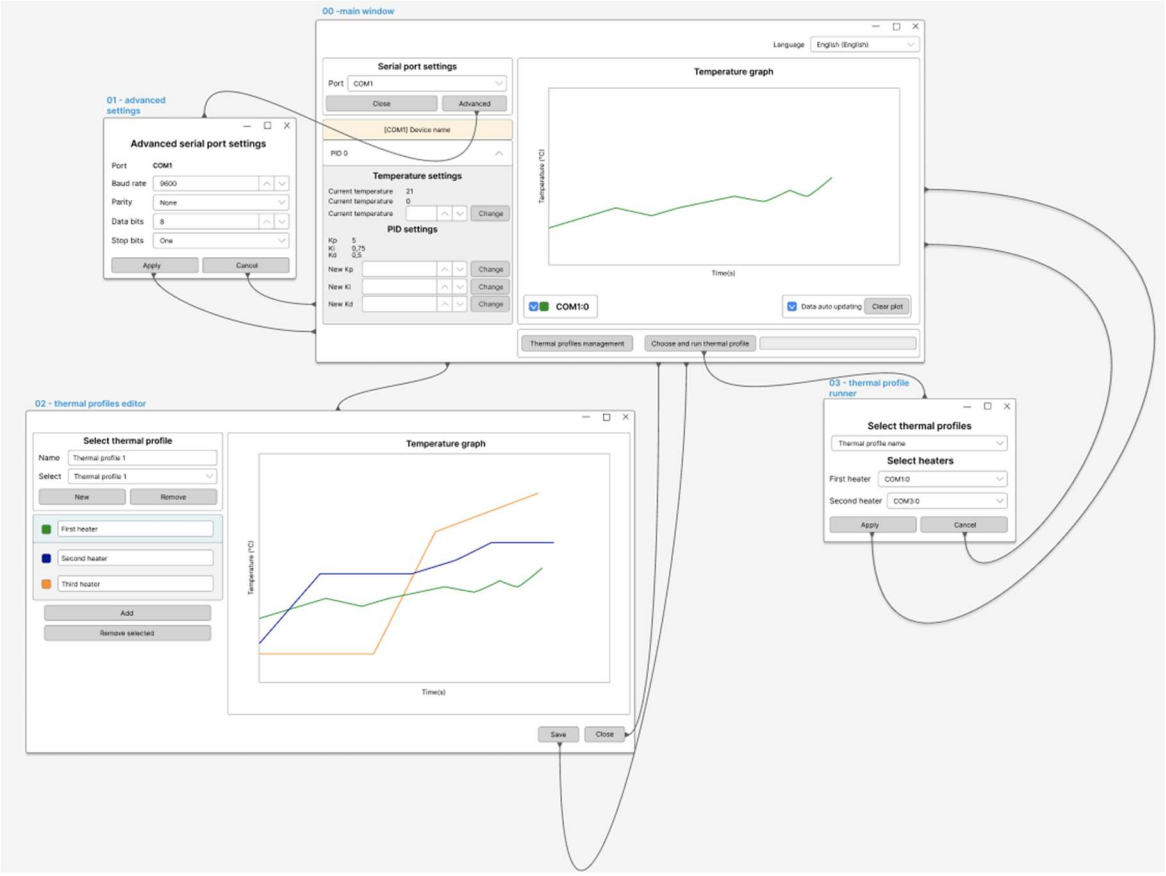


Рисунок 4.1 – Прототип інтерфейсу

4.1.2 Для користувача:

- відображення усіх послідовних портів;
- підключення/відключення пристрою;
- зміна налаштувань послідовного порту;
- відображення усіх підключених контролерів;
- відображення графіку температур;
- можливість змінити температуру;
- можливість змінити коефіцієнти контролера температури;
- можливість очистити графік температур;
- можливість змінити колір кривої на графіку;
- можливість приховати контролер на графіку;
- створення, редагування та видалення температурних профілів;
- запуск роботи за температурним профілем;
- відображення процесу роботи за температурним профілем.

4.2 Вимоги до надійності

Передбачити контроль введення інформації та захист від некоректних дій користувача. Забезпечити цілісність інформації в базі даних. Забезпечити цілісність інформації в енергонезалежній пам’яті мікроконтролера.

4.3 Умови експлуатації

Умови експлуатації згідно ДСанПІН 3.3.2.007-98.

4.3.1 Вид обслуговування

Вимоги до виду обслуговування не висуваються.

4.3.2 Обслуговуючий персонал

Вимоги до обслуговуючого персоналу не висуваються.

4.4 Вимоги до складу і параметрів технічних засобів

Програмне забезпечення повинно функціонувати на IBM-сумісних персональних комп‘ютерах.

Мінімальна конфігурація технічних засобів:

– тип процесору: Intel Pentium 4/Athlon 64 або пізнішої версії з підтримкою SSE2;
– об‘єм ОЗП: 4 Гб;
– наявна інтегрована або дискретна відеокарта;
– монітор з роздільною здатністю не менше 1280x720 (HD);
– вільне місце на диску – 300 Мб

Рекомендована конфігурація технічних засобів:

– тип процесору: Intel Core i7;
– об‘єм ОЗП: 8 Гб;
– дискретна відеокарта;
– монітор з роздільною здатністю 1920x1080 (FullHD);
– вільне місце на диску – 1 Гб

4.5 Вимоги до інформаційної та програмної сумісності

Програмне забезпечення повинно працювати під управлінням операційних систем Windows 10-11, Ubuntu 16.04 і вище, Debian 9 і вище.

4.5.1 Вимоги до вхідних даних

Вхідні дані повинні бути представлені в наступному форматі: числовий формат

4.5.2 Вимоги до вихідних даних

Результати повинні бути представлені в наступному форматі: числовий формат, графік.

4.5.3 Вимоги до мови розробки

Розробку клієнта виконати на мові програмування C#, розробку вбудованого ПЗ виконати на мові програмування C.

4.5.4 Вимоги до середовища розробки

Розробку клієнтського додатку виконати на платформі за допомогою інтегрованого середовища розробки (IDE) Rider, розробку програмного забезпечення для контролера виконати в середовищі Visual Studio Code з плагіном PlatformIO.

4.5.5 Вимоги до представленню вихідних кодів

Вихідний код програми має бути представлений у вигляді текстових файлів, структурованих у відповідності до каталогової структури. Кожен файл повинен містити весь необхідний код для певного класу або модуля.

4.6 Вимоги до маркування та пакування

Вимоги до маркування та пакування не висуваються.

4.7 Вимоги до транспортування та зберігання

Вимоги до транспортування та зберігання не висуваються.

4.8 Спеціальні вимоги

Згенерувати інсталяційну версію клієнтського програмного забезпечення.

5 ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ

5.1 Попередній склад програмної документації

У склад супроводжувальної документації повинні входити наступні документи на аркушах формату А4:

- пояснювальна записка;
- технічне завдання;
- текст програми;
- програма та методика тестування;
- керівництво користувача;

Графічна частина повинна бути виконана на аркушах формату А3 та містити наступні документи:

- схема структурна варіантів використання;
- схема структурна класів програмного забезпечення;
- схема бази даних;
- креслення вигляду екранних форм;

5.2 Спеціальні вимоги до програмної документації

Програмні модулі, котрі розробляються, повинні бути задокументовані, тобто тексти програм повинні містити всі необхідні коментарі.

6 СТАДІЇ І ЕТАПИ РОЗРОБКИ

Стадії та етапи розробки застосування наведено у таблиці 6.1.

Таблиця 6.1 – Стадії та етапи розробки

№	Назва етапу	Строк	Звітність
1.	Вивчення рекомендованої літератури	10.03	
2.	Аналіз існуючих методів розв’язання задачі	14.03	Порівняльна таблиця
3.	Постановка та формалізація задачі	10.03	Схема структурна варіантів використання, матриця трасування вимог
4.	Розробка інформаційного забезпечення	25.03	
5.	Алгоритмізація задачі	30.03	Схема структурна програмного забезпечення та специфікація компонентів (діаграма класів)
6.	Обґрунтування вибору використаних технічних засобів	05.04	
7.	Розробка програмного забезпечення	11.05	Тексти програмного забезпечення
8.	Налагодження програми	25.05	Тести, результати тестування
9.	Виконання графічних документів	29.05	Графічний матеріал проекту
10.	Оформлення пояснювальної записки	02.06	Пояснювальна записка

7 ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ

Тестування розробленого програмного продукту виконується відповідно до “Програми та методики тестування”.

Пояснювальна записка
до дипломного проєкту

на тему: Програмне забезпечення для підтримки діяльності паяльної станції

КПІ.ІТ-8105в.045490.02.81

Київ-2023

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

IDE	–	Integrated Development Environment – інтегроване середовище розробки
ПЗ	–	програмне забезпечення
SDK	–	Software development kit
IT	–	Інформаційні технології
ER	–	Entity-Relation diagram
ОС	–	Операційна система
БД	–	База даних
ПК	–	Персональний комп’ютер
ПІД	–	Пропорціонально-інтегрально-диференціальний (закон)

ВСТУП

В наш час майже кожна людина має персональний комп’ютер, ноутбук, планшет або телефон. Інколи трапляється так, що техніка перестає працювати, або починає працювати некоректно. Одні люди одразу викидають та купують нову, а інші – йдуть в найближчий сервісний центр її ремонтувати.

Для ремонту сучасної техніки дуже часто інженеру з ремонту необхідне високотехнологічне обладнання, оскільки замінити, чи навіть просто демонтувати інтегральні мікросхеми в корпусі BGA без нього неможливо. В перелік цього обладнання завжди входить паяльна станція, яка забезпечує плавний нагрів материнської плати до заданої температури. Частіше за все для цього використовується термічний профіль для того чи іншого виду припою, товщини плати та інших фізичних характеристик.

Українські сервісні центри та приватні підприємці частіше за все купують дороге іноземне обладнання, конструюють свої напівавтоматичні паяльні станції (з якими не завжди зручно працювати), або повністю відмовляються від ремонту. Якщо розглянути вже представлені на ринку варіанти, то можна помітити, що якісних рішень, які задовольняють усі вимоги, не так багато, і всі вони є важкодоступними з економічної або логістичної точки зору.

Створення системи з контролера управління та клієнтського програмного забезпечення для паяльної станції дадуть можливість українським сервісним центрам виконувати складні ремонти, не витрачаючи величезні кошти на паяльне обладнання. За допомогою клієнтського програмного забезпечення можна буде підключити декілька блоків управління, налаштувати будь-який нагрівач, підключений до них, створити чітко визначений температурний профіль, а також запустити процес виконання цього профілю.

1 АНАЛІЗ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1 Загальні положення

У сучасному світі, коли електроніка проникає у абсолютно всі сфери нашого життя, виникає потреба у обладнанні для створення, модифікації та ремонту електронних пристроїв. З кожним роком техніка стає все складнішою, отже виникає і потреба в більш якісних інструментах. Одним з типів таких інструментів є паяльне обладнання. Якщо ще у 1900х роках можна було обійтися звичайнісіньким паяльником, то зараз вже майже неможливо щось зробити без термоповітряної або професійної інфрачервоної паяльної станції.

Паяльна станція є спеціалізованим інструментом, призначеним для нагрівання друкованої плати та компонентів на ній до температури плавлення припою. Таким чином відбувається запаювання як пасивних компонентів (резисторів, конденсаторів, індуктивностей, запобіжників), так і багатовивідних інтегральних схем та роз’ємів.

Для досягнення якісного монтажу електронних компонентів нерідко використовується професійна інфрачервона паяльна станція, що дозволяє фахівцям працювати відповідно до встановленого технологічного процесу. Інфрачервона технологія забезпечує точне та контрольоване нагрівання плати та компонентів на ній. Цей тип паяльних станцій дозволяє регулювати температуру та інші параметри паяння, що важливо для виконання роботи відповідно до вимог технології та специфікацій компонентів. Температурний профіль[1] дозволяє точно та рівномірно досягти необхідної температури, щоб забезпечити надійне та якісне з'єднання компонентів з друкованою платою. Дотримання технології та специфікацій дозволяє підтримувати високу якість паяння, забезпечувати надійність електричних з'єднань та уникнути можливих пошкоджень компонентів внаслідок нерівномірного або занадто швидкого нагріву.

1.2 Змістовний опис і аналіз предметної області

Інфрачервона паяльна станція зазвичай складається з декількох інфрачервоних нагрівачів (частіше за все їх 2) та їх контролерів. Деякі виробники об’єднують контролери в один, який управляє всіма нагрівачами. Дешеві паяльні станції зазвичай мають автономні контролери, які дозволяють встановити лише статичну температуру та не містять ніяких додаткових налаштувань. Більш професійні варіанти мають інтерфейс для підключення до персонального комп’ютера та спеціальне клієнтське програмне забезпечення для налаштування та управління процесом пайки. Проблему роботи за певним температурним профілем вирішують також різними способами. Перший спосіб – запис підготовленого температурного профілю в пам’ять контролера у вигляді списку пар значень часової мітки та температури для кожного з нагрівачів (якщо їх більше одного). Другий спосіб полягає в прямому управлінні параметрами контролера паяльної станції в реальному часі, що потребує постійного підключення до ПК. Кожен з цих способів має свої переваги та недоліки. Так, перевагою першого способу є можливість роботи без ПК, а недоліком – відсутність гнучкості, відсутність можливості нормально проаналізувати дані, обмежений вивід даних через можливості вбудованої системи. Другий спосіб не має таких недоліків, проте потребує постійного стабільного з’єднання з ПК та передачі даних в режимі реального часу.

Проблемою усіх наявних рішень є те, що клієнтське програмне забезпечення є сильно прив’язаним до конкретної паяльної станції, з якою вони використовуються, користувацький інтерфейс є застарілим, незручним, дуже часто лише китайською мовою.

З метою вирішення існуючих проблем, було прийнято рішення розробити власне програмне забезпечення. Клієнтський додаток повинен забезпечувати підтримку різних контролерів температури, які, в свою чергу, можуть забезпечувати керування кількома нагрівачами (тобто, фактично,

функціонувати в багатоканальному режимі). Керування має відбуватися в реальному часі, оскільки це дає змогу працювати як з власноруч розробленим контролером для паяльної станції, так і з іншими комерційними контролерами загального призначення для управління нагрівом. Архітектура клієнтського додатку повинна бути побудована таким чином, щоб було можливо інтегрувати будь-який контролер та підключити його не лише за допомогою послідовного порту, а й за допомогою інших інтерфейсів в майбутньому. Для забезпечення високої стабільності та зручності роботи паяльної станції, було вирішено розробити програмне забезпечення для контролера, яке до того ж буде гнучким, тобто його можна буде застосувати для різних конфігурацій паяльних станцій в залежності від потреб сервісного центру.

1.3 Аналіз існуючих технологій та успішних ІТ-проєктів

Проаналізуємо відоме на сьогоднішній день алгоритмічне забезпечення в даній предметній області та технічні рішення, що допоможуть у реалізації «Програмного забезпечення для підтримки роботи паяльної станції». Далі будуть розглянуті допоміжні програмні засоби, засоби розробки та вже повністю готові програмні рішення.

1.3.1 Аналіз відомих алгоритмічних та технічних рішень

Для вирішення задачі підтримки та зміни температури нагрівачів майже завжди використовується пропорціонально-інтегрально-диференціальний (ПІД) закон регулювання. ПІД регулятор[2] відноситься до регуляторів з оберненим зв’язком, основною перевагою є те, що вони можуть адаптуватися до змін у системі, оскільки вони отримують зворотний зв'язок про стан системи. Це дозволяє системі ефективно реагувати на зміни параметрів та умов роботи. Найчастіше вони застосовуються в системах опалення, кондиціонування повітря, котлах, печах та інших системах теплопостачання для досягнення стабільної температури і оптимального керування тепловими

процесами, що і необхідно в рамках предметної області. На рисунку 1.1 зображено блок-схему ПІД регулятора зі зворотнім зв’язком.

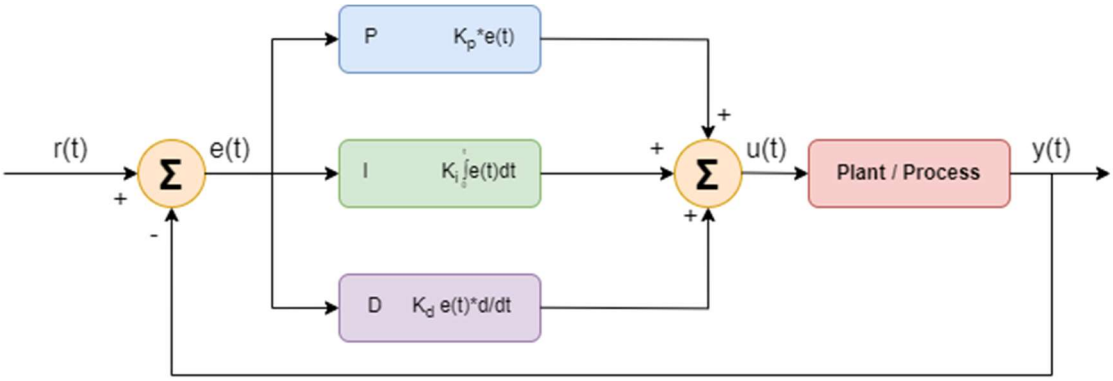


Рисунок 1.1 – Блок-схема ПІД регулятора зі зворотнім зв'язком

Такі регулятори значно поліпшують якість регулювання, особливо якщо об’єкт володіє великим запізненням та інерційністю, тому для реалізації ПЗ контролера станції буде використовуватися саме цей алгоритм.

1.3.2 Аналіз допоміжних програмних засобів та засобів розробки

Для розробки клієнту програмного забезпечення можна використовувати будь-яку кросплатформову високорівневу мову програмування. Найбільш популярними такими мовами є Python, Java, C#[3]. Мови програмування Java та C# мають строгу статичну типізацію, в той час як Python має динамічну типізацію. Статична типізація дозволяє уникнути великої кількості помилок під час виконання програми, оскільки помилки, такі як невідповідність типів або недопустимі операції, виявляються ще на етапі компіляції. Покращена швидкодія програми є ще однією перевагою статичної типізації, оскільки компілятор має повну інформацію про типи даних і може здійснювати оптимізацію коду. Мова програмування C# є однією з ключових мов, які активно розвиваються разом з платформою .NET. Ця мова програмування була розроблена компанією Microsoft і вперше випущена у 2000 році, однак з того часу вона пройшла значні зміни і вдосконалення[4], платформа стала відкритою. Крім того, мова C# має велику підтримку з боку

розробників, активну спільноту та багато інструментів, що полегшують розробку та підтримку проєктів. Ця екосистема надає доступ до різноманітних бібліотек та фреймворків, розширюють можливості мови C# і допомагають забезпечити високу якість і швидкість розробки.

Для розробки графічного інтерфейсу можна використовувати як технології запропоновані Microsoft, так і сторонні бібліотеки. До технологій Microsoft належать Windows Forms (WinForms), Windows Presentation Foundation (WPF), MAUI (.NET Multi-platform App UI). До сторонньої технології можна віднести бібліотеку з відкритим вихідним кодом AvaloniaUI[5]. Windows Forms на даний момент не є оптимальним вибором, оскільки, у порівнянні з сучасними рішеннями, є досить застарілою технологією. Її функціональність обмежена лише роботою в середовищі Windows, і не надає можливості мультиплатформного використання. WPF є досить сучасною технологією, проте також може працювати лише в середовищі Windows. Отож, для вибору залишаються лише AvaloniaUI та MAUI. Обидва фреймворки підтримують різні платформи та є досить молодими, однак AvaloniaUI має більшу спільноту розробників та є більш орієнтованою на десктопні додатки, аніж на додатки для мобільних пристроїв. Формат розмітки для опису графічного інтерфейсу користувача дуже схожий на формат, який використовується в WPF, тому вивчення технології займе набагато менше часу. Отож, для створення графічного інтерфейсу, який буде добре працювати на різних платформах, найкраще підійде фреймворк AvaloniaUI.

Для розробки на платформі .NET використовують наступні IDE:

– Visual Studio;
– Visual Studio Code;
– Rider.

Visual Studio – це інтегроване середовище розробки (IDE) від відомої на весь світ компанії Microsoft, яке надає широкий набір інструментів для

написання, налагодження та тестування коду. Це один з найбільш популярних інструментів для розробки програмного забезпечення на платформі Windows станом на 2023 рік. Ліцензія версії «Community» є безкоштовною, що дозволяє навчатися та створювати некомерційні додатки без додаткових витрат. Останні версії середовища включають також в себе технологію доповнення коду «IntelliSense» на базі штучного інтелекту. IDE підтримує різні мови програмування, такі як C#, C++, Visual Basic, Python, JavaScript і багато інших. Він також має можливості для створення графічних інтерфейсів користувача, роботи з базами даних, розробки мобільних додатків та інше. Одним з найбільших недоліків даного середовища є швидкість роботи. Ще одним недоліком є відсутність підтримки операційних систем на базі ядра Linux.

Rider[6] – спеціалізована IDE для платформи .NET створена компанією JetBrains. Rider надає потужні інструменти та функції для розробки, призначені для програмування на мовах C#, VB.NET, F#, а також роботи з ASP.NET, .NET Core та іншими технологіями .NET. Однією з його сильних сторін є потужний статичний аналізатор коду, який дозволяє уникати не тільки синтаксичних помилок коду, але і логічних. Також він має вбудовані інструменти для інтеграції з різними сервісами, базами даних, має розвинену систему плагінів. Цей інструмент є платним, однак він надає безкоштовну ліцензію для студентів. В цілому, він є зручнішим інструментом, дозволяє зекономити час на написання та налагоджування коду, тому для розробки було обрано саме його.

Для розробки контролера паяльної станції зазвичай необхідно завантажувати SDK (Software development kit) або IDE від виробника, однак існує інструмент, який дозволяє зекономити час та полегшити розробку. PlatformIO є універсальним інструментом, створеним українськими розробниками, для розробки програмного забезпечення для вбудованих систем. Він є крос-платформним, крос-архітектурним та багатошаровим, підтримує різні мікроконтролери та платформи, такі як Arduino, ESP32,

STM32, Raspberry Pi та багато інших, що дозволяє розробникам працювати з різноманітними апаратними платформами одночасно. Він дозволяє використовувати одну і ту ж кодову базу, щоб виконати збірку для різних апаратних платформ. Крім того, PlatformIO забезпечує зручні інструменти для збірки коду, керування залежностями, налагодження та розгортання проектів. Він розповсюджується як плагін для VSCode, тому наявність цього текстового редактору є обов'язковою умовою для його роботи. Завдяки своїй універсальності PlatformIO став популярним інструментом і знайшов широке застосування у проектах, що вимагають якісної розробки та налагодження програмного забезпечення для різних платформ. Прикладами таких проектів є програмне забезпечення для 3D принтерів «Marlin firmware», ПЗ для автопілотних систем «ArduPilot», а також ПЗ для створення IoT пристроїв «ESPHome». Отож, він зарекомендував себе як якісний інструмент, тому для розробки ПЗ для контролера було вирішено використати саме його.

1.3.3 Аналіз відомих програмних продуктів

На ринку є багато паяльних станцій, які мають досить обмежений функціонал, тому їх розглядати не будемо, проаналізуємо лише найкращі рішення, на які звертають увагу професійні інженери з ремонту техніки. Найбільш функціональними є такі рішення:

- ACHI IR6500;
- Quick 7710;
- Термопро ИК-650.

Інфрачервона паяльна станція ACHI IR-6500 - досить потужний пристрій, який може застосовуватися для ремонту ПК, материнських і серверних плат, промислових комп'ютерів, телефонів та інших пристроїв. Як для пристрою такого класу, ціна є доступною (близько 30 тисяч гривень). Станція застосовується для безсвинцевої і свинцевої пайки, процес демонтажу займає близько 5-ти хвилин. ACHI IR-6500 може підключатися до ПК або

ноутбука через інтерфейс для підключення RS232 або USB(через конвертер сигналів). Контроль температури незалежний. Клієнт «IRSOFT» (рис. 1.2) надає змогу лише підключитися до станції, створити температурний профіль або запустити існуючий, працює лише на операційній системі Windows. Станція керується переважно кнопками на корпусі, може зберігати в пам'яті до 10 температурних профілів.

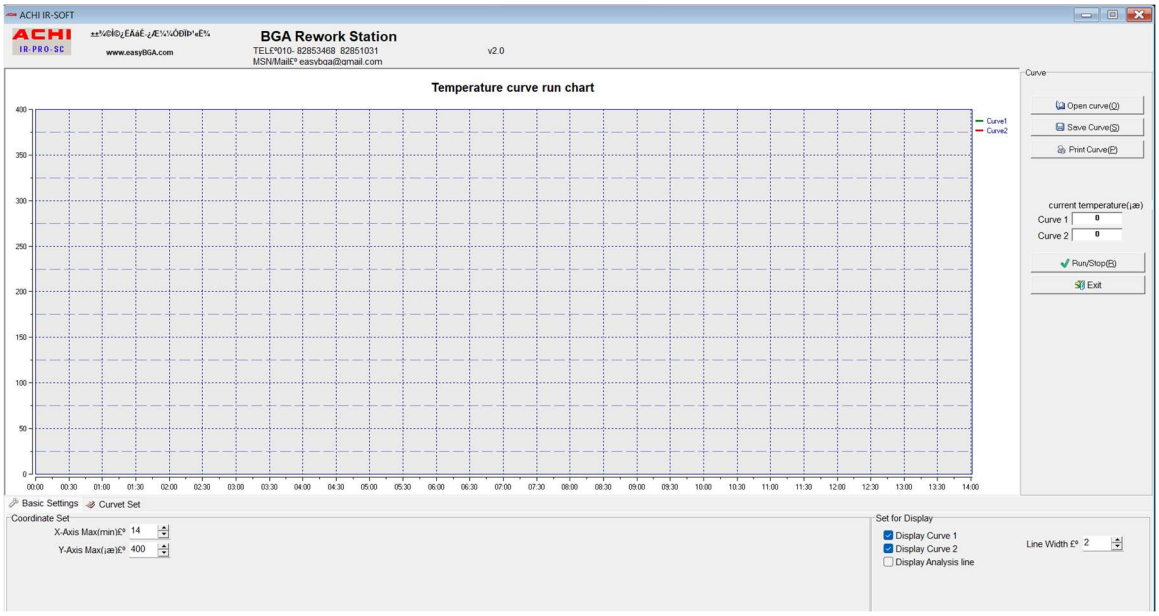


Рисунок 1.2 – Головний екран застосунку «IR-SOFT»

Quick 7710 – паяльно-ремонтний комплекс з мікропроцесорним управлінням преміум класу. Призначений для безпечного і точного монтажу або демонтажу BGA, а також інших компонентів з використанням технологій ІЧ-нагрівання і паяння гарячим повітрям, в тому числі безсвинцевим припоєм. Комплекс дозволяє повністю автоматично виконувати процеси запаювання та демонтажу мікросхем і може працювати як автономно, так і з підключенням до ПК. Інформація відображається на дисплеї та у клієнтському ПЗ «BGA Soft» (рис. 1.3).

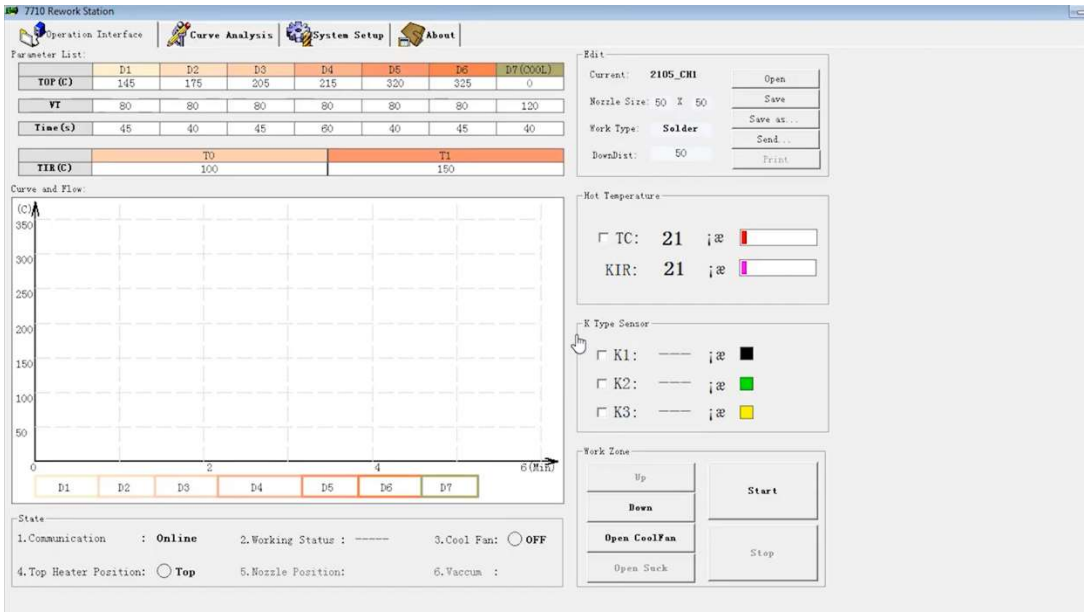


Рисунок 1.3 – Головний екран застосунку «BGA Soft»

Клієнтський застосунок до цієї станції має такі можливості:

– робота на операційній системі Windows;
– відображення поточної температури нагрівачів;
– управління станом кулерів;
– завантаження температурного профілю в пам’ять програми;
– аналіз та створення температурного профілю;
– системні налаштування (увімкнення спікера, пароль).

Термопро ИК-650 – це універсальна система для монтажу та відновлення BGA. Станція може використовуватись для ремонту плат ноутбуків, відеоприставок, материнських і серверних друкованих плат, плат телевізорів, моніторів, а також плат промислового, комунікаційного та іншого обладнання. Ціна на даний момент становить від 70 до 100 тисяч гривень (в залежності від комплектації). Вона має модульну структуру, контри для управління нагрівачами та самі інфрачервоні нагрівачі можна придбати окремо, програма автоматично розпізнає тип підключеного контролера. Він може працювати як автономно (підтримуючи статичну температуру), так і в зв’язці з клієнтом для ПК.

Особливості паяльної станції:

– точний контроль робочої температури;
– пайка по температурному профілю;
– робота зі свинцевими та безсвинцевими припоями;
– модульна конструкція, можливість модернізації;
– автономна робота з обмеженим функціоналом;
– управління за допомогою клієнтського ПЗ «Термопро-Центр».

Можливості ПЗ «Термопро-Центр» (рис. 1.4):

– робота на операційній системі Windows;
– відображення поточної, встановленої температури нагрівачів;
– створення температурних профілів;
– робота по температурному профілю;
– налаштування різних параметрів контролерів температури;
– графік температур нагрівачів, який працює в реальному часі.

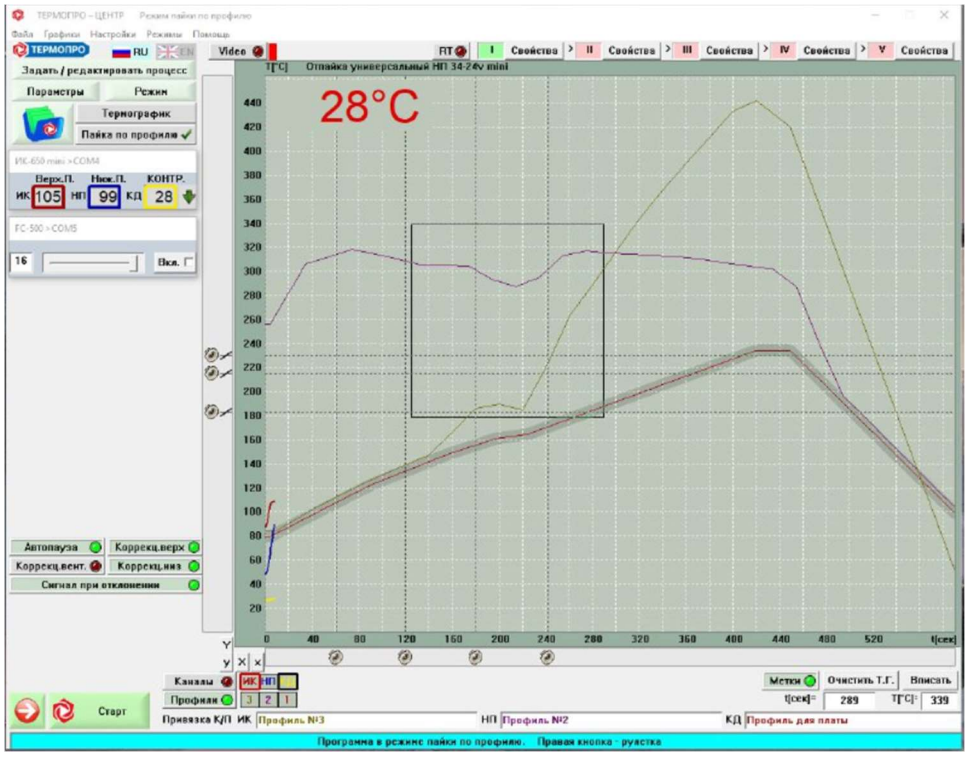


Рисунок 1.4 – Головний екран застосунку «Термопро-Центр»

Проаналізувавши наявні успішні рішення, можна помітити, що всі вони мають як переваги, так і недоліки. Основною перевагою цих рішень є налаштування за допомогою клієнтського програмного забезпечення, що робить процес більш гнучким та зручним. Також можна зазначити, що всі застосунки до розглянутих паяльних станцій працюють лише на операційній системі сімейства Windows, що є недоліком, оскільки потребує додаткових витрат на ліцензію для інженерів сервісного центру. Незважаючи на багатий функціонал, інтерфейс зазвичай виглядає застаріло, в окремих випадках некоректно відображається на сучасних операційних системах. В таблиці 1.1 можна побачити порівняння функціоналу всіх розглянутих аналогів зі своїм проєктом.

Таблиця 1.1 – Порівняння функціоналу

Функціонал	**Дипломний проєкт**	**IR-SOFT**	**BGA Soft**	**Термопро-Центр**
Графічне створення та редагування термпературних профілів	+	-	-	-
Робота по температурному профілю	+	+	+	+
Підтримка Linux	+	-	-	-
Кількість підтримуваних нагрівачів	Необмежено	2	2	2
Аналіз даних	-	-	+	-

1.4 Аналіз вимог до програмного забезпечення

Програмне забезпечення перш за все призначено для інженерів з ремонту техніки, які хочуть скористатися паяльною станцією для правильного нагріву материнської плати, яка ремонтується, та компонентів на ній. Отже, важливі вимоги такі: підключення контролерів, управління їхніми налаштуваннями, управління температурними профілями, перегляд зібраної з контролерів інформації на графіку та запуск роботи по температурному профілю. На рисунку 1.5 можна побачити повну діаграму варіантів використання[7].

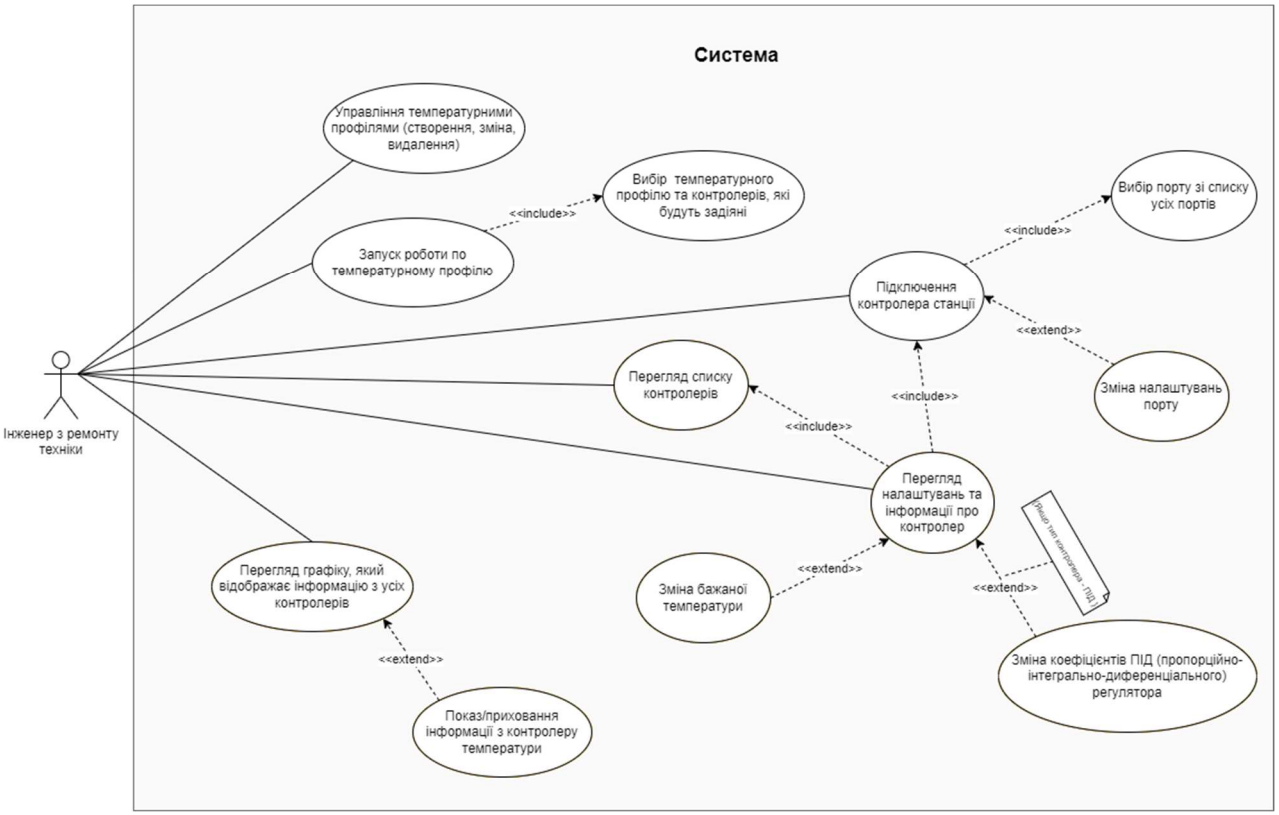


Рисунок 1.5 – Діаграма варіантів використання

В таблицях 1.2 - 1.12 наведені варіанти використання програмного забезпечення.

Таблиця 1.2 - Варіант використання UC-1

Use case name	Підключення контролера станції
Use case ID	UC-01
Goals	Підключення контролера станції до клієнтського застосунку
Actors	Інженер з ремонту
Trigger	Користувач бажає підключити контролер до програми
Pre-conditions	-
Flow of Events	Користувач запускає програму. Обирає порт для підключення зі списку доступних портів, натискає кнопку «Підключити». Після цього з’являється новий пристрій у списку пристроїв.
Extension	В випадку, якщо порт уже використовується або до порту підключено некоректний пристрій, має бути виведено відповідне повідомлення з помилкою.
Post-Condition	Поява нового пристрою в списку пристроїв.

Таблиця 1.3 - Варіант використання UC-2

Use case name	Зміна налаштувань порту
Use case ID	UC-02
Goals	Зміна та збереження налаштувань послідовного порту
Actors	Інженер з ремонту
Trigger	Користувач хоче змінити налаштування порту
Pre-conditions	Існує хоча б один порт.
Flow of Events	Користувач обирає порт зі списку, натискає кнопку «Розширені налаштування порту». Може змінити такі параметри як: швидкість, паритет, кількість бітів даних, стоп-біт. Після цього натискає кнопку «Ок».
Extension	-
Post-Condition	Внесено зміни в налаштування порту.

Таблиця 1.4 - Варіант використання UC-3

Use case name	Перегляд списку контролерів
Use case ID	UC-03
Goals	Отримання інформацію про всі підключені контролери
Actors	Інженер з ремонту
Trigger	Користувач хоче бачити всі підключені контролери.
Pre-conditions	-
Flow of Events	Користувач переглядає список пристроїв, після підключення нового пристрою він одразу ж з’являється в списку.
Extension	-
Post-Condition	В списку присутні підключені пристрої

Таблиця 1.5 - Варіант використання UC-4

Use case name	Перегляд налаштувань та інформації з контролера
Use case ID	UC-04
Goals	Отримання інформації щодо поточної та очікуваної температури, яку вимірює контролер
Actors	Інженер з ремонту
Trigger	Користувач хоче бачити інформацію з контролера
Pre-conditions	Підключено хоча б один пристрій.
Flow of Events	Користувач обирає контролер температури в списку підключених пристроїв. Переглядає інформацію з обраного контролера.
Extension	Температура періодично оновлюється.
Post-Condition	-

Таблиця 1.6 - Варіант використання UC-5

Use case name	Зміна температури контролера
Use case ID	UC-05
Goals	Зміна бажаної температуру контролера
Actors	Інженер з ремонту
Trigger	Користувач хоче змінити температуру контролера температури.
Pre-conditions	Підключено хоча б один пристрій
Flow of Events	Користувач обирає контролер температури в списку підключених пристроїв. Обирає або вводить значення бажаної температури у відповідне поле для вводу та натискає кнопку “Встановити”.
Extension	-
Post-Condition	Бажану температуру встановлено, температура нагрівача наближається до заданої.

Таблиця 1.7 - Варіант використання UC-6

Use case name	Налаштування коефіцієнтів PID контролера
Use case ID	UC-06
Goals	Зміна коефіцієнтів PID контролера
Actors	Інженер з ремонту
Trigger	Користувач хоче змінити коефіцієнти PID контролера.
Pre-conditions	Підключено пристрій з підтримкою PID управління
Flow of Events	Користувач обирає контролер температури в списку підключених пристроїв. В налаштуваннях PID обирає або вводить значення кожного з трьох коефіцієнтів, по черзі натискає кнопку «Встановити».
Extension	-
Post-Condition	Коефіцієнти змінено.

Таблиця 1.8 - Варіант використання UC-7

Use case name	Початок/зупинка збору даних з контролерів
Use case ID	UC-07
Goals	Розпочати або зупинити автоматичне отримання нових даних з контролера.
Actors	Інженер з ремонту
Trigger	Користувач хоче розпочати або зупинити отримання даних з конкретного контролера.
Pre-conditions	Підключено хоча б один пристрій
Flow of Events	Користувач натискає на кнопку або «чекбокс», щоб зупинити або розпочати оновлення даних з контролера.
Extension	-
Post-Condition	-

Таблиця 1.9 - Варіант використання UC-8

Use case name	Вибір температурного профілю та задіяних контролерів
Use case ID	UC-08
Goals	Обрати температурний профіль та контролери, які будуть задіяні
Actors	Інженер з ремонту
Trigger	Користувач хоче обрати температурний профіль та задіяні контролери для роботи за ним.
Pre-conditions	Підключено необхідну кількість контролерів для температурного профіля.
Flow of Events	Користувач натискає на головному вікні кнопку «Обрати і запустити температурний профіль», після цього відкривається вікно, в якому він може обрати температурний профіль зі списку та контролери для нього.
Extension	-
Post-Condition	-

Таблиця 1.10 - Варіант використання UC-9

Use case name	Управління температурними профілями
Use case ID	UC-9
Goals	Додати, видалити або відредагувати температурний профіль
Actors	Інженер з ремонту
Trigger	Користувач хоче управляти температурними профілями
Pre-conditions	-
Flow of Events	Користувач натискає на головному вікні кнопку «Управління температурними профілями», після цього відкривається вікно, в якому він може створювати, редагувати або видаляти температурні профілі.
Extension	-
Post-Condition	-

Таблиця 1.11 - Варіант використання UC-10

Use case name	Запуск роботи за температурним профілем
Use case ID	UC-10
Goals	Запуск роботи обраних контролерів за температурним профілем.
Actors	Інженер з ремонту
Trigger	Користувач хоче почати роботу за температурним профілем
Pre-conditions	Підключено необхідну кількість контролерів для температурного профіля.
Flow of Events	Користувач натискає на головному вікні кнопку «Обрати і запустити температурний профіль», після цього відкривається вікно, в якому він може обрати температурний профіль зі списку та контролери для нього.
Extension	-
Post-Condition	-

Таблиця 1.12 - Варіант використання UC-11

Use case name	Перегляд графіку температур
Use case ID	UC-11
Goals	Перегляд температури всіх контролерів на графіку
Actors	Інженер з ремонту
Trigger	Користувач хоче переглянути температури всіх контролерів на графіку
Pre-conditions	Підключено хоча б один пристрій і увімкнено автоматичне оновлення даних
Flow of Events	Користувач може побачити графік температур
Extension	-
Post-Condition	-

1.4.1 Розроблення функціональних вимог

На рисунку 1.6 зображено модель вимог у загальному вигляді.

#	**Name**	**Priority**	**Risk**
FR-1	Відображення доступних портів	High	High
FR-2	Зміна параметрів порту	Medium	Medium
FR-3	Підключення/відключення контролера	High	High
FR-4	Відображення списку підключених контролерів та їх стан	High	Medium
FR-5	Показ налаштувань контролера	High	Low
FR-6	Зміна налаштувань контролера	High	High
FR-7	Відображення інформації про кожен контролер на графіку	Medium	Medium
FR-8	Вимкнення або увімкнення збору даних з контролера температури	Low	Low
FR-9	Управління температурними профілями	High	High
FR-10	Запуск роботи за температурним профілем	High	High
FR-11	Відображення роботи за температурним профілем	Medium	Low

Рисунок 1.6 – Модель вимог у загальному вигляді

Функціональні вимоги та їх опис наведено в таблицях 1.13-1.23.

Таблиця 1.13 – Функціональна вимога FR-1

Назва	Відображення доступних портів
Опис	Система повинна надавати можливість користувачеві отримати список усіх підключених на даний момент портів.

Таблиця 1.14 – Функціональна вимога FR-2

Назва	Зміна параметрів порту
Опис	Система повинна надавати можливість зміни параметрів порту та зберігати їх.

Таблиця 1.15 – Функціональна вимога FR-3

Назва	Підключення/відключення контролера
Опис	Система повинна надавати можливість підключення та відключення підтримуваного контролеру паяльної станції.

Таблиця 1.16 – Функціональна вимога FR-4

Назва	Відображення списку підключених контролерів та їх стан
Опис	Система повинна відображати усі підключені пристрої та відомості про їх стан.

Таблиця 1.17 – Функціональна вимога FR-5

Назва	Показ налаштувань контролера
Опис	Система повинна надавати можливість вручну змінювати параметри кожного каналу контролера (бажану температуру, коефіцієнти регулятора, якщо є підтримка контролером станції).

Таблиця 1.18 – Функціональна вимога FR-6

Назва	Зміна налаштувань контролера
Опис	Система повинна надавати можливість вручну змінювати параметри кожного каналу контролера (бажану температуру, коефіцієнти регулятора, якщо є підтримка контролером станції).

Таблиця 1.19 – Функціональна вимога FR-7

Назва	Відображення інформації про кожен контролер на графіку
Опис	Система повинна надавати можливість відобразити криву температури відносно часу та бажану температура для кожного з регуляторів в реальному часі.

Таблиця 1.20 – Функціональна вимога FR-8

Назва	Вимкнення або увімкнення збору даних з контролера температури
Опис	Система повинна надавати можливість зупинити або знову запустити збір та оновлення даних з регулятора температури.

Таблиця 1.21 – Функціональна вимога FR-9

Назва	Управління температурними профілями
Опис	Система повинна надавати можливість створити, відредагувати або видалити температурний профіль. Кожен температурний профіль складається зі списку кривих температури, які можна створювати, видаляти та редагувати.

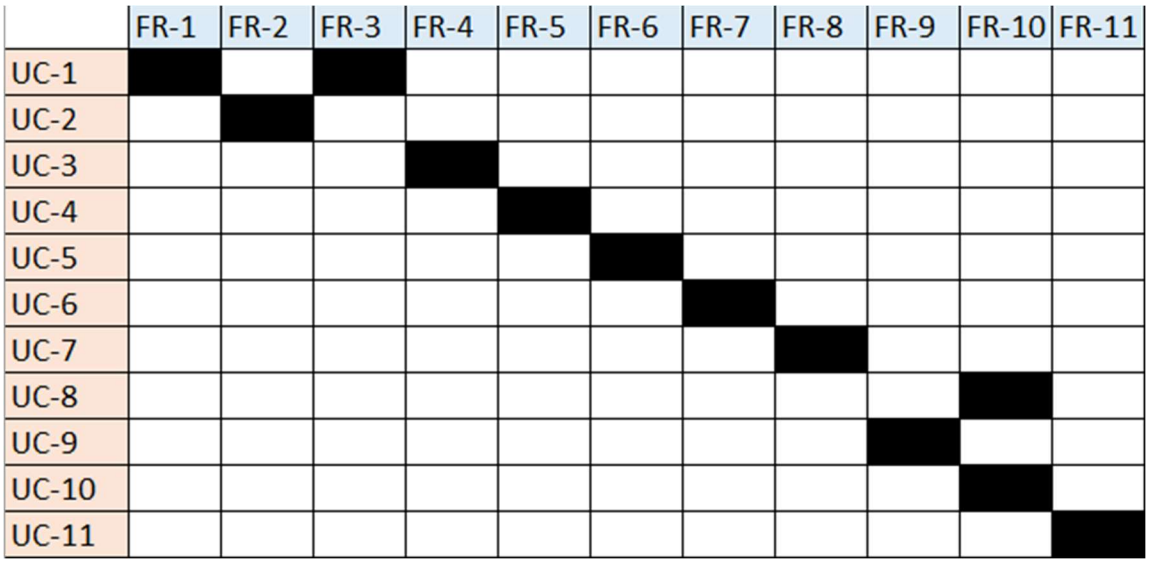
Таблиця 1.22 – Функціональна вимога FR-10

Назва	Запуск роботи за температурним профілем
Опис	Система повинна надавати можливість обрати температурний профіль, прив’язати регулятори температури до нього та запустити роботу.

Таблиця 1.23 – Функціональна вимога FR-11

Назва	Відображення роботи за температурним профілем
Опис	Система повинна вивести на графіку криві температурного профілю та інформацію з регуляторів, які виконують роботу на даний момент. Також необхідно вивести прогрес роботи.

На рисунку 1.6 зображено матрицю трасування вимог та варіантів використання.



	FR-1	FR-2	FR-3	FR-4	FR-5	FR-6	FR-7	FR-8	FR-9	FR-10	FR-11
UC-1	■		■								
UC-2		■									
UC-3				■							
UC-4					■						
UC-5						■					
UC-6							■				
UC-7								■			
UC-8										■	
UC-9									■		
UC-10										■	
UC-11											■

Рисунок 1.6 – Матриця трасування вимог та варіантів використання

1.4.2 Розроблення нефункціональних вимог

Нефункціональні вимоги програмного забезпечення визначають його характеристики та властивості, які не стосуються безпосередньо функціональності системи, але визначають якість, ефективність та інші аспекти системи. Основна відмінність нефункціональних вимог від функціональних полягає у тому, що вони описують те, як система має працювати, а не що система має робити.

Для застосунку з підтримки роботи паяльної станції, який розглядається в даній дипломній роботі визначено наступні нефункціональні вимоги:

– система має бути надійною, стійкою до відмов, бути відновлювальною;

– має бути сумісною з операційними системами Windows та на базі ядра Linux;

– повинна мати зручний, сучасний, інтуїтивно зрозумілий користувацький інтерфейс;

– повинна бути легко розширюваною, тобто має бути змога швидко додати новий тип контролера паяльної станції, новий функціонал або тип підключення;

– повинна мати підтримку різних локалізацій, включаючи, принаймні, українську та міжнародну (англійську).

1.5 Постановка задачі

Цільовою аудиторією даного програмного забезпечення є інженери з ремонту високотехнологічної техніки, яким потрібне сучасне, надійне та зручне клієнтське програмне забезпечення для управління паяльною станцією та програмне забезпечення для самого контролеру станції.

Метою розробки є забезпечення зручної роботи з паяльною станцією в автоматичному режимі завдяки розробці універсального, більш зручного та покращеного програмного забезпечення. Клієнтська програма повинна мати функціонал для підключення та конфігурування контролерів; створення, редагування та видалення температурних профілів, запуску режиму роботи за температурним профілем, перегляд графіку температур. Блок управління, в свою чергу, повинен мати змогу обробляти та виконувати команди як з персонального комп’ютера, так і з панелі керування на ньому.

Для досягнення мети необхідно вирішити наступні задачі:

– створити надійний протокол для обміну даними між клієнтським ПЗ та вбудованою системою у вигляді бібліотеки;

– розробити функціонал для пошуку та підключення підтримуваних пристроїв;

– забезпечити можливість управління всіма підключеними пристроями;

– реалізувати підтримку власного блоку управління та закласти можливість підключення іншого блоку в майбутньому;

– створити функціонал для перегляду графіків температур для кожного нагрівача, надати можливість зупинити відслідковування стану окремих нагрівачів;

– забезпечити підтримку контролерів, що підтримують управління декількома нагрівачами одночасно;

– створити функціонал для створення, редагування та видалення температурного профілю для одного або декількох контролерів;

– забезпечити можливість виконання температурного профілю для декількох контролерів одночасно;

– створити сучасний, зручний та адаптивний інтерфейс користувача для клієнтського ПЗ;

– реалізувати збереження та завантаження налаштувань та температурних профілів в базі даних;

– реалізувати обробку команд в реальному часі на стороні вбудованого пристрою;

– забезпечити можливість спрощеного управління контролером за допомогою панелі управління на ньому без підключення до ПК.

Висновки до розділу

У першому розділі було описано та детально проаналізовано предметну область, досліджено актуальність розробки програмного забезпечення для підтримки роботи паяльної станції. Також було досліджено способи вирішення задачі, вже наявні аналоги програмного забезпечення та їхні проблеми, які мають бути усунені в даній роботі. Потім було проаналізовано інструменти, які можна використати для розробки в контексті теми дипломної роботи, було обрано найбільш оптимальні інструменти, мови програмування та бібліотеки.

Під час аналізу вимог до програмного забезпечення були описані основні сценарії його використання. Після цього було досліджено та описано

функціональні та нефункціональні вимоги до програмного забезпечення. В кінці було виконано чітку постановку задачі, вирішення якої буде описано в наступних розділах пояснювальної записки.

2 МОДЕЛЮВАННЯ ТА КОНСТРУЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Моделювання та аналіз програмного забезпечення

Перед безпосередньо реалізацією програмного забезпечення варто розглянути та змоделювати бізнес процеси, в яких будуть задіяні усі користувачі застосунку. В попередньому розділі було визначено, що користувач лише один, тому він буде пов’язаний з усіма процесами. На рисунку 2.1 наведено структурну схему першого бізнес-процесу. Для опису використовується модель BPNM.

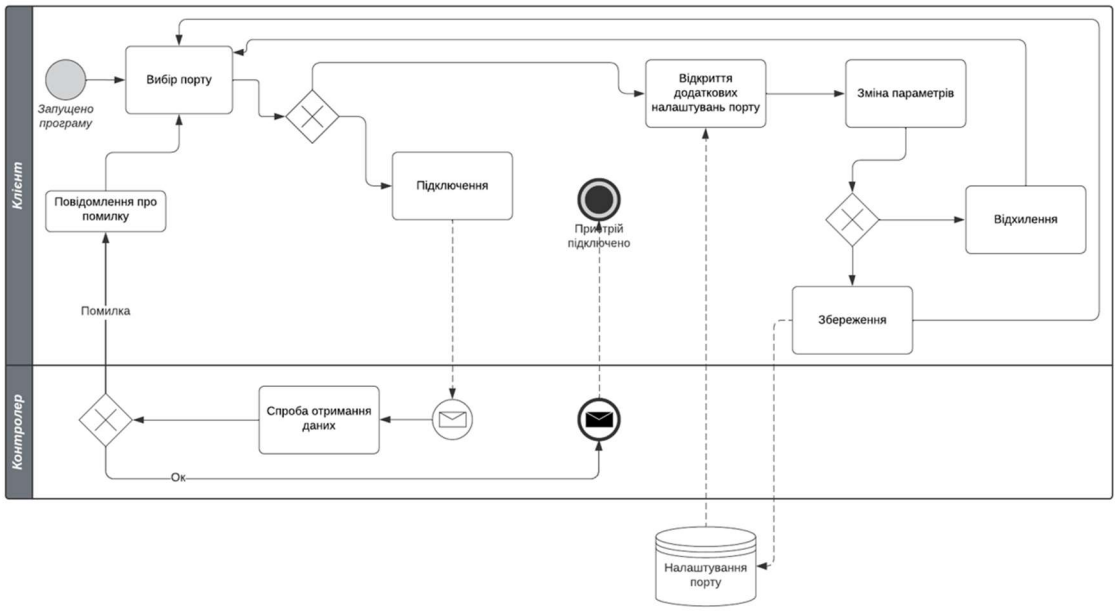


Рисунок 2.1 – Бізнес процес підключення пристрою

Першим бізнес процесом є підключення нового контролера. Опис даного процесу виглядає так:

– користувач запускає програму;

– обирає порт з переліку або відкриває додаткові налаштування порту, встановлює значення відповідно до інструкції контролера паяльної станції;

– натискає кнопку «Підключитися»;

– якщо підключення неможливе, з’являється вікно з помилкою;

– після успішного підключення контролер з’являється у списку пристроїв.

Другим бізнес процесом є перегляд та редагування налаштувань контролерів. На рисунку 2.2 зображено BPMN модель процесу.

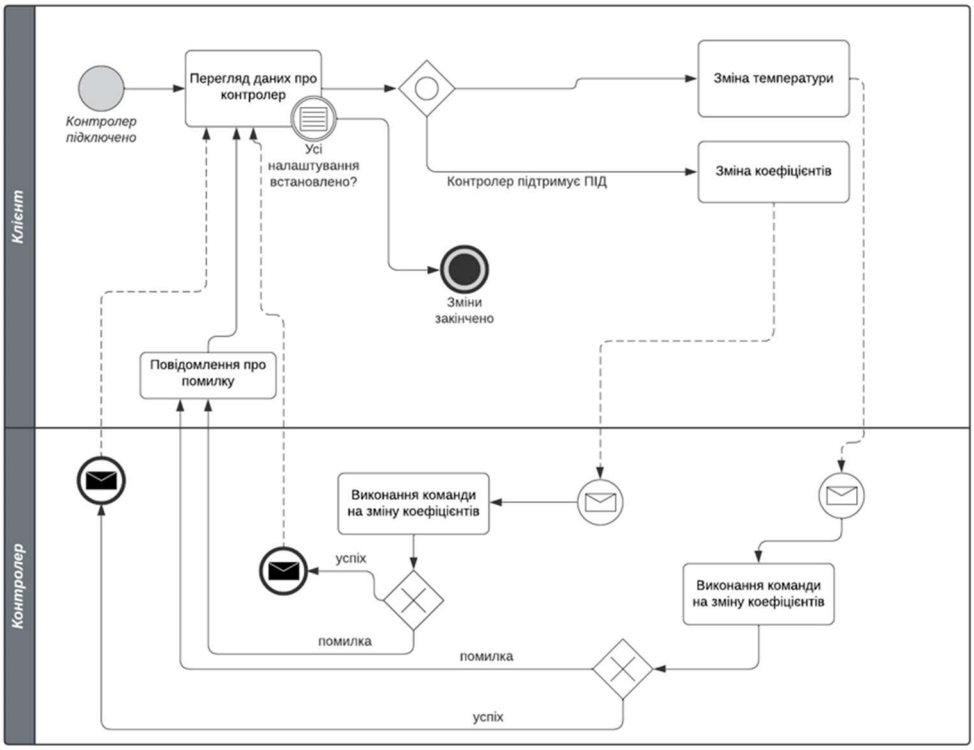


Рисунок 2.2 – Бізнес процес редагування налаштувань контролера

Опис даного процесу виглядає так:

– користувач розгортає налаштування контролера в списку підключених пристроїв;

– змінює значення температури, натискає кнопку «Встановити»

– якщо значення виходить за межі допустимих, то поле підсвічується червоним, нове значення не встановлюється.

– якщо контролер підтримує ПІД регулювання, то можемо змінити будь-який коефіцієнт.

Третім бізнес процесом є управління температурними профілями. Опис даного процесу виглядає так:

– користувач відкриває вікно управління температурного профілю;
– обирає температурний профіль, якщо список не пустий;
– створює новий температурний профіль;
– додає криву температурного профілю;
– редагує її;
– змінює ім’я температурного профілю та кривої;
– видаляє температурний профіль;
– зберігає або скасовує зміни.

Останнім бізнес процесом є запуск роботи за температурним профілем:

– користувач відкриває вікно запуску роботи за температурним профілем;
– обирає потрібний профіль;
– обирає нагрівачі серед підключених;
– якщо нагрівачів немає, користувач може тільки закрити вікно або обрати інший температурний профіль;
– розпочинає процес роботи за температурним профілем.

2.2 Архітектура програмного забезпечення

Від архітектури додатку залежить його якість, надійність та швидкість і легкість впровадження нового функціоналу в майбутньому, незважаючи на те, що користувач дуже часто навіть не знає, як побудований додаток, яким він користується. Архітектура дає можливість формалізувати вимоги клієнта, розділити програму на частини, які можуть бути реалізовані декількома розробниками паралельно, що дозволяє значно зекономити час розробки ПЗ. Крім того, вона сприяє збереженню чіткої документації, що спрощує роботу новим розробникам у проєкті і допомагає їм швидше розпочати працювати.

На початковому етапі проєктування важливо побудувати архітектуру всієї системи: виділити основні елементи системи та зв’язки між ними, щоб

було зрозуміло, як вони мають взаємодіяти між собою. На рисунку 2.3 показано архітектуру всієї системи для даної роботи.

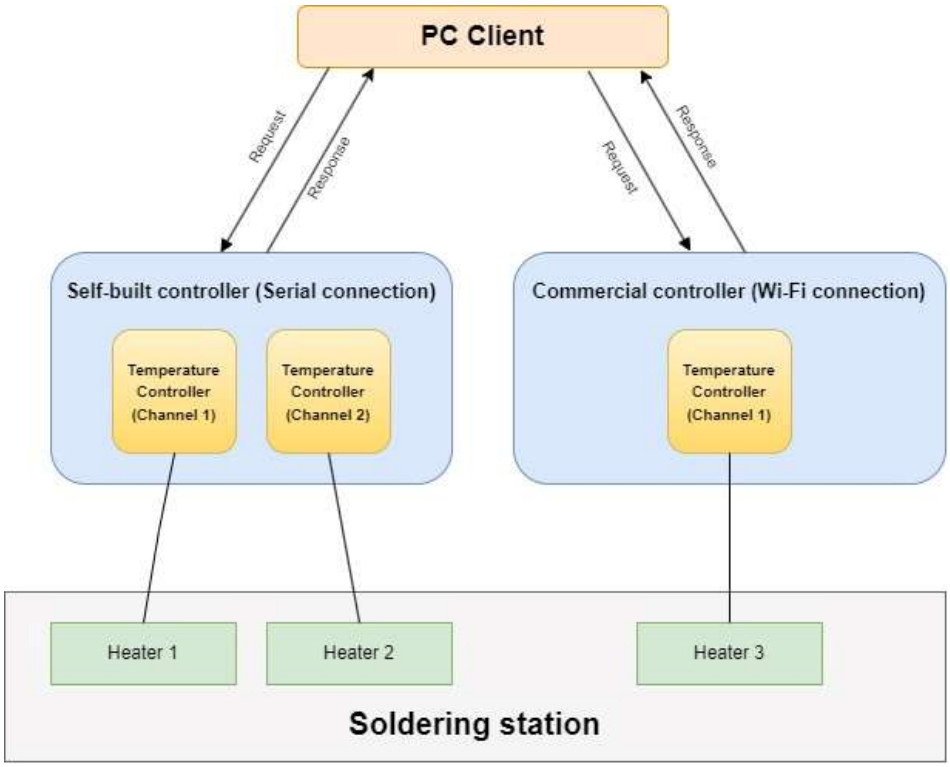


Рисунок 2.3 – Схема структурна системної архітектури

Вибір архітектури програми насамперед залежить від нефункціональних вимог програмного забезпечення, дуже часто для реалізації можна використовувати різні архітектури, тоді вибір вже частіше за все залежить від минулого досвіду розробників. Для побудови хорошої архітектури необхідно вміти обирати компромісні рішення, оскільки вимоги до різних характеристик можуть протирічити між собою.

2.2.1 Архітектура вбудованого програмного забезпечення

Для реалізації поставленої задачі необхідно перш за все обрати мікроконтролер, який буде відповідати усім вимогам, а також буде найбільш оптимальним варіантом.

MCU серії Atmega є одними з найбільш популярних на даний момент, належать до сімейства мікроконтролерів AVR[8] 8-bit. В останній час набрали

популярності завдяки платформі Arduino, яка використовує саме цю серію мікроконтролерів. Розглянемо цю серію на прикладі мікроконтролера Atmega8. Atmega8 – малопотужний восьмирозрядний мікроконтролер фірми Microchip, який належить до архітектури RISC. Має наступні характеристики:

– 130 інструкцій, більшість з яких виконуються за один такт;

– 8 Кбайт внутрішньосистемної самопрограмованої флеш-пам'яті;

– 512 байт енергонезалежної пам’яті (EEPROM);

– 1 Кбайт оперативної пам’яті (SRAM);

– 10 000 циклів запису/стирання флеш-пам’яті програм;

– 100 000 циклів запису/стирання пам’яті EEPROM;

– внутрішньосхемне програмування;

– 32 регістри вводу/виводу загального призначення (GPIO);

– 8-канальний 10-бітний АЦП (аналого-цифровий перетворювач);

– два 8-бітних таймера/лічильника з окремим попереднім дільником, один режим порівняння;

– один 16-бітний таймер/лічильник з окремим попереднім дільником, режимом порівняння та захоплення;

– внутрішній тактовий генератор на 8 МГц;

– інтерфейси USART та SPI.

Для контролера станції знадобиться:

– інтерфейс SPI для підключення цифрового підсилювача термопари;

– інтерфейс USART/UART;

– 11 GPIO для виводу температури на трьохзначний семисегментний індикатор;

– 1 GPIO для управління нагрівачем;

– 2 GPIO для кнопок регулювання.

Електричну схему підключення наведено в додатку Б. Структурну схему архітектури програми при такій конфігурації наведено на рисунку 2.4.

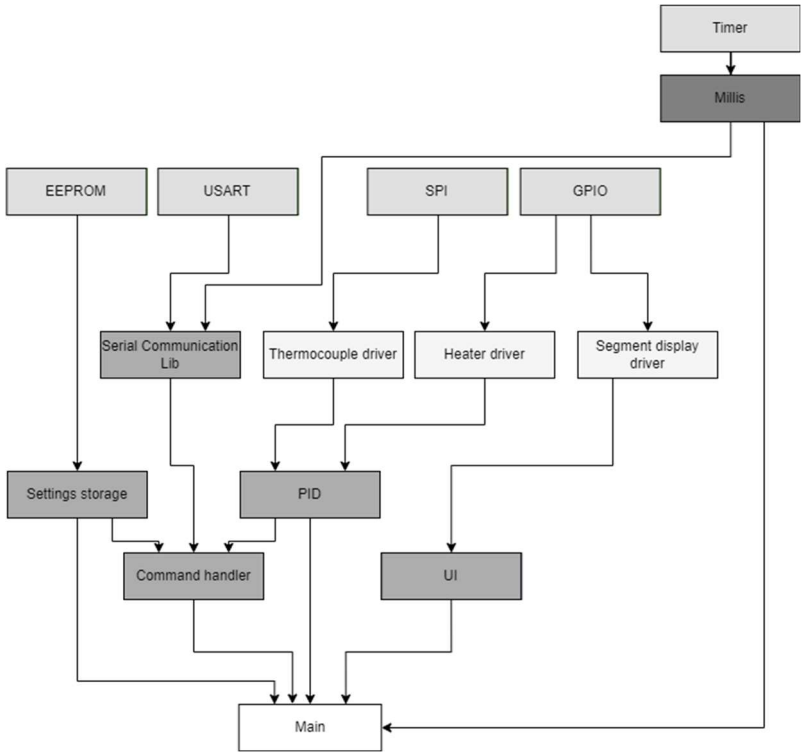


Рисунок 2.4 – Схема структурна архітектури програми контролера

Модулі програми розділено на шари, які виділено окремим кольором на структурній схемі. Найнижчий шар відображає безпосередньо периферію контролера, на вищому шарі знаходяться драйвери для фізичних пристроїв, і на найвищому знаходяться модулі, які відображають бізнес-логіку програми.

2.2.2 Архітектура бібліотеки для обміну даними

Для вирішення задачі обміну даними між контролером температури та комп’ютером є різні підходи. Можна виділити декілька найчастіше вживаних:

– команди у формі текстових рядків та термінального символу;

– використання стандартизованого протоколу, наприклад, Modbus;

– написання власного протоколу для обміну даними.

Обмін даними за допомогою команд у формі текстових рядків має як переваги, так і ряд недоліків. Основною перевагою є те, що цей формат даних

зручно читати людині, а також зручно виділяти окремі команди за допомогою термінального символу, наприклад, символу переносу рядка. Недоліком же є те, що у випадку передачі не лише текстової команди, а й певних даних (наприклад, температури), виникає потреба у конвертуванні простих типів даних у текст і навпаки. Такі операції є досить складними, і займають багато процесорного часу та пам'яті, якщо мова іде про мікроконтролер. Крім того, є ризик отримати некоректні дані під час виникнення перешкод у каналі зв'язку, оскільки цілісність даних ніяк не гарантується.

Modbus[9] – комунікаційний протокол, розроблений компанією Modicon ще в 1979 році, проте широко використовується навіть зараз. Цей протокол розроблявся спеціально для мережі промислових пристроїв, таких як: датчики, кнопки, панелі керування, панелі моніторингу та реле управління. Протокол Modbus має кілька варіантів, включаючи Modbus RTU, Modbus ASCII і Modbus TCP. Modbus RTU та Modbus ASCII використовуються для передачі даних через серійний інтерфейс, такий як RS-485, за допомогою двійкової або ASCII-кодування відповідно. Modbus TCP використовує TCP/IP для передачі даних через мережу Ethernet. Оскільки даний протокол був розроблений для промислових мереж ще в минулому сторіччі, то в ньому є і недоліки. Одним з найбільш вагомих недоліків є те, що в ньому немає механізму підтвердження доставки, отож гарантувати те, що команда дійшла до пристрою не можна.

Було вирішено створити власний бінарний напівдуплексний (master-slave) протокол, який вирішує проблеми вище розглянутих варіантів. Перед побудовою власне архітектури самої бібліотеки необхідно визначитися з специфікаціями протоколу передачі даних. Для забезпечення швидкості та надійності було вирішено використовувати бінарну передачу даних. Для початку необхідно створити специфікації для пакетів даного протоколу.

Вихідний пакет складається з:

– номеру команди (два байти);
– довжини параметрів (один байт);

– самих параметрів у вигляді бінарних даних (до 250 байт), довжина визначається попереднім параметром;

– контрольної сума (два байти).

Приклад вихідного пакету можна побачити на рисунку 2.5.

Рисунок 2.5 – Схема структурна вихідного пакету даних

Вхідний пакет (відповідь на команду) складається з:

– коду, який сигналізую про статус результату виконання (один байт);

– довжини даних відповіді (один байт);

– самих даних у вигляді бінарних даних (до 250 байт), довжина визначається попереднім параметром;

– контрольної сума (два байти).

Приклад вхідного пакету можна побачити на рисунку 2.6

Рисунок 2.6 – Схема структурна вхідного пакету даних

Алгоритм головного пристрою працює таким чином:

– будується та відправляється повний пакет даних;

– очікується хоча б один байт в буфері послідовного порту;

– якщо даних немає, то виникає помилка, якщо є – програма переходить до наступного кроку;

– очікується заголовок (2 байти);

– якщо заголовок отримано, програма переходить до наступного кроку, якщо ні, то виникає помилка;

– очікується кількість байтів, вказана в 2 байті заголовку плюс два (контрольна сума);

– якщо повний пакет було отримано, перевіряється контрольна сума, якщо ні – повертається помилка;

– якщо контрольна сума зійшлася, то перевіряється відповідь від контролера, яка надійшла в першому байті заголовку, якщо ні, то викликається помилка;

– якщо в першому байті одиниця, то метод повертає дані, отримані від контролера, якщо вони наявні.

Алгоритм залежного пристрою працює наступним чином:

– очікується заголовок (3 байти);

– якщо заголовок отримано, програма переходить до наступного кроку, якщо ні, то виникає помилка;

– очікується кількість байтів, вказана в 3 байті заголовку плюс два (контрольна сума);

– якщо повний пакет було отримано, то перевіряється контрольна сума, якщо ні – повертається помилка;

– якщо контрольна сума зійшлася, то запускається обробник для команди та надсилається відповідь;

2.2.3 Архітектура клієнтського програмного забезпечення

Для клієнтського застосунку було використано класичну багатошарову архітектуру[10]. Клієнтський додаток складається з чотирьох шарів, які можна побачити на рисунку 2.7.

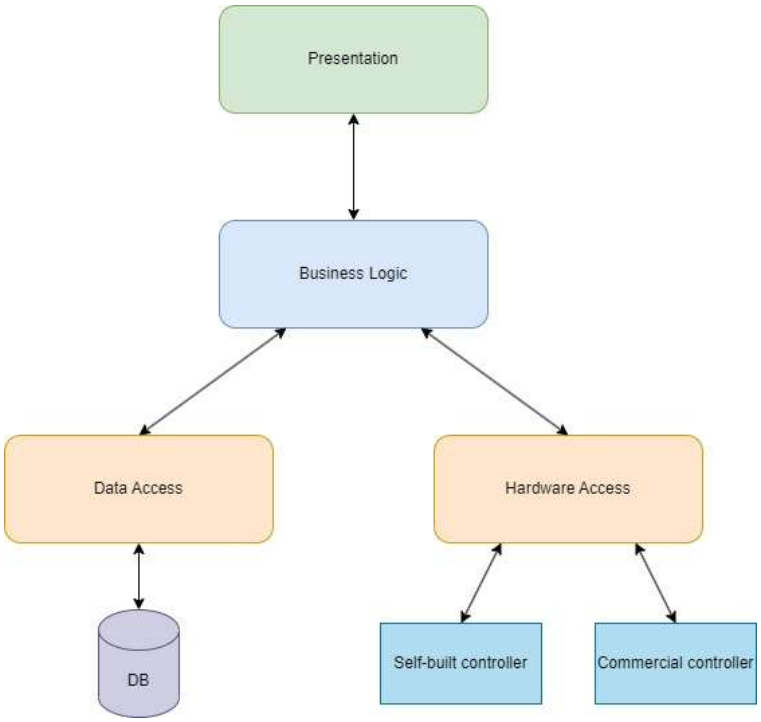


Рисунок 2.7 – Архітектура клієнтського додатку

Шар доступу до даних відповідає за взаємодію з базою даних та абстрагує її, тобто дозволяє використати будь-яку базу даних. Таких підхід дозволяє не лише змінити СУБД, але і легко протестувати бізнес-логіку додатку. Використовується шаблон проектування UnitOfWork[11]. Основна ідея UnitOfWork полягає в тому, що всі операції збереження, оновлення та видалення даних виконуються в межах одного об'єкта, який утримує стан змінених об'єктів. Після виконання всіх операцій, UnitOfWork запускає транзакцію бази даних і зберігає всі зміни одночасно. Якщо транзакція успішна, то зміни фіксуються у базі даних, а в іншому випадку вони скасовуються. Шаблон зазвичай використовується разом з іншим шаблоном, який називається «Репозиторій». Він в свою чергу визначає стандартизований інтерфейс для роботи з об'єктами даних, тобто надає методи для створення, читання, оновлення та видалення об'єктів даних, абстрагуючи деталі взаємодії з базою даних. Для виконання складних запитів можна скористатися архітектурним шаблоном «Спеціалізація», яка дозволяє розділити логіку запитів та самі репозиторії, що робить код легшим для тестування. На рисунку 2.8 зображено ER-діаграму в нотації Чена.

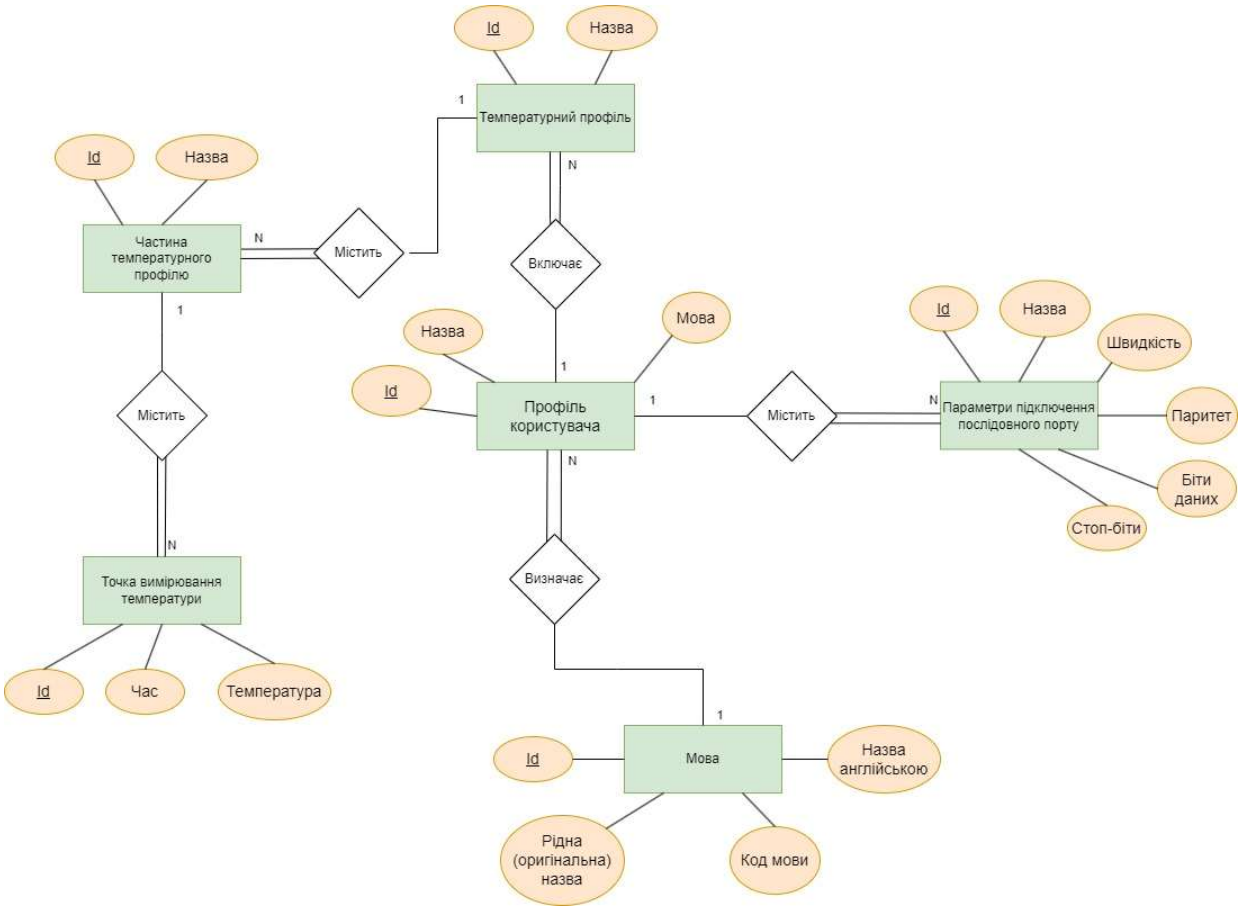


Рисунок 2.8 – ER-діаграма

Шар доступу до контролерів абстрагує реалізацію підключень та команд, які може виконувати контролер. Також він дозволяє отримати інформацію про всі підключені контролери температури без безпосереднього звертання до них. Згідно з нефункціональними вимогами, програма повинна бути розширюваною та забезпечувати обернену та пряму сумісність з контролерами. Це є проблемою, оскільки без порушення принципів SOLID це зробити неможливо. Така проблема називається «expression problem», тобто проблема виразів. Це складна проблема в мовах програмування зі статичною типізацією, яка стосується розширюваності і модульності абстракцій даних. Вона полягає в тому, що до абстракцій не можна додавати типи та поведінку одночасно без зміни вже існуючого коду, зберігаючи при цьому безпечність типів. Вона існує як об’єктно-орієнтованих мовах програмування, так і функціональних. Вперше її помітив Джон Рейольдс у 1975 році, а сформулював Філіп Уодлер. Оскільки пристрої будуть об’єднуватися за

схожим функціоналом, то можна використати поліморфізм та реалізувати окремі інтерфейси в кожному з пристроїв. Далі через менеджер можна отримати інформацію про те, який функціонал реалізує той чи інший пристрій та отримати доступ до нього. Завдяки композиції та узагальненому типу даних кожен пристрій зможе реалізувати будь-який тип підключення без зміни коду самого пристрою. Структурну схему класів шару можна побачити на рисунку 2.9.

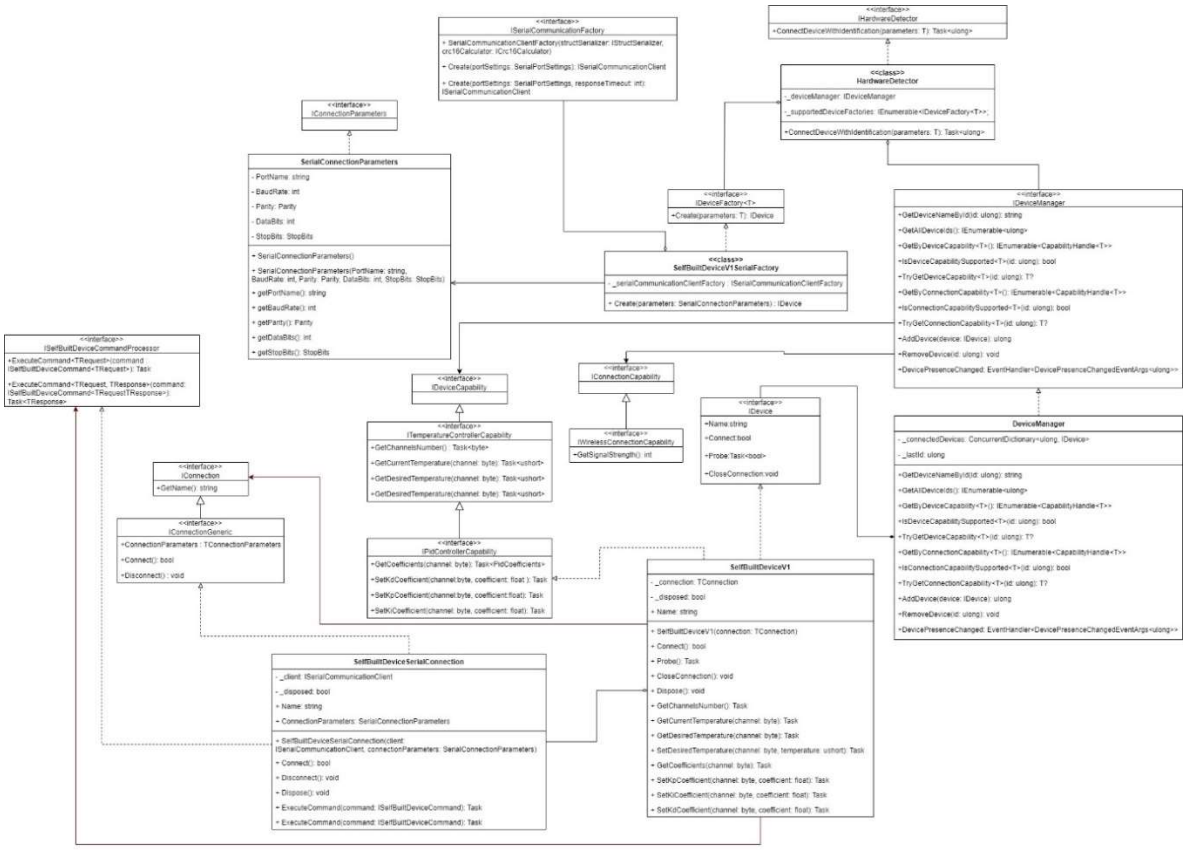


Рисунок 2.9 – Схема структурна класів шару доступу до пристроїв

Шар бізнес-логіки відповідає за реалізацію всіх варіантів використання, які відбуваються в рамках предметної області, тобто підключення контролерів, збирання з них інформації, управління ними, управління температурними профілями та іншим. Кожен з сервісів цього шару звертається до шару доступу до даних або шару доступу до контролерів, оброблює дані, та повертає бізнес-моделі на верхній шар – шар представлення.

Шар представлення побудований за архітектурним шаблоном MVVM. Шаблон дозволяє повністю відділити бізнес-логіку від представлення. Для передачі даних між View та ViewModel використовується механізм «Binding». Модель повністю незалежна від шару представлення, вона представляє бізнес-логіку та дані програми. ViewModel є посередником між представленням та моделлю, він викликає методи моделі та оновлює свій стан при виникненні подій чи отриманні даних від моделі. Представлення (View) відповідає за візуальне відображення елементів користувачу, анімації, переходи. Архітектуру шару представлення зображено на рисунку 2.11.

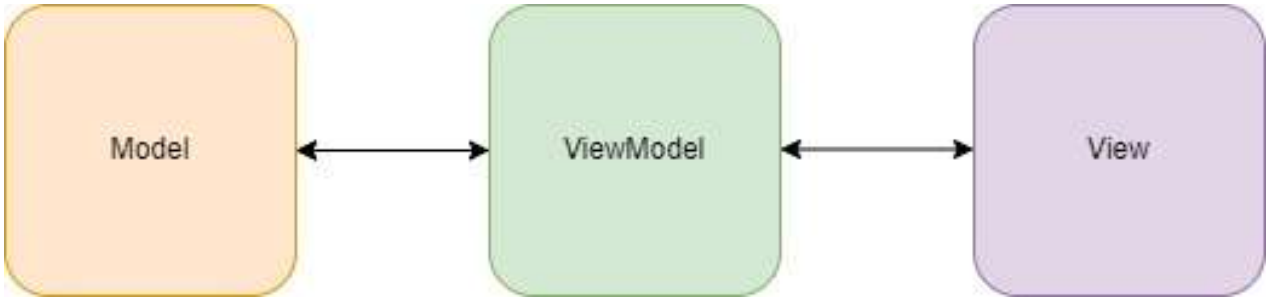


Рисунок 2.11 – Архітектура шару представлення

2.3 Конструювання програмного забезпечення

Далі розглянуто конструювання клієнтського програмного забезпечення, власної бібліотеки для обміну даними між клієнтським застосунком та мікроконтролером, а також вбудованого програмного забезпечення в окремих підрозділах.

2.3.1 Конструювання вбудованого програмного забезпечення

Повний перелік модулів наведено в таблиці 2.1.

Таблиця 2.1 – Повний перелік модулів вбудованого ПЗ

Модуль	Опис
Heater	Відповідає генерацію сигналу, який управляє нагрівачем.

Кінець таблиці 2.1

Модуль	Опис
thermocouple	Відповідає за отримання даних з термопари
pid	Реалізує регулятор температури з термопари та нагрівача.
seven_seg_display	Драйвер семисегментного дисплею
ui	Відповідає за відображення різних даних на семисегментному дисплеї
main	Точка входу в програму
serial_communcication	Протокол обміну даними (slave частина).
settings_storage	Зберігання налаштувань в енергонезалежній пам’яті
serial_crc	Обрахунок контрольної суми
usart	Відповідає за відправку та отримання даних по USART
gpio	Драйвер виводів загального призначення (GPIO)
millis	Містить функцію для отрмання часу з моменту запуску мікроконтролера в секундах.

2.3.2 Конструювання бібліотеки для обміну даними

Повний перелік класів бібліотеки наведено в таблиці 2.2.

Таблиця 2.2 – Повний перелік модулів бібліотеки для обміну даними

Клас	Опис
SerialCommunicationClient	Головний клас бібліотеки, який реалізує протокол передачі даних
SerialPortDefault	Клас послідовного порту, є обгорткою над статичним класом SerialPort

Кінець таблиці 2.2

Клас	Опис
Crc16Calculator	Калькулятор контрольної суми
StructSerializer	Серіалізатор структур в бінарні дані
SerialPortSettings	Параметри підключення до послідовного порту
SerialCommunicationClientFactory	Фабрика підключень

В таблиці 2.3 наведено усі публічні методи основного класу бібліотеки.

Таблиця 2.3 – Перелік публ. методів класу SerialCommunicationClient

Метод	Опис
Connect()	Підключення
Close()	Закриття порту
ExecuteCommand(commandCode: ushort, parameters: byte[]) : byte[]	Виконує команду, де в якості параметрів та результату виступає масив байтів.
ExecuteCommand<TParameters> (commandCode: ushort, parameters: TParameters) : void	Виконує команду без відповіді, де в якості параметрів виступає структура.
ExecuteCommand<TParameters, TResult> (commandCode: ushort, parameters: TParameters) : TResult	Виконує команду, де в якості параметрів та результату виступає структура даних.

2.3.3 Конструювання клієнтського програмного забезпечення

Повний перелік класів шару доступу до даних наведено в таблиці 2.4.

Таблиця 2.4 – Перелік класів шару доступу до даних

Клас	Опис
SolderingStationDbContext	Контекст бази даних
SolderingStationDbContextFactory	Фабрика контекстів, використовується тільки при створенні міграцій
UnitOfWork	Клас реалізує шаблон «Одиниця роботи»
GenericRepository<TEntity>	Клас реалізує шаблон «Репозиторій», є загальним, має методи для специфікацій.
DbServices	Статичний клас з методом розширення, для IoC контейнера, який додає всі сервіси шару роботи з даними до контейнера.
LanguageConfiguration	Клас конфігурації для моделі «LanguageEntity»
ProfileConfiguration	Клас конфігурації для моделі «ProfileEntity»
SerialConnectionParameters Configuration	Клас конфігурації для моделі «SerialConnectionParametersEntity»
TemperatureMeasurementPoint Configuration	Клас конфігурації для моделі «TemperatureMeasurementPointEntity»
ThermalProfileConfiguration	Клас конфігурації для моделі «ThermalProfileEntity»
ThermalProfilePartConfiguration	Клас конфігурації для моделі «ThermalProfilePartEntity»
20230601063104_InitialCreate	Автоматично згенерована початкова міграція бази даних

Самі моделі більш детально розглянуто далі в цьому ж розділі.

Повний перелік класів шару доступу до контролерів наведено в табл. 2.5.

Таблиця 2.5 – Перелік класів шару доступу до контролерів

Клас	Опис
DeviceManager	Точка доступу до всіх підключених пристроїв
HardwareDetector	Клас, відповідальний за автоматичне визначення підключеного пристрою
SelfBuiltDeviceV1	Клас конкретного пристрою зі своїм набором функцій
SelfBuiltDeviceSerialConnection	Реалізація підключення через послідовний порт для пристрою SelfBuiltDeviceV1
SelfBuiltDeviceV1SerialFactory	Фабричний метод для пристрою SelfBuiltDeviceV1 з послідовним підключенням
HardwareServices	Статичний клас з методом розширення для IoC контейнера, який додає всі сервіси шару роботи з пристроями
GetCurrentTemperatureCommand	Команда отримання поточної температури для пристрою SelfBuiltDeviceV1
GetDesiredTemperatureCommand	Команда отримання бажаної температури для пристрою SelfBuiltDeviceV1
GetPidCoefficientsCommand	Команда отримання коефіцієнтів ПІД регулятора для пристрою SelfBuiltDeviceV1
GetChannelsNumberCommand	Команда отримання кількості каналів регуляторів для пристрою SelfBuiltDeviceV1

Кінець таблиці 2.5

Клас	Опис
SetDesireTemperatureCommand	Команда встановлення бажаної температури для пристрою SelfBuiltDeviceV1
SetKpCoefficientCommand	Команда встановлення пропорційного коефіцієнту для пристрою SelfBuiltDeviceV1
SetKiCoefficientCommand	Команда встановлення інтегрального коефіцієнту для пристрою SelfBuiltDeviceV1
SetKdCoefficientCommand	Команда встановлення диференціального коефіцієнту для пристрою SelfBuiltDeviceV1
CapabilityHandle	Модель, яка надає доступ до певного функціоналу пристрою
DevicePresenceChangedEventArgs	Модель, яка повертається класом DeviceManager при підключенні або відключенні пристрою.
PidCoefficients	Модель PID коефіцієнтів
SerialConnectionParameters	Параметри підключення за допомогою послідовного порту

Повний перелік класів шару бізнес логіки наведено в таблиці 2.6.

Таблиця 2.6 – Перелік класів шару бізнес логіки

Клас	Опис
DeviceConnectedEventArgs	Модель, яка відправляється подією підключення пристрою у шар представлення

Продовження таблиці 2.6

Клас	Опис
Device	Модель пристрою, є фасадом для всіх пристроїв, містить методи, які дозволяють дізнатися, які саме функції реалізує конкретний контролер, наприклад, чи підтримує він зміну PID коефіцієнтів
DeviceDisconnectedEventArgs	Модель, яка відправляється подією відключення пристрою у шар представлення
Locale	Модель локалізації (назва, код)
PortPresenceEventArgs	Модель, яка відправляється подією зміни статусу підключення порту
ThermalProfile	Модель всього температурного профілю
ControllerThermalProfile	Модель однієї частини температурного профілю
SerialPortInfo	Модель інформації про послідовний порт (назва; ід пристрою, якщо він підключений)
SerialPortInfoEventArgs	Модель, яка відправляється подією оновлення даних порту
SerialPortInfoDto	Модель об’єкту передачі даних, містить інформацію про статус підключення та назву порту
PidCoefficients	Модель, що містить PID коефіцієнти контролера

Продовження таблиці 2.6

Клас	Опис
PidTemperatureController	Модель даних контролера температури, який підтримує PID регулювання
TemperatureController	Модель найпростішого контролера температури
TemperatureControllerKey	Ідентифікатор контролера температури, який складається з ідентифікатору пристрою та порядкового номеру регулятора в підключеному пристрої
TemperatureMeasurement	Модель виміру температури у конкретний час
TemperatureMeasurementEventArgs	Модель, яка відправляється подією вимірювання температури сервісу моніторингу температури
JobStartedEventArgs	Модель, яка відправляється підписникам після запуску роботи (наприклад, роботи по термопрофілю)
JobProgressUpdatedEventArgs	Модель, яка відправляється підписникам після кожної зміни прогресу виконання
DeviceService	Сервіс, який повідомляє шар представлення про підключення/відключення пристроїв, дозволяє отримати інформацію про всі пристрої

Продовження таблиці 2.6

Клас	Опис
JobStateService	Сервіс стану наявності та виконання роботи
LocalizationService	Сервіс локалізації, дозволяє отримати інформацію про доступні мови, зберігати обрану мову
SerialPortsService	Сервіс управління послідовними портами
TemperatureControllerService	Сервіс температурного контролера
PidTemperatureControllerService	Сервіс контролера температури з підтримкою PID, розширює сервіс TemperatureControllerService
ThermalProfileService	Сервіс, що дозволяє отримувати інформацію про температурні профілі, зберігати їх, редагувати
ThermalProfileProcessingService	Сервіс, що відповідає за вибір температурного профілю, контролерів для нього та запуск роботи
TemperatureMonitorService	Сервіс, що відповідає за моніторинг температур усіх контролерів
SerialPortsMonitor	Відповідає за моніторинг послідовних портів
SerialPortsProvider	Відповідає за отримання списку усіх портів, використовується сервісом з моніторингу портів

Кінець таблиці 2.6

Клас	Опис
Timer	Клас таймера, оболонка над System.Timers
ThermalProfileProcessingJob	Клас, що відповідає за роботу по температурному профілю.
ThermalProfileProcessingJobFactory	Фабрика для створення класу роботи по температурному профілю.
BusinessLogicServices	Статичний клас з методом розширення, що додає сервіси в IoC контейнер.

Повний перелік класів презентаційного шару наведено в таблиці 2.7.

Таблиця 2.7 – Перелік класів презентаційного шару

Клас	Опис
Program	Точка входу в програму
ViewLocator	Автоматично згенерований клас фреймворком Avalonia, відповідає за створення View в рантаймі.
App	Відповідає за конфігурацію додатку, управління життєвим циклом.
ApplicationDispatcher	Відповідальний за синхронізацію доступу до ресурсів та оновлення інтерфейсу з використанням багатопоточності.
ResourceProvider	Провайдер ресурсів (рядків локалізації)
MessageBoxService	Сервіс для відображення віконець повідомлень

Продовження таблиці 2.7

Клас	Опис
MainWindowViewModel	Модель представлення головного вікна програми
DevicesListViewModel	Модель представлення списку пристроїв
DeviceViewModel	Модель представлення списку пристроїв
ConnectionViewModel	Модель представлення управління підключеннями
LanguageSettingsViewModel	Модель представлення вибору мови
LanguageViewModel	Модель представлення мови
LanguageViewModel	Модель представлення мови
SerialPortInfoViewModel	Модель представлення інформації про послідовний порт.
SerialPortAdvancedSettings WindowViewModel	Модель представлення вікна налаштування послідовного порту.
TemperatureControllerSettings ViewModel	Модель представлення налаштувань загального контролера температури
PidTemperatureControllerSettings ViewModel	Модель представлення налаштувань контролера температури з підтримкою PID управління
TemperatureControllerViewModel	Модель представлення загального контролера температури на графіку.
DefaultDeviceViewModelFactory	Фабрика моделей представлення пристроїв
DefaultTemperatureViewModelFactory	Фабрика моделей представлення контролерів температури

Для збереження налаштувань та температурних профілів потрібно створити базу даних. Для даного застосунку було вирішено використати однофайлову СУБД SQLite[12], оскільки вона ідеально підходить для десктопних застосунків. На відміну від інших СУБД вона не має окремого серверного, а робота відбувається напряму з текстовим файлом. Формат бази даних є кросплатформеним, що задовольняє вимоги до ПЗ. При конструюванні шару доступу до даних було використано фреймворк «EntityFramework.Core» та підхід Code First[13]. Опис таблиць бази даних наведено у таблицях 2.8 - 2.13. Модель бази даних наведена на рисунку 2.10.

Таблиця 2.8 – Опис таблиці «Profiles»

Таблиця	Назва поля	Тип даних	Опис
Profiles	Id	integer	Ідентифікаційний номер
	Name	text	Назва профілю користувача
	LanguageId	text	Посилання на мову, яку використовує користувач

Таблиця 2.9 – Опис таблиці «SerialConnectionsParameters»

Таблиця	Назва поля	Тип даних	Опис
SerialConnectionsParameters	Id	integer	Ідентифікаційний номер
	PortName	text	Назва порту
	BaudRate	integer	Назва мови мовою носіїв в оригіналі.
	Parity	integer	Біт парності
	DataBits	integer	Кількість бітів даних в байті
	StopBits	integer	Число стопових бітів в байті

Таблиця 2.10 – Опис таблиці «Languages»

Таблиця	Назва поля	Тип даних	Опис
Languages	Id	integer	Ідентифікаційний номер
	EnglishName	text	Назва мови англійською
	NativeName	text	Назва мови мовою носіїв в оригіналі.
	Code	text	Код мови у форматі ISO 639-1

Таблиця 2.11 – Опис таблиці «ThermalProfiles»

Таблиця	Назва поля	Тип даних	Опис
ThermalProfiles	Id	integer	Ідентифікаційний номер
	Name	text	Назва температурного профілю
	ProfileEntityId	integer	Посилання на профіль з налаштуваннями.

Таблиця 2.12 – Опис таблиці «ThermalProfileParts»

Таблиця	Назва поля	Тип даних	Опис
ThermalProfileParts	Id	integer	Ідентифікаційний номер
	Name	text	Назва температурного профілю
	Color	integer	Колір у форматі ARGB, обраний користувачем при побудові температурного профілю
	ThermalProfileEntityId	integer	Посилання на запис в таблиці ThermalProfileEntityId, де зберігається інформація про всі термпературні профілі

Таблиця 2.13 – Опис таблиці «TemperatureMeasurementPoints»

Таблиця	Назва поля	Тип даних	Опис
TemperatureMeasurementPoints	Id	integer	Ідентифікаційний номер
	SecondsElapsed	real	Кількість секунд, які мають пройти з моменту запуску роботи за термпературним профілем
	Temperature	integer	Значення температури в момент часу
	ThermalProfilePartEntityId	integer	Посилання на запис в таблиці ThermalProfilePartEntityId, де зберігається інформація про всі криві температурних профілів

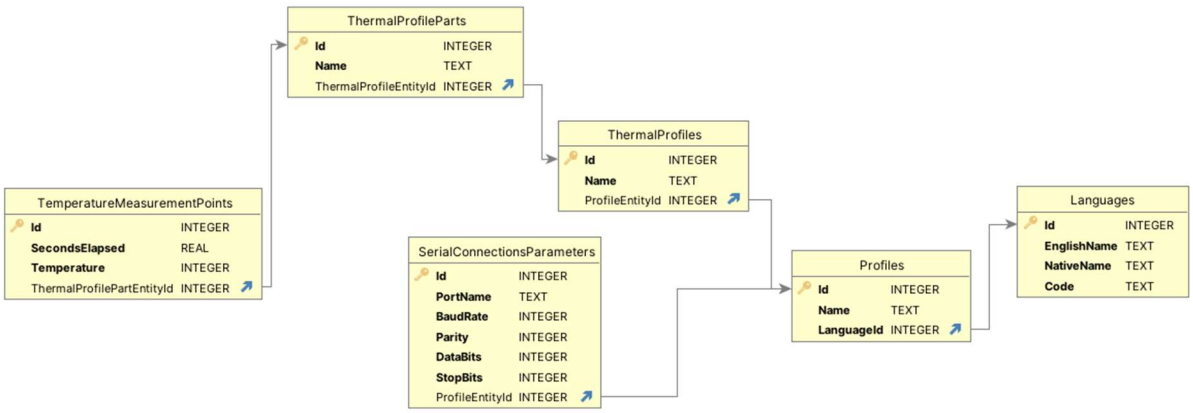


Рисунок 2.13 – Схема бази даних

Висновки до розділу

В другому розділі було побудовано бізнес-процеси у рамках предметної області. За ними було змодельовано архітектуру для всіх шарів клієнтського програмного забезпечення: шару доступу до даних, шару доступу до підключених пристроїв, шару бізнес логіки та презентаційного шару. Шар доступу до даних містить налаштування користувача та усі дані про температурні профілі. Шар доступу до підключених пристроїв забезпечує підключення, відключення, та управління контролерами, які можуть мати власний інтерфейс та неповну реалізацію всього можливого функціоналу, який підтримує даний застосунок. Шар бізнес-логіки має доступ до попередніх шарів та реалізує усі варіанти використання програмного забезпечення, які було розглянуто у першому розділі. Презентаційний шар є шаром верхнього рівня, його було реалізовано за допомогою архітектурного шаблону MVVM, що дозволяє повністю відокремити модель від представлення.

Також було спроектовано та реалізовано власний простий, зручний та надійний протокол передачі даних між клієнтським застосунком та контролером. Він дозволяє передавати різні команди та отримувати відповіді від мікроконтролера, що надає можливість налаштувати параметри контролера станції, отримати дані в реальному часі або змінити температуру нагрівачів, якими він управляє.

3 АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Аналіз якості ПЗ

Для аналізу якості програмного забезпечення застосовується тестування та аналіз метрик коду. Тестування проводиться для перевірку функціоналу коду, а метрики використовуються для оцінки різних характеристик коду, що допомагають поліпшувати його якість та ефективність розробки. Одним з основних показників є «Maintainability», цей показник має бути не менше 80%, якщо менше, то це свідчить про те, що такий код буде важче підтримувати. На рисунку 3.1 зображено аналіз метрик клієнтського ПЗ.

Иерархия	Индекс удобства поддержки	Сложность организации цикл...	Глубина наследов...	Взаимозависимость ...	Строки исходного кода
SolderingStation.DAL.Abstractions (Debug)	99	24	2	15	51
SolderingStation.DAL.Implementation (Debug)	68	64	2	84	983
SolderingStation.Entities (Debug)	98	41	2	7	61
SolderingStation.Hardware.Abstractions (Debug)	100	38	2	20	111
SolderingStation.Hardware.Implementation (Debug)	88	126	2	76	543
SolderingStation.Hardware.Models (Debug)	97	14	2	5	54
SolderingStationClient.BLL.Abstractions (Debug)	100	46	1	31	127
SolderingStationClient.BLL.Implementation (Debug)	85	161	3	116	966
SolderingStationClient.Models (Debug)	96	55	2	12	262
SolderingStationClient.Presentation (Debug)	90	299	15	186	1 628

Рисунок 3.1 – Аналіз метрик клієнтського додатку

З рисунку видно, що показник підтримки перевищує 80 для усіх проектів, крім реалізації шару доступу до даних, отож код є досить якісним. Глибина наслідування в презентаційному шарі пов’язана з високим рівнем наслідування UserControl із фреймоврку AvaloniaUI. Показник 68 для шару доступу до даних не є низьким через те, що в цьому шарі створено клас ContextSeed з методом розширення для початкового заповнення бази даних.

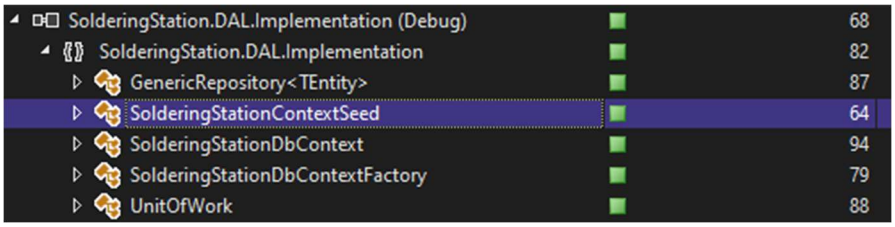
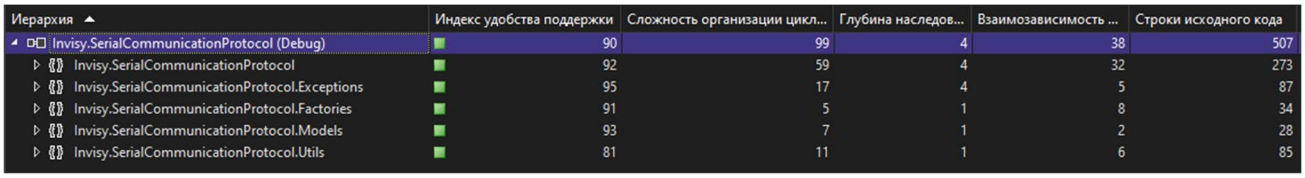


Рисунок 3.2 – Клас з середнім показником підтримуваності

Також окремо потрібно проаналізувати проаналізувати бібліотеку для обміну даними. На рисунку 3.3 можна побачити результати з аналізом бібліотеки для обміну даними. На ньому видно, що показники хороші, отже код – якісний.



Иерархия	Индекс удобства поддержки	Сложность организации цикл...	Глубина наследов...	Взаимозависимость ...	Строки исходного кода
Invisy.SerialCommunicationProtocol (Debug)	90	99	4	38	507
Invisy.SerialCommunicationProtocol	92	59	4	32	273
Invisy.SerialCommunicationProtocol.Exceptions	95	17	4	5	87
Invisy.SerialCommunicationProtocol.Factories	91	5	1	8	34
Invisy.SerialCommunicationProtocol.Models	93	7	1	2	28
Invisy.SerialCommunicationProtocol.Utils	81	11	1	6	85

Рисунок 3.3 – Аналіз метрик бібліотеки для обміну даними

3.2 Опис процесів тестування

Тестування програмного забезпечення є невід'ємною частиною життєвого циклу розробки ПЗ, оскільки без якісно проведеного тестування можливі помилки в роботі програми, що неодмінно призведуть до поганого користувацького досвіду та відтоку клієнтів неякісного програмного забезпечення. Під час проведення тестування оцінюється відповідність вимогам до програмного забезпечення, швидкість виконання, сумісність з різними операційними системами.

Тестування може бути автоматизованим, так і мануальним. При автоматизованому тестуванні використовуються спеціальні інструменти – фреймворки для проведення тестування. Вони дозволяють написати код тестування модуля або інтеграційної системи незалежно від інших модулів чи підсистем. Мануальне тестування, в свою чергу, проводиться вручну, без використання скриптів чи інших інструментів. Для проведення мануального тестування необхідно визначити стратегію тестування, описати тест-кейси. Після цього проводиться саме тестування, яке включає в себе введення тестових даних, покрокове виконання тесту, порівняння отриманих результатів з очікуваними, фіксацію дефектів, перевірку виправлень, документування результатів. З метою забезпечення стабільності та ефективності процесу тестування Майком Коном було вигадано піраміду

тестування, яку він описав в книзі «Succeeding With Agile. Software Development Using Scrum». Хоча спочатку в ній було 4 рівні, наразі всі знають тришарову модель цієї абстракції, яку наведено на рисунку 3.4. Основна ідея піраміди тестування полягає в тому, що слід мати більше базових тестів на нижній частині піраміди, що представляють широкий спектр недорогих і швидких тестів. По мірі руху вгору піраміди, кількість тестів зменшується, але вони стають складнішими та дорожчими.

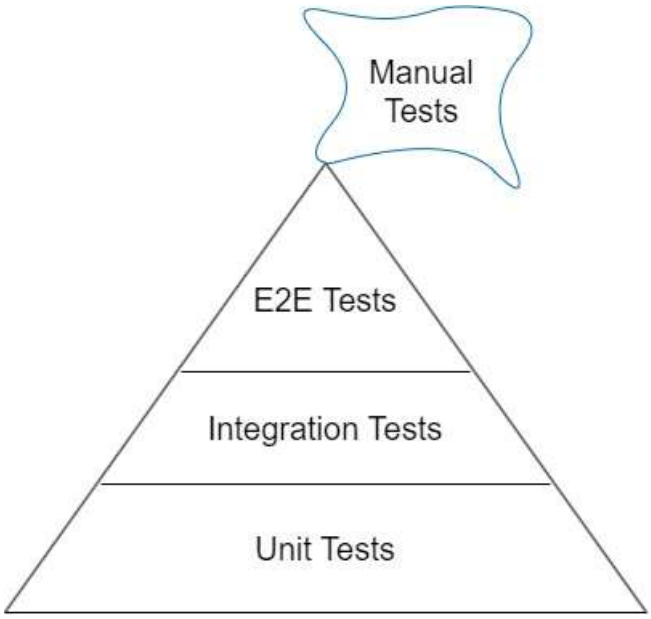


Рисунок 3.4 – Схема структурна піраміди тестування

Юніт-тестів зазвичай найбільше, оскільки вони невеликі, бо тестуються окремі методи класів. Для їх виконання потрібна найменша кількість часу, тобто, вони виконуються швидко відносно інших тестів.

Інтеграційні тести пишуть для тестування взаємодії між різними компонентами системи. Основна мета інтеграційних тестів полягає у виявленні проблем, які можуть виникнути при об'єднанні окремих частин системи та їх взаємодії.

E2E тести є дорогими, оскільки охоплюють тестування продукту від початку до кінця з позиції користувача. Вони симулюють реальний сценарій використання системи та перевіряють, чи відбувається взаємодія між компонентами належним чином. В рамках предметної області дипломної

роботи такий тип тестування не використовується, оскільки його досить складно впровадити.

В самому кінці проводиться мануальне тестування, якщо автоматизованих тестів було недостатньо, ці тести є найдорожчими, оскільки потребують прямого втручання людей, однак вони є найбільш надійними. При виконанні роботи було проведене мануальне тестування кожного з компонентів системи, опис тестів наведено у таблицях 3.1-3.13.

Таблиця 3.1 – Виявлення нового послідовного порту

Тест	Виявлення нового послідовного порту
Номер тесту	1.1
Початковий стан системи	Користувач запустив додаток
Вхідні дані	-
Опис проведення тесту	Користувач підключає USB-UART конвертер в USB порт, новий порт з’являється в списку.
Очікуваний результат	Користувач побачив новий порт в списку
Фактичний результат	Користувач побачив новий порт в списку

Таблиця 3.2 – Виявлення відключення послідовного порту

Тест	Виявлення відключення послідовного порту
Номер тесту	1.2
Початковий стан системи	Користувач запустив додаток та підключив USB-UART конвертер в USB порт, який відображається у програмі.
Вхідні дані	-
Опис проведення тесту	Користувач відключає USB-UART конвертер, порт зі списку зникає, при цьому не виникають ніякі помилки.

Кінець таблиці 3.2

Тест	Виявлення відключення послідовного порту
Очікуваний результат	Порт зник зі списку
Фактичний результат	Порт зник зі списку

Таблиця 3.3 – Підключення підтримуваного пристрою

Тест	Підключення підтримуваного пристрою
Номер тесту	1.3
Початковий стан системи	Користувач натиснув кнопку «Підключити».
Вхідні дані	Обраний правильний порт та налаштування порту
Опис проведення тесту	Після того, як користувач обрав порт з підтримуваним пристроєм, він натискає кнопку «Підключати»
Очікуваний результат	Пристрій з’явився у списку пристроїв, відображається на графіку температур
Фактичний результат	Пристрій з’явився у списку пристроїв, відображається на графіку температур

Таблиця 3.4 – Відключення пристрою

Тест	Відключення пристрою
Номер тесту	1.4
Початковий стан системи	Пристрій підключено
Вхідні дані	-
Опис проведення тесту	Користувач відключає пристрій кнопкою «Відключити».
Очікуваний результат	Пристрій зникне зі списку пристроїв, перестане відображатися на графіку температур, при цьому не виникне ніяких помилок.

Кінець таблиці 3.4

Тест	Відключення пристрою
Фактичний результат	Пристрій зник зі списку пристроїв, не відображається на графіку температур, помилок немає.

Таблиця 3.5 – Повторне підключення підтримуваного пристрою

Тест	Повторне підключення підтримуваного пристрою
Номер тесту	1.5
Початковий стан системи	Користувач щойно відключив пристрій.
Вхідні дані	-
Опис проведення тесту	Користувач підключає пристрій кнопкою «Підключити».
Очікуваний результат	Пристрій з’явився у списку пристроїв, відображається на графіку температур
Фактичний результат	Пристрій з’явився у списку пристроїв, відображається на графіку температур

Таблиця 3.6 – Зміна бажаної температури контролера

Тест	Зміна бажаної температури контролера
Номер тесту	1.6
Початковий стан системи	Пристрій підключено, обрано контролер температури
Вхідні дані	Бажана температура
Опис проведення тесту	Користувач вводить бажану температуру, натискає кнопку «Встановити».

Кінець таблиці 3.6

Тест	Зміна бажаної температури контролера
Очікуваний результат	Поле «Очікувана температура» відображає нову бажану температуру, поточна температура нагрівача починає наближатися до вказаного значення та зупиняється на ньому.
Фактичний результат	Поле «Очікувана температура» відображає нову бажану температуру, значення в полі «Поточна температура» починає наближатися до вказаного значення та зупиняється на ньому.

Таблиця 3.7 – Перегляд коефіцієнтів PID

Тест	Перегляд коефіцієнтів PID
Номер тесту	1.7
Початковий стан системи	Підключено пристрій з підтримкою PID
Вхідні дані	-
Опис проведення тесту	Користувач підключає пристрій з підтримкою PID. У списку пристроїв з’являється новий пристрій. У списку налаштувань контролера з’являється можливість переглянути та налаштувати PID коефіцієнти.
Очікуваний результат	В списку налаштувань контролера з’явилася можливість переглянути та змінити PID коефіцієнти.
Фактичний результат	В списку налаштувань контролера з’явилася можливість переглянути та змінити PID коефіцієнти.

Таблиця 3.8 – Зміна коефіцієнтів PID

Тест	Зміна коефіцієнтів PID
Номер тесту	1.8
Початковий стан системи	Підключено пристрій з підтримкою PID
Вхідні дані	-
Опис проведення тесту	Користувач підключає пристрій з підтримкою PID. У списку пристроїв з’являється новий пристрій. Користувач може змінити кожен з коефіцієнтів.
Очікуваний результат	Після зміни коефіцієнту, поле з відповідною назвою коефіцієнта змінюється. Після перепідключення пристрою результат зберігається.
Фактичний результат	Після зміни коефіцієнту, поле з відповідною назвою коефіцієнта змінюється. Після перепідключення пристрою результат зберігається.

Таблиця 3.9 – Відкриття вікна для управління термопрофілями

Тест	Відкриття вікна для управління термопрофілями
Номер тесту	1.9
Початковий стан системи	Запущено програму
Вхідні дані	-
Опис проведення тесту	Користувач натискає кнопку налаштувань термопрофілів.
Очікуваний результат	Відкривається нове вікно, в якому можна створити/видалити/відредагувати термопрофілі.

Кінець таблиці 3.9

Тест	Відкриття вікна для управління термопрофілями
Фактичний результат	Відкривається нове вікно, в якому можна створити/видалити/відредагувати термопрофілі.

Таблиця 3.10 – Додавання нового температурного профілю

Тест	Додавання нового температурного профілю
Номер тесту	1.10
Початковий стан системи	Відкрито вікно налаштування температурних профілів
Вхідні дані	-
Опис проведення тесту	Користувач натискає кнопку для створення нового температурного профілю.
Очікуваний результат	В списку температурних профілів з’явився ще один профіль.
Фактичний результат	В списку температурних профілів з’явився ще один профіль.

Таблиця 3.11 – Видалення температурного профілю

Тест	Видалення температурного профілю
Номер тесту	1.11
Початковий стан системи	Відкрито вікно налаштування температурних профілів
Вхідні дані	-
Опис проведення тесту	Користувач натискає кнопку для видалення температурного профілю.
Очікуваний результат	Температурний профіль було видалено
Фактичний результат	Температурний профіль було видалено

Таблиця 3.12 – Додавання нової кривої

Тест	Додавання нової кривої на температурний профіль
Номер тесту	1.12
Початковий стан системи	Відкрито вікно налаштування температурних профілів, обрано один з них
Вхідні дані	-
Опис проведення тесту	Користувач натискає кнопку для додавання нової кривої.
Очікуваний результат	Додано нову криву на графік та у список
Фактичний результат	Додано нову криву на графік та у список

Таблиця 3.13 – Запуск роботи за термопрофілем

Тест	Запуск роботи за температурним профілем
Номер тесту	1.13
Початковий стан системи	Відкрито вікно запуску роботи за температурним профілем.
Вхідні дані	-
Опис проведення тесту	Користувач обирає температурний профіль, необхідні для нього контролери, та запускає роботу.
Очікуваний результат	Програма переходить в стан роботи за температурним профілем.
Фактичний результат	Програма переходить в стан роботи за температурним профілем.

3.3 Опис контрольного прикладу

Після запуску програми користувач бачить перед собою вікно, яке зображено на рисунку 3.5. З лівої сторони вікна можна побачити блок відповідальний за підключення пристроїв. У списку на даний момент має бути видно усі послідовні порти комп’ютера.

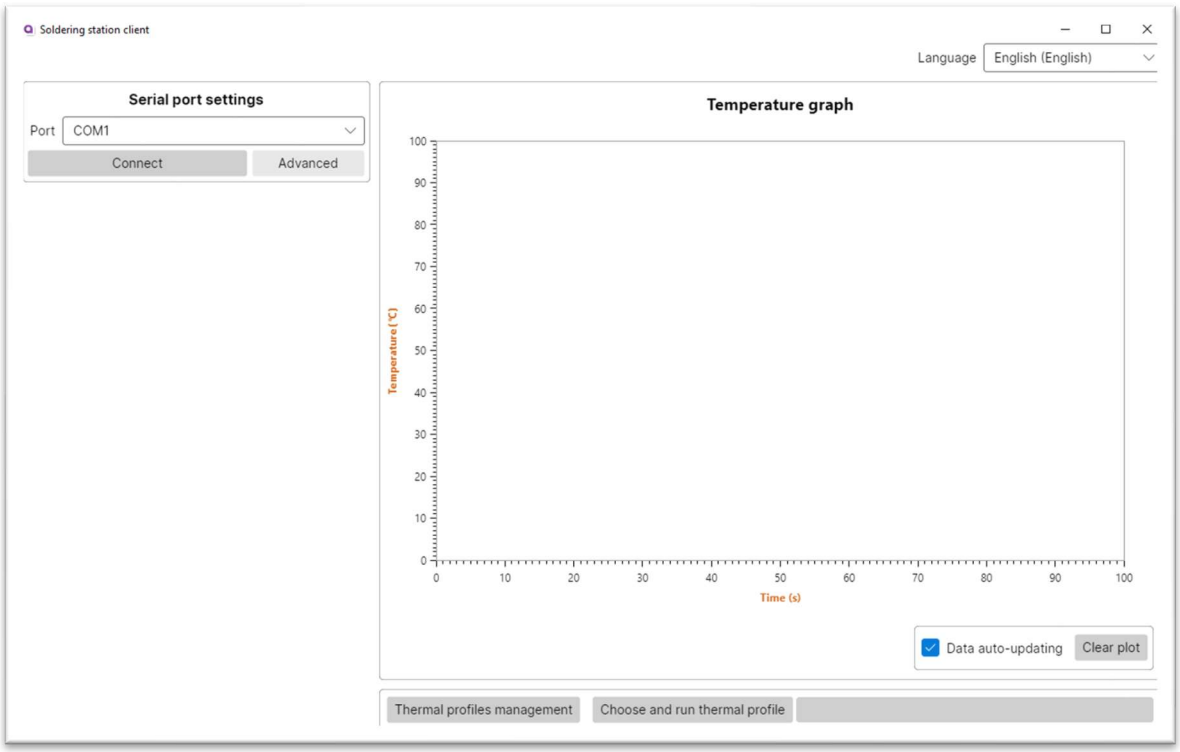


Рисунок 3.5 – Головне вікно програми

Користувач обирає порт та натискає кнопку «Connect». У випадку, якщо було обрано порт, до якого не підключений підтримуваний контролер, або не було виставлено правильні параметри порту для конкретного пристрою, то користувач через декілька секунд побачить вікно з помилкою, яке зображено на рисунку 3.6.

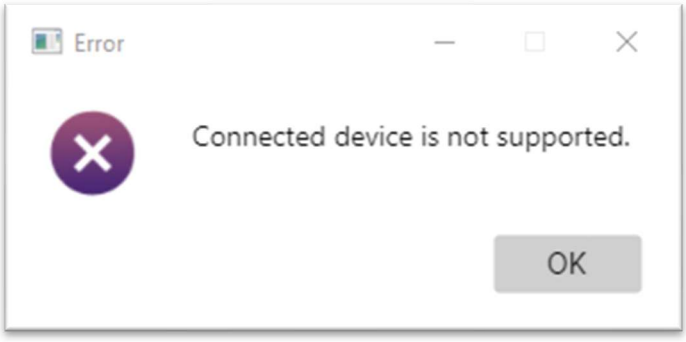


Рисунок 3.6 – Вікно з помилкою

Потім користувач програми може відкрити вікно для перевірки або зміни налаштувань порту, після зміни параметрів користувач натискає кнопку «Apply». Вікно налаштувань зображено на рисунку 3.7.

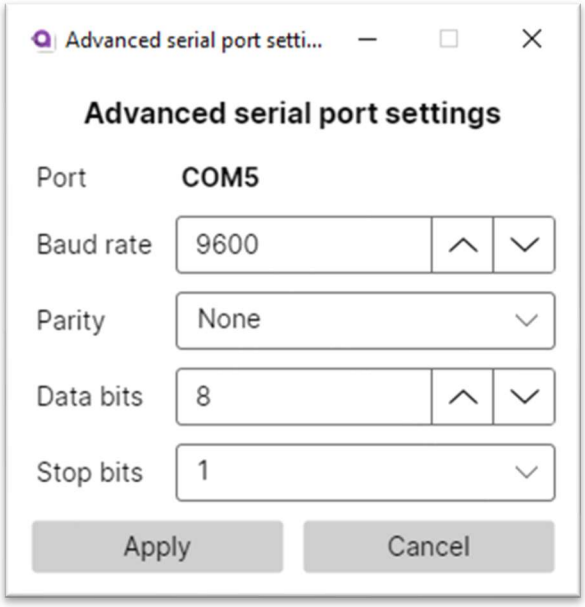


Рисунок 3.7 – Вікно налаштувань порту

Після збереження або відхилення зміни параметрів порту користувач знову повертається на головне вікно програми, де може спробувати підключитися ще раз. У разі успішної спроби, контролер відобразиться у списку пристроїв та на графіку, що можна побачити на рисунку 3.8.

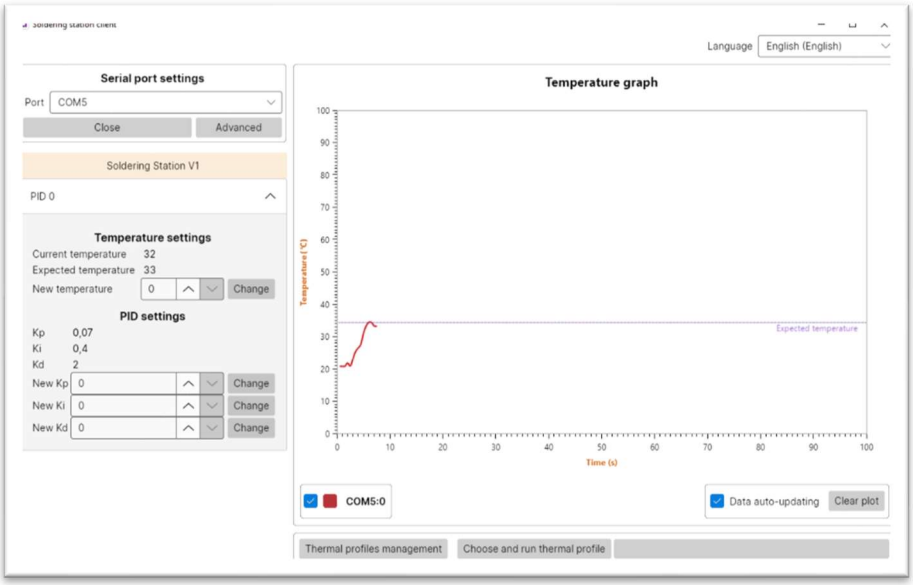


Рисунок 3.8 – Головне вікно з підключеним пристроєм

Тепер користувач може змінити температуру або PID коефіцієнти (якщо пристрій підтримує PID управління), команди на зміну одразу ж будуть виконані контролером. Також він може зупинити автоматичне оновлення даних з контролера, очистити графік або змінити колір на графіку.

Далі користувач може відкрити вікно для управління температурними профілями, натиснувши кнопку «Thermal profiles management». Вікно редактора температурних профілів зображено на рисунку 3.9. Якщо температурні профілі були вже створені раніше, то перший профіль зі списку автоматично завантажиться. В полі «Name» користувач може змінити назву температурного профілю. До моменту натискання кнопки «Save» зміни в базу даних не будуть зроблені, лише візуально. Кнопкою «New» користувач може створити новий температурний профіль або кнопкою «Remove» видалити вже готовий. Кнопками «Add» та «Remove» додає або видаляє відповідно одну з кривих. Крива створюється та редагується натисканням лівої кнопки миші по графіку. Якщо натиснути по вільному місцю, то буде створено нову точку кривої, яка обрана у лівій частині програми. Також можна змінити назву кривої, відредагувавши її в полі, або змінити її колір, натиснувши на кнопку з відповідним кольором. Після закінчення користувач може зберегти або відхилити зміни кнопками «Save» та «Close» відповідно.

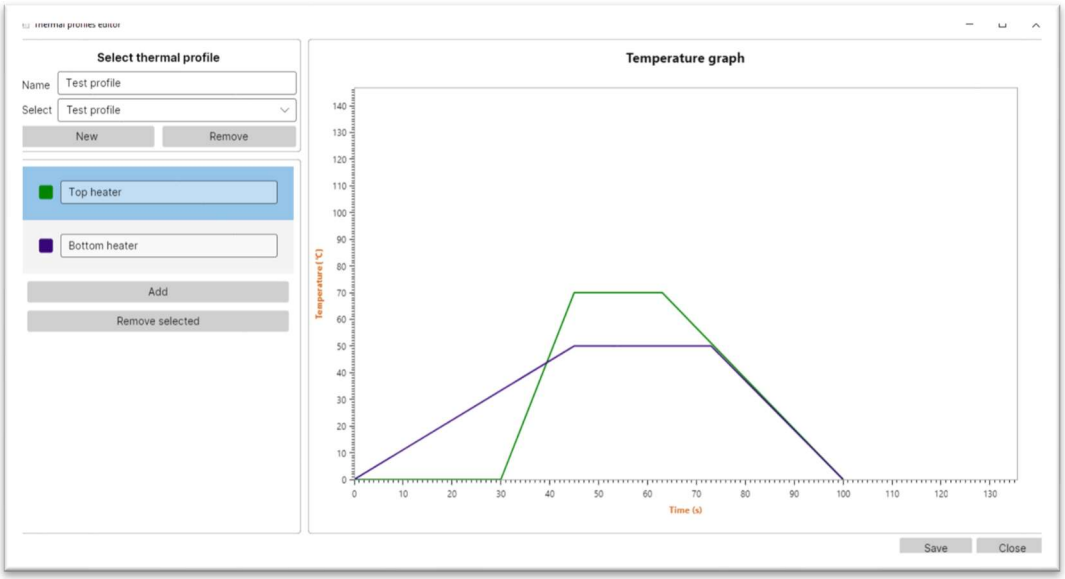


Рисунок 3.9 – Вікно редактора температурних профілів

Після того, як користувач знову повернувся до основного вікна, він може натиснути кнопку «Choose and run thermal profile». На рисунку 3.10 зображено вікно, яке з’являється після цього.

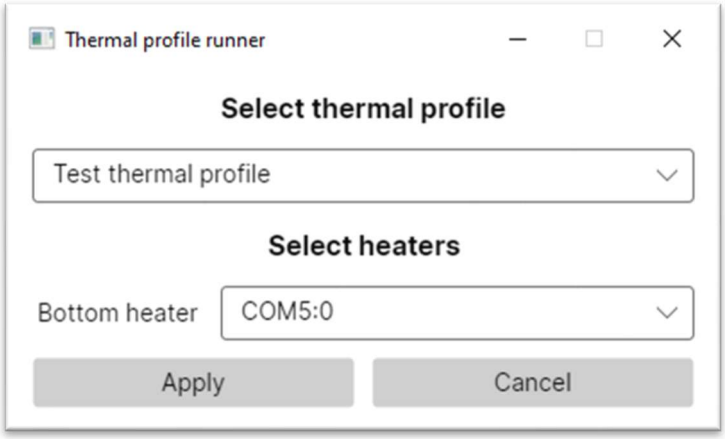


Рисунок 3.10 – Вікно налаштування роботи за температурним профілем

У вікні користувач обирає потрібний температурний профіль та прив’язує контролери до кривих, після чого може запустити процес роботи за температурним профілем. Якщо контролер не підключено, то запустити процес неможливо.

Висновки до розділу

В даному розділі описується процес аналізу якості і тестування програмного забезпечення. Було досліджено можливі варіанти тестування та обрано ті, які найкраще підходять в рамках предметної області. Були описані тестові кейси та проведено мануальне тестування, в результаті якого було підтверджено, що застосунок працює стабільно та без помилок.

4 ВПРОВАДЖЕННЯ ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Розгортання програмного забезпечення

Розгортання програмного забезпечення є важливим кроком у життєвому циклі розробки програмного забезпечення. Цей процес включає в себе підготовку, встановлення та налаштування додатку.

Розповсюдження клієнтської частини додатку відбувається через функцію «Releases» на платформі для контролю версій «GitHub». Для цього було інтегровано процеси CI/CD (Continuous integration/Continuous deployment), які дозволяють автоматично зібрати, протестувати та доставити програмне забезпечення. На рисунку 4.1 зображено робочий процес для збірки та релізу проєкту.

```yaml
---
name: "release"

on:
  push:
    branches:
      - "main"

jobs:
  release:
    name: "Release"
    runs-on: "ubuntu-latest"

    steps:
      - uses: actions/checkout@v2
      - name: Setup .NET
        uses: actions/setup-dotnet@v1
        with:
          dotnet-version: '6.0.x'
          include-prerelease: true

      - name: "Build & test"
        run: |
          cd SolderingStationClient.Presentation
          dotnet publish -r win-x64 -c Release /p:PublishSingleFile=true /p:IncludeNativeLibrariesForSelfExtract=true
          dotnet publish -r linux-x64 -c Release /p:PublishSingleFile=true /p:IncludeNativeLibrariesForSelfExtract=true
          echo "done!"

      - name: "Release"
        uses: "marvinpinto/action-automatic-releases@latest"
        with:
          repo_token: "${{ secrets.GITHUB_TOKEN }}"
          automatic_release_tag: "latest"
          prerelease: false
          title: "Release Build"
          files: |
            */bin/Release/*/win-x64/publish/SolderingStationClient.exe
            */bin/Release/*/linux-x64/publish/SolderingStationClient
```

Рисунок 4.1 – Робочий процес для збірки та релізу проєкту

Розповсюдження бібліотеки для обміну даними відбувається за допомогою пакетного менеджера Nuget. NuGet є пакетним менеджером для платформи .NET, що дозволяє легко встановлювати, оновлювати та керувати залежностями програмного забезпечення .NET. Він надає можливість зручного використання сторонніх бібліотек, фреймворків та інших компонентів, які можна використовувати в проектах .NET. На рисунку 4.2 зображено процес відправки готового пакету з бібліотекою на сервер.

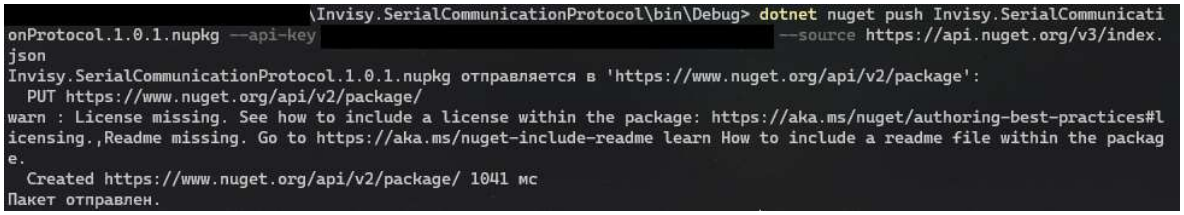


Рисунок 4.2 – Реліз нової версії nuget пакету

4.2 Підтримка програмного забезпечення

Забезпечення підтримки програмного забезпечення здійснюється з використанням робочого процесу «GitHub Actions», який був розглянутий раніше. Після того, як розробник здійснює зміни у головній гілці «main», цей сервіс автоматично проводить процес компіляції та тестування програми, а потім створює новий реліз. Потрібно відзначити, що цей метод не є бездоганним, але є досить поширеним серед розробників відкритого програмного забезпечення для розповсюдження та підтримки своїх десктопних додатків. В майбутньому, при зростанні популярності застосунку, доцільно розглянути інтеграцію можливості перевірки наявності нових версій прямо у програмі, що сприятиме максимальному поліпшенню користувацького досвіду. На рисунку 4.3 зображено сторінку, де можна завантажити програму як для операційної системи Linux, так і для Windows.

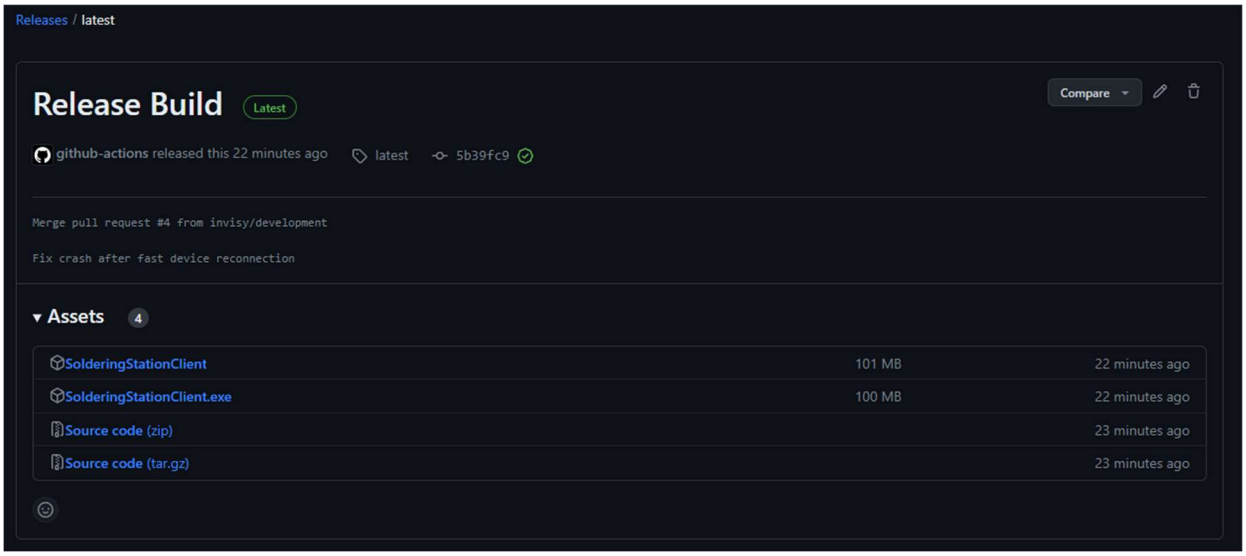


Рисунок 4.3 – Автоматично створений реліз на платформі GitHub

Для оновлення додатку користувач періодично перевіряє сторінку репозиторію та завантажує нову версію, якщо це необхідно. Після цього йому достатньо замінити виконуваний файл, запустити його, і одразу ж можна приступати до роботи.

Висновки до розділу

У даному розділі було розглянуто способи розгортання програмного забезпечення проєкту, також були описані вимоги до обладнання та системного програмного забезпечення. Було використано інструменти для забезпечення CI/CD процесів, щоб забезпечити підтримку продукту.

ВИСНОВКИ

Під час виконання дипломної роботи було розроблено усі частини програмного забезпечення, які забезпечують підтримку діяльності паяльної станції. Однією з частин є унікальний клієнтський додаток, який складається з системи управління контролерами, якого не має жоден аналог, зручного графічного редактору температурних профілів, а також сервісу для запуску роботи контролерів за температурними профілями. Для розкриття усіх можливостей програми було також спроєктовано та реалізовано повнофункціональний контролер паяльної станції. Здійснений проєкт включав розробку та успішне впровадження швидкої, надійної та зручної у використанні бібліотеки для обміну даними між власним контролером та клієнтським програмним забезпеченням. Ця бібліотека виявилась доречною та ефективною для вирішення завдань передачі даних, дозволяючи забезпечити швидкість передачі, стабільність та безпеку інформації.

Даний проєкт має великий потенціал для розширення функціоналу та є актуальним у сучасній промисловості. Зокрема, є кілька перспективних напрямків розвитку, які можна врахувати. По-перше, рекомендується реалізувати систему для автоматичного налаштування та калібрування температурних контролерів. Це дозволить значно спростити процес початкового налаштування та забезпечити оптимальну роботу контролера з мінімальними зусиллями з боку користувача. По-друге, варто розглянути можливість реалізації функції експорту та аналізу даних, отриманих під час роботи програми. Це дозволить користувачам отримати цінну інформацію та провести детальний аналіз, що сприятиме удосконаленню процесу регулювання температури та покращенню його точності. Крім того, рекомендується реалізувати функцію виявлення аномалій в роботі контролера. Це дозволить оперативно виявляти відхилення від нормального функціонування контролера або неправильну роботу температурних датчиків, що допоможе уникнути можливих поломок та зберегти обладнання від

пошкоджень. Всі ці напрямки розвитку дозволять підвищити функціональність та корисність проєкту, а також покращити ефективність його використання у промисловому середовищі.

У розділі «Аналіз вимог до програмного забезпечення» було проведено детальне дослідження предметної області, визначено вимоги до програмного забезпечення та сформульовано задачі, які воно має вирішувати. Також у ньому було обрано найоптимальніші алгоритми, технічні засоби, інструменти та мови програмування.

У розділі «Моделювання та аналіз вимог до програмного забезпечення» було створено моделі усіх частин програмного забезпечення, описано бізнес процеси, архітектуру та алгоритми роботи як клієнтської програми, так і бібліотеки для обміну даними та контролера паяльної станції.

Проведене тестування підтвердило стабільну роботу всього програмного забезпечення для підтримки діяльності паяльної станції. Всі функції системи працюють бездоганно, а їх поведінка є передбачуваною.

Розділ «Розгортання та підтримка» містить опис процесів розгортання та підтримки розробленого програмного забезпечення, що дозволяє користувачам ефективно впроваджувати та підтримувати систему.

Наукова новизна даної роботи полягає в розробці архітектури клієнтського програмного забезпечення, яка забезпечує зручну інтеграцію різних типів контролерів з різними інтерфейсами підключення, а також простоту розширення функціональних можливостей програми.

Загалом, результати дипломної роботи підтверджують успішну реалізацію поставлених цілей, що включають створення програмного забезпечення для контролера паяльної станції, універсального клієнтського програмного забезпечення та розробку бібліотеки для передачі даних. Робота має практичне значення для українських сервісних центрів, які можуть використовувати розроблену систему для виконання складних ремонтів без значних витрат на придбання іноземного паяльного обладнання.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1) J. Gao, Y. Wu, H. Ding, N. Wan, (2008) “Thermal profiling: reflow process based on the heating factor”, Soldering &Surface Mount Technology. Vol. 20, №4. P. 20 – 27. DOI: 10.1108/09540910810902679.

2) K. J. Astrom, T. Hagglund. PID Controllers: Theory, Design, and Tuning – с.64.

3) Офіційна докупентація для мови програмування C#. URL: https://learn.microsoft.com/en-us/dotnet/csharp (дата звернення: 30.05.2023).

4) Mark J. Price. C# 11 and .NET 7 - Modern Cross-Platform Development Fundamentals, 7th Edition – с.312.

5) Кросплатформний графічний фреймфорк Avalonia UI. URL: https://avaloniaui.net (дата звернення: 08.05.2023).

6) Інтегроване середовище розробки «Rider». URL: https://www.jetbrains.com/rider (дата звернення: 10.05.2023).

7) Fowler, Martin. UML distilled: a brief guide to the standard object modeling language. Addison-Wesley Professional, 2004. – с.103.

8) Joe Pardue. C Programming for Microcontrollers – с.12.

9) Офіційна документація до протоколу Modbus. С. 2-16. URL: https://www.modbus.org/docs/PI_MBUS_300.pdf (дата звернення: 30.04.2023).

10) Mark Richards. Software Architecture Patterns. Understanding Common Architectural Styles and When to Use Them – с.1-9.

11) Martin Fowler, Dave Rice, Matthew Foemmel, Edward Hieatt, Robert Mee, and Randy Stafford. Patterns of Enterprise Application Architecture – с.166.

12) Michael Owens. The Definitive Guide to SQLite – с.1-17.

13) Julia Lerman, Rowan Miller. Programming Entity Framework: Code First – с.4.

ДОДАТОК А ЗВІТ ПОДІБНОСТІ

UNICHECK

Ім'я користувача:
Лісовиченко Олег Іванович

ID перевірки:
1015410541

Дата перевірки:
04.06.2023 11:35:42 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
04.06.2023 11:58:37 EEST

ID користувача:
76913

Назва документа: **ІТ-91_Власюк_ПЗ**

Кількість сторінок: **64** Кількість слів: **10094** Кількість символів: **80300** Розмір файлу: **2.82 MB** ID файлу: **1015073842**

3.8%
Схожість

Найбільша схожість: **1.5%** з джерелом з Бібліотеки (ID файлу: **1015073853**)

1.55% Джерела з Інтернету	80	Сторінка 66
3.68% Джерела з Бібліотеки	272	Сторінка 67

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Замінені символи	1

ДОДАТОК Б ЕЛЕКТРИЧНА СХЕМА КОНТРОЛЕРА СТАНЦІЇ

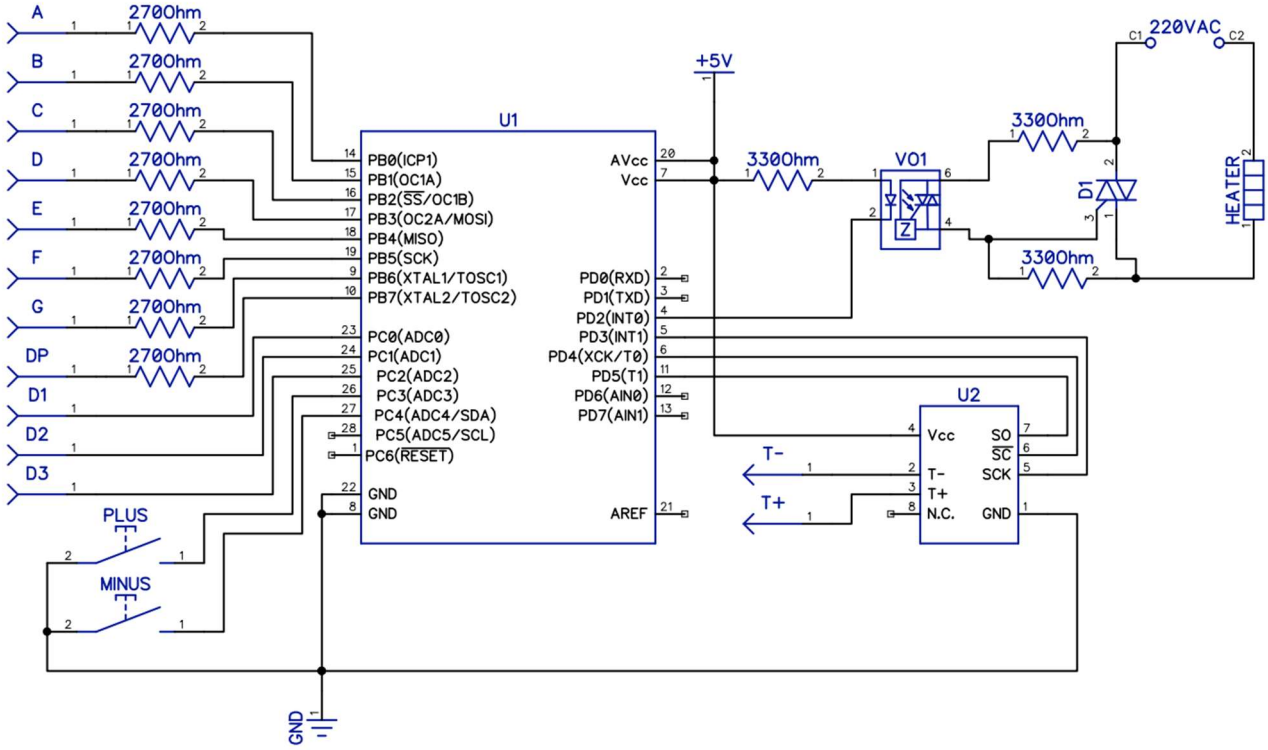


Рисунок Б.1 – Електрична схема контролера станції

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2023 р.

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ ПІДТРИМКИ ДІЯЛЬНОСТІ ПАЯЛЬНОЇ СТАНЦІЇ

Текст програми

КПІ.ІТ-8105в.045490.03.12

“ПОГОДЖЕНО”
Керівник проєкту:
_____ Олена ХАЛУС

Нормоконтроль:
_____ Максим ГОЛОВЧЕНКО

Виконавець:
_____ Владислав ВЛАСЮК

Київ – 2023

Файл ProfileEntity.cs

```
 public class ProfileEntity : BaseEntity<uint>
{
    //For EF
    public ProfileEntity()
    {

    }

    public ProfileEntity(string name, uint languageId)
    {
        Name = name;
        LanguageId = languageId;
    }

    public string Name { get; set;}
    public uint LanguageId { get; set; }
    public LanguageEntity Language { get; set;}
    public IEnumerable<ThermalProfileEntity> ThermalProfiles { get; set;}
    public IEnumerable<SerialConnectionParametersEntity>
SerialConnectionsParameters { get; set; }
}
```

Файл LanguageEntity.cs

```
public class LanguageEntity : BaseEntity<uint>
{
    //For EF
    public LanguageEntity() {    }

    public LanguageEntity(string englishName, string nativeName, string code)
    {
        EnglishName = englishName;
        NativeName = nativeName;
        Code = code;
    }

    public string EnglishName { get; set; }
    public string NativeName { get; set;}
    public string Code { get; set;}
}
```

Файл SerialConnectionParametersEntity.cs

```
public class SerialConnectionParametersEntity : BaseEntity<uint>
{
    public uint ProfileId { get; set; }
    public string PortName { get; set; } = string.Empty;
    public int BaudRate { get; set; }
    public int Parity { get; set; }
    public int DataBits { get; set; }
    public int StopBits { get; set; }


    //For EF
```

```
    public SerialConnectionParametersEntity() {    }

    public SerialConnectionParametersEntity(uint profileId, string portName,
int baudRate, int parity, int dataBits, int stopBits)
    {
        ProfileId = profileId;
        PortName = portName;
        BaudRate = baudRate;
        Parity = parity;
        DataBits = dataBits;
        StopBits = stopBits;
    }
}
```

Файл TemperatureMeasurementPointEntity.cs

```
public class TemperatureMeasurementPointEntity : BaseEntity<uint>
{
    public float SecondsElapsed { get; set; }
    public ushort Temperature { get; set; }

    //For EF
    public TemperatureMeasurementPointEntity() { }

    public TemperatureMeasurementPointEntity(uint id, float secondsElapsed,
ushort temperature)
    {
        Id = id;
        SecondsElapsed = secondsElapsed;
        Temperature = temperature;
    }
}
```

Файл ThermalProfileEntity.cs

```
public class ThermalProfileEntity : BaseEntity<uint>
{
    public string Name { get; set; }
    public IEnumerable<ThermalProfilePartEntity> Parts { get; set; }
    public uint ProfileId { get; set; }

    //For EF
```

```
    public ThermalProfileEntity() { }

    public ThermalProfileEntity(uint id, string name,
IEnumerable<ThermalProfilePartEntity> parts, uint profileId)
    {
        Id = id;
        Name = name;
        Parts = parts;
        ProfileId = profileId;
    }
}
```

Файл ThermalProfilePartEntity.cs

```
public class ThermalProfilePartEntity : BaseEntity<uint>
{
    public string Name { get; set; }
    public int Color { get; set; }
    public IEnumerable<TemperatureMeasurementPointEntity> TemperatureCurve {
get; set; }

    //For EF
    public ThermalProfilePartEntity() { }

    public ThermalProfilePartEntity(uint id, string name, int color,
IEnumerable<TemperatureMeasurementPointEntity> measurements)
    {
        Id = id;
        Name = name;
        Color = color;
        TemperatureCurve = measurements;
    }
}
```

Файл SolderingStationDbContext.cs

```
public class SolderingStationDbContext : DbContext
{
    #pragma warning disable CS8618 // Required by Entity Framework to
suppress warnings

    private readonly string _dbPath;
```

```
    public DbSet<ProfileEntity> Profiles { get; set; }
    public DbSet<LanguageEntity> Languages { get; set; }
    public DbSet<ThermalProfileEntity> ThermalProfiles { get; set; }
    public DbSet<ThermalProfilePartEntity> ThermalProfileParts { get; set; }
    public DbSet<TemperatureMeasurementPointEntity>
TemperatureMeasurementPoints { get; set; }
    public DbSet<SerialConnectionParametersEntity>
SerialConnectionsParameters { get; set; }

    public
SolderingStationDbContext(DbContextOptions<SolderingStationDbContext>
options) : base(options) {}

    public SolderingStationDbContext(string dbPath)
    {
        _dbPath = dbPath;
    }

    protected override void OnModelCreating(ModelBuilder builder)
    {
        base.OnModelCreating(builder);

builder.ApplyConfigurationsFromAssembly(Assembly.GetExecutingAssembly());
    }

    protected override void OnConfiguring(DbContextOptionsBuilder
optionsBuilder)
    {
        optionsBuilder.UseSqlite($"Data Source={_dbPath}");

optionsBuilder.UseQueryTrackingBehavior(QueryTrackingBehavior.NoTracking);
    }
}
```

Файл SolderingStationDbContextFactory.cs

```
public class SolderingStationDbContextFactory :
IDesignTimeDbContextFactory<SolderingStationDbContext>
{
    public SolderingStationDbContext CreateDbContext(string[] args)
    {
```

```
        var optionsBuilder = new
DbContextOptionsBuilder<SolderingStationDbContext>();
        optionsBuilder.UseSqlite("Data Source=test.db");

        return new SolderingStationDbContext(optionsBuilder.Options);
    }
}
```

Файл SolderingStationDbContextSeed.cs

```
public static class SolderingStationContextSeed
{
    public static void Seed(this SolderingStationDbContext dbContext)
    {
        if (dbContext.Database.IsSqlite())
        {
            dbContext.Database.Migrate();
        }

        if (!dbContext.Languages.Any())
        {
            dbContext.Languages.AddRange(
                new LanguageEntity("English", "English", "en"),
                new LanguageEntity("Ukrainian", "Українська", "uk")
                );
            dbContext.SaveChanges();
        }

        if (!dbContext.Profiles.Any())
        {
            dbContext.Profiles.Add(new ProfileEntity("Main", 1));
            dbContext.SaveChanges();
        }

        if (!dbContext.ThermalProfiles.Any())
        {
            var thermalProfile = new ThermalProfileEntity(1, "TestProfile",
new[]
            {
                new ThermalProfilePartEntity(0, "BottomHeater",
Color.Green.ToArgb(), new[]
                {
                    new TemperatureMeasurementPointEntity(0, 0, 0),
```

```
                    new TemperatureMeasurementPointEntity(0, 17, 5),
                    new TemperatureMeasurementPointEntity(0, 25, 40)
                }),
                new ThermalProfilePartEntity(0, "TopHeater",
Color.Blue.ToArgb(), new[]
                {
                    new TemperatureMeasurementPointEntity(0, 0, 0),
                    new TemperatureMeasurementPointEntity(0, 9, 40),
                })
            }, 1);

            dbContext.ThermalProfiles.Add(thermalProfile);

            dbContext.SaveChanges();
        }
    }
}
```

Файл UnitOfWork.cs

```
public class UnitOfWork : IUnitOfWork
{
    private readonly SolderingStationDbContext _context;
    private IDbContextTransaction? _transaction;

    public IGenericRepository<ProfileEntity> ProfilesRepository { get; }
    public IGenericRepository<LanguageEntity> LanguagesRepository { get; }
    public IGenericRepository<SerialConnectionParametersEntity>
SerialConnectionParametersRepository { get; }
    public IGenericRepository<ThermalProfileEntity> ThermalProfilesRepository
{ get; }
    public IGenericRepository<ThermalProfilePartEntity>
ThermalProfilePartsRepository { get; }
    public IGenericRepository<TemperatureMeasurementPointEntity>
TemperatureMeasurementPointsRepository { get; }

    public UnitOfWork(SolderingStationDbContext dbContext,
        IGenericRepository<ProfileEntity> profilesRepository,
        IGenericRepository<LanguageEntity> languagesRepository,
        IGenericRepository<SerialConnectionParametersEntity>
serialConnectionParametersRepository,
        IGenericRepository<ThermalProfileEntity> thermalProfilesRepository,
```

```
        IGenericRepository<ThermalProfilePartEntity>
thermalProfilePartsRepository,
        IGenericRepository<TemperatureMeasurementPointEntity>
temperatureMeasurementPointsRepository)
    {
        _context = dbContext;
        ProfilesRepository = profilesRepository;
        LanguagesRepository = languagesRepository;
        SerialConnectionParametersRepository =
serialConnectionParametersRepository;
        ThermalProfilesRepository = thermalProfilesRepository;
        ThermalProfilePartsRepository = thermalProfilePartsRepository;
        TemperatureMeasurementPointsRepository =
temperatureMeasurementPointsRepository;
    }

    public async Task SaveChanges()
    {
        try
        {
            int result = await _context.SaveChangesAsync();

            if (result == 0)
                throw new DatabaseException($"There were no changes applied
to database during saving changes task!");
        }
        catch (Exception e) when (e is not DatabaseException)
        {
            throw new DatabaseException($"Exception while applying changes to
database!", e);
        }
    }

    public async Task BeginTransaction()
    {
        if(_transaction != null)
            throw new DatabaseException($"There is an uncommitted
transaction!");

        _transaction = await _context.Database.BeginTransactionAsync();
    }

    public async Task TransactionCommit()
```

```
    {
        if(_transaction == null)
            throw new DatabaseException($"The transaction can not be
committed because it has not been started!");

        await _transaction.CommitAsync();
        await _transaction.DisposeAsync();
    }

    public async Task TransactionRollback()
    {
        if (_transaction == null)
            throw new DatabaseException($"The transaction can not be rolled
back because it has not been started!");

        await _transaction.RollbackAsync();
        await _transaction.DisposeAsync();
        _transaction = null;
    }
```

Файл GenericRepository.cs

```
public class GenericRepository<TEntity> : IGenericRepository<TEntity> where
TEntity : BaseEntity<uint>
{
    private readonly SolderingStationDbContext _dbContext;
    private readonly ISpecificationEvaluator _specificationEvaluator;

    public GenericRepository(SolderingStationDbContext dbContext)
    {
        _dbContext = dbContext;
        _specificationEvaluator = SpecificationEvaluator.Default;
    }

    public GenericRepository(SolderingStationDbContext dbContext,
        ISpecificationEvaluator specificationEvaluator)
    {
        _dbContext = dbContext;
        _specificationEvaluator = specificationEvaluator;
    }

    public virtual TEntity Add(TEntity entity)
```

```
    {
        _dbContext.Set<TEntity>().Add(entity);
        return entity;
    }

    public virtual async Task<TEntity?> GetByIdAsync(uint id)
    {
        return await _dbContext.Set<TEntity>().FindAsync(new object[] { id
});
    }

    public async Task<TEntity?> GetBySpecAsync(ISpecification<TEntity>
specification)
    {
        return await ApplySpecification(specification).FirstOrDefaultAsync();
    }

    public async Task<TResult?>
GetBySpecAsync<TResult>(ISpecification<TEntity, TResult> specification)
    {
        return await ApplySpecification(specification).FirstOrDefaultAsync();
    }

    public virtual async Task<IList<TEntity>> GetListAsync()
    {
        return await _dbContext.Set<TEntity>().ToListAsync();
    }

    public async Task<IList<TEntity>>
GetListBySpecAsync(ISpecification<TEntity> specification)
    {
        var queryResult = await
ApplySpecification(specification).ToListAsync();
        return specification.PostProcessingAction == null ? queryResult :
specification.PostProcessingAction(queryResult).ToList();
    }

    public async Task<IList<TResult>>
GetListBySpecAsync<TResult>(ISpecification<TEntity, TResult> specification)
    {
        var queryResult = await
ApplySpecification(specification).ToListAsync();
```

```
        return specification.PostProcessingAction == null ? queryResult :
specification.PostProcessingAction(queryResult).ToList();
    }

    public virtual void Update(TEntity entity)
    {
        _dbContext.Set<TEntity>().Update(entity);
    }

    public virtual void Delete(TEntity entity)
    {
        _dbContext.Set<TEntity>().Remove(entity);
    }

    public virtual void Delete(uint id)
    {
        var selected = _dbContext.Set<TEntity>().FirstOrDefault(entity =>
entity.Id == id);
        if (selected == null)
            throw new ArgumentException(nameof(id));
        _dbContext.Set<TEntity>().Remove(selected);
    }

    protected virtual IQueryable<TEntity>
ApplySpecification(ISpecification<TEntity> specification, bool
evaluateCriteriaOnly = false)
    {
        return
_specificationEvaluator.GetQuery(_dbContext.Set<TEntity>().AsQueryable(),
specification, evaluateCriteriaOnly);
    }

    protected virtual IQueryable<TResult>
ApplySpecification<TResult>(ISpecification<TEntity, TResult> specification)
    {
        return
_specificationEvaluator.GetQuery(_dbContext.Set<TEntity>().AsQueryable(),
specification);
    }
}
```

Файл ProfileConfiguration.cs

```
public class ProfileConfiguration : IEntityTypeConfiguration<ProfileEntity>
{
    public void Configure(EntityTypeBuilder<ProfileEntity> builder)
    {
        builder.HasKey(entity => entity.Id);

        builder.Property(entity => entity.Name)
            .IsRequired()
            .HasMaxLength(100);

        builder.HasOne(entity =>
entity.Language).WithMany().HasForeignKey(entity => entity.LanguageId)
            .IsRequired();


builder.HasMany<ThermalProfileEntity>().WithOne().HasForeignKey(entity =>
entity.ProfileId)
            .IsRequired();


builder.HasMany<SerialConnectionParametersEntity>().WithOne().HasForeignKey(e
ntity => entity.ProfileId)
            .IsRequired();
    }
}
```

Файл DeviceManager.cs

```
public class DeviceManager : IDeviceManager
{
    private readonly ConcurrentDictionary<ulong, IDevice> _connectedDevices =
new();
    private ulong _lastId;

    public string GetDeviceNameById(ulong id)
    {
        return _connectedDevices[id].Name;
    }

    public IEnumerable<ulong> GetAllDeviceIds()
    {
```

```
            return _connectedDevices.Keys;
        }

        public IEnumerable<CapabilityHandle<T>> GetByDeviceCapability<T>() where
    T : class, IDeviceCapability
        {
            foreach (var deviceKey in _connectedDevices.Keys)
                if (_connectedDevices[deviceKey] is T)
                    yield return new CapabilityHandle<T>(deviceKey,
    (T)_connectedDevices[deviceKey]);
        }

        public string GetConnectionNameByDeviceId(ulong id)
        {
            return _connectedDevices[id].Connection.Name;
        }

        public bool IsDeviceCapabilitySupported<T>(ulong id) where T :
    IDeviceCapability
        {
            _connectedDevices.TryGetValue(id, out var device);
            return device is T;
        }

        public T? TryGetDeviceCapability<T>(ulong id) where T : IDeviceCapability
        {
            var capabilityInterface = default(T);

            _connectedDevices.TryGetValue(id, out var device);

            if (device is T value)
                capabilityInterface = value;

            return capabilityInterface;
        }

        public IEnumerable<CapabilityHandle<T>> GetByConnectionCapability<T>()
    where T : class, IConnectionCapability
        {
            foreach (var deviceKey in _connectedDevices.Keys)
                if (_connectedDevices[deviceKey].Connection is T)
                    yield return new CapabilityHandle<T>(deviceKey,
    (T)_connectedDevices[deviceKey].Connection);
```

```
    }

    public bool IsConnectionCapabilitySupported<T>(ulong id) where T :
IConnectionCapability
    {
        _connectedDevices.TryGetValue(id, out var device);
        return device?.Connection is T;
    }

    public T? TryGetConnectionCapability<T>(ulong id) where T :
IConnectionCapability
    {
        var capabilityInterface = default(T);

        _connectedDevices.TryGetValue(id, out var device);

        if (device?.Connection is T value)
            capabilityInterface = value;

        return capabilityInterface;
    }

    public ulong AddDevice(IDevice device)
    {
        var id = Interlocked.Add(ref _lastId, 1);

        if (!_connectedDevices.TryAdd(id, device))
            throw new ArgumentException("Unable to add connection.",
nameof(device));

        DevicePresenceChanged?.Invoke(this, new
DevicePresenceChangedEventArgs<ulong>(id, PresenceStatus.Connected));
        return id;
    }

    public void RemoveDevice(ulong id)
    {
        if (!_connectedDevices.TryRemove(id, out var device))
            throw new ArgumentException("Unable to remove connection.",
nameof(id));

        device.CloseConnection();
```

```
        DevicePresenceChanged?.Invoke(this, new
DevicePresenceChangedEventArgs<ulong>(id, PresenceStatus.Disconnected));
    }

    public event EventHandler<DevicePresenceChangedEventArgs<ulong>>?
DevicePresenceChanged;
}
```

Файл HardwareDetector.cs

```
public class HardwareDetector<T> : IHardwareDetector<T> where T :
IConnectionParameters
{
    private readonly IDeviceManager _deviceManager;
    private readonly IEnumerable<IDeviceFactory<T>>
_supportedDeviceFactories;

    public HardwareDetector(IDeviceManager deviceManager,
IEnumerable<IDeviceFactory<T>> supportedDeviceFactories)
    {
        _deviceManager = deviceManager;
        _supportedDeviceFactories = supportedDeviceFactories;
    }

    public async Task<ulong> ConnectDeviceWithIdentification(T parameters)
    {
        foreach (var deviceFactory in _supportedDeviceFactories)
        {
            var device = deviceFactory.Create(parameters);
            var isSuccessfullyConnected = device.Connect() && await
device.Probe();

            if (isSuccessfullyConnected)
            {
                var id = _deviceManager.AddDevice(device);
                return id;
            }

            device.Dispose();
        }

        throw new UnsupportedDeviceException();
    }
```

```
}
```

Файл SelfBuiltDeviceV1.cs

```
public class
    SelfBuiltDeviceV1<TConnectionParameters, TConnection> :
ISelfBuiltDeviceV1<TConnectionParameters, TConnection>
    where TConnectionParameters : IConnectionParameters
    where TConnection : IConnectionGeneric<TConnectionParameters>,
ISelfBuiltDeviceCommandProcessor
{
    private readonly TConnection _connection;
    private bool _disposed;

    public IConnection Connection => _connection;

    public SelfBuiltDeviceV1(TConnection connection)
    {
        _connection = connection;
    }

    public string Name => "Soldering Station V1";

    public bool Connect()
    {
        return _connection.Connect();
    }

    public virtual async Task<bool> Probe()
    {
        try
        {
            var channels = await GetChannelsNumber();
            return true;
        }
        catch (Exception e)
        {
            return false;
        }
    }

    public void CloseConnection()
    {
```

```
        _connection.Disconnect();
    }

    public void Dispose()
    {
        if (_disposed)
            return;

        _connection.Dispose();
        _disposed = true;
        GC.SuppressFinalize(this);
    }

    public async Task<byte> GetChannelsNumber()
    {
        var command = new GetChannelsNumberCommand();
        var result = await _connection.ExecuteCommand(command);

        return result.ChannelsNumber;
    }

    public async Task<ushort> GetCurrentTemperature(byte channel)
    {
        var command = new GetCurrentTemperatureCommand(channel);
        var result = await _connection.ExecuteCommand(command);

        return result.Temperature;
    }

    public async Task<ushort> GetDesiredTemperature(byte channel)
    {
        var command = new GetDesiredTemperatureCommand(channel);
        var result = await _connection.ExecuteCommand(command);

        return result.Temperature;
    }

    public async Task SetDesiredTemperature(byte channel, ushort temperature)
    {
        var command = new SetDesireTemperatureCommand(channel, temperature);
        await _connection.ExecuteCommand(command);
    }
```

```
    public async Task<PidCoefficients> GetCoefficients(byte channel)
    {
        var command = new GetPidCoefficientsCommand(channel);
        var commandResult = await _connection.ExecuteCommand(command);

        var result = new PidCoefficients(commandResult.Kp, commandResult.Ki,
commandResult.Kd);

        return result;
    }

    public async Task SetKpCoefficient(byte channel, float coefficient)
    {
        var command = new SetKpCoefficientCommand(channel, coefficient);
        await _connection.ExecuteCommand(command);
    }

    public async Task SetKiCoefficient(byte channel, float coefficient)
    {
        var command = new SetKiCoefficientCommand(channel, coefficient);
        await _connection.ExecuteCommand(command);
    }

    public async Task SetKdCoefficient(byte channel, float coefficient)
    {
        var command = new SetKdCoefficientCommand(channel, coefficient);
        await _connection.ExecuteCommand(command);
    }
}
```

Файл SelfBuiltDeviceSerialConnection.cs

```
public class SelfBuiltDeviceSerialConnection :
IConnectionGeneric<SerialConnectionParameters>,
    ISelfBuiltDeviceCommandProcessor
{
    private readonly ISerialCommunicationClient _client;
    private bool _disposed;

    public string Name => _client.PortSettings.PortName;

    public SelfBuiltDeviceSerialConnection(ISerialCommunicationClient client,
```

```
        SerialConnectionParameters connectionParameters)
    {
        ConnectionParameters = connectionParameters;
        _client = client;
    }

    public SerialConnectionParameters ConnectionParameters { get; }

    public bool Connect()
    {
        try
        {
            _client.Connect();
        }
        catch (ConnectionException)
        {
            return false;
        }
        catch (Exception)
        {
            //TODO log exception
        }

        return true;
    }

    public void Disconnect()
    {
        _client.Close();
    }

    public void Dispose()
    {
        if (_disposed)
            return;

        Disconnect();
        _disposed = true;
        GC.SuppressFinalize(this);
    }

    public async Task
ExecuteCommand<TRequest>(ISelfBuiltDeviceCommand<TRequest> command)
```

```
        where TRequest : struct
    {
        await _client.ExecuteCommandAsync(command.Command,
command.ParamsRequest);
    }

    public async Task<TResponse> ExecuteCommand<TRequest, TResponse>(
        ISelfBuiltDeviceCommand<TRequest, TResponse> command)
        where TRequest : struct where TResponse : struct
    {
        //TODO rethrow exception
        var result = await _client.ExecuteCommandAsync<TRequest,
TResponse>(command.Command, command.ParamsRequest);

        return result;
    }
}
```

Файл CommandsEnum.cs

```
public enum CommandsEnum : ushort
{
    GetPidsNumber = 1,
    GetCurrentTemperature,
    GetExpectedTemperature,
    GetPidCoefficients,
    SetExpectedTemperature,
    SetKpCoefficient,
    SetKiCoefficient,
    SetKdCoefficient
}
```

Файл GetCurrentTemperatureCommand.cs

```
[StructLayout(LayoutKind.Sequential, Pack = 1)]
public record struct GetCurrentTemperatureCommandParameters(byte PidId);

[StructLayout(LayoutKind.Sequential, Pack = 1)]
public record struct GetCurrentTemperatureCommandResult(ushort Temperature);

public class
```

```
    GetCurrentTemperatureCommand :
ISelfBuiltDeviceCommand<GetCurrentTemperatureCommandParameters,
        GetCurrentTemperatureCommandResult>
{
    public GetCurrentTemperatureCommand(byte pidId)
    {
        ParamsRequest = new GetCurrentTemperatureCommandParameters
        {
            PidId = pidId
        };
    }

    public ushort Command => (ushort)CommandsEnum.GetCurrentTemperature;

    public GetCurrentTemperatureCommandParameters ParamsRequest { get; }
}
```

Файл CapabilityHandle.cs

```
public class CapabilityHandle<TCapabilityHandle>
{
    public CapabilityHandle(ulong deviceId, TCapabilityHandle handle)
    {
        DeviceId = deviceId;
        Handle = handle;
    }

    public ulong DeviceId { get; }
    public TCapabilityHandle Handle { get; }
}
```

Файл DevicePresenceChangedEventArgs.cs

```
public class DevicePresenceChangedEventArgs<T> : EventArgs
{
    public DevicePresenceChangedEventArgs(T deviceId, PresenceStatus status)
    {
        DeviceId = deviceId;
        Status = status;
    }

    public T DeviceId { get; }
    public PresenceStatus Status { get; }
}
```

Файл PidCoefficients.cs

```
public class PidCoefficients
{
    public PidCoefficients(float kp, float ki, float kd)
    {
        Kp = kp;
        Ki = ki;
```

```
        Kd = kd;
    }

    public float Kp { get; }
    public float Ki { get; }
    public float Kd { get; }
}
```

Файл DevicesService.cs

```
public class DevicesService : IDevicesService
{
    private readonly IDeviceManager _deviceManager;

    public DevicesService(IDeviceManager deviceManager)
    {
        _deviceManager = deviceManager;
        _deviceManager.DevicePresenceChanged += OnDevicePresenceChanged;
    }

    public event EventHandler<DeviceConnectedEventArgs>? DeviceConnected;
    public event EventHandler<DeviceDisconnectedEventArgs>?
DeviceDisconnected;

    public async Task<Device> GetDevice(ulong deviceId)
    {
        var controllersKeys = new List<TemperatureControllerKey>();

        var deviceName = GetDeviceName(deviceId);
        var connectionName = GetConnectionName(deviceId);
        var supportsPid = SupportsPid(deviceId);

        await foreach (var key in GetControllersKeys(deviceId))
            controllersKeys.Add(key);

        return new Device(deviceId, deviceName, connectionName,
controllersKeys, supportsPid);
    }

    public async Task<IEnumerable<Device>> GetDevices()
    {
        var deviceIdsList = _deviceManager.GetAllDeviceIds();
        var devices = new List<Device>();

        foreach (var deviceId in deviceIdsList)
            devices.Add(await GetDevice(deviceId));

        return devices;
    }

    private string GetDeviceName(ulong deviceId)
    {
        return _deviceManager.GetDeviceNameById(deviceId);
    }

    private string GetConnectionName(ulong deviceId)
    {
        return _deviceManager.GetConnectionNameByDeviceId(deviceId);
    }

    private bool SupportsPid(ulong deviceId)
    {
```

```
        return
_deviceManager.IsDeviceCapabilitySupported<IPidControllerCapability>(deviceId
);
    }

    private async IAsyncEnumerable<TemperatureControllerKey>
GetControllersKeys(ulong deviceId)
    {
        var capability =
_deviceManager.TryGetDeviceCapability<ITemperatureControllerCapability>(devic
eId);

        if (capability == null)
            yield break;

        var controllersNumber = await capability.GetChannelsNumber();
        for (byte i = 0; i < controllersNumber; i++)
            yield return new TemperatureControllerKey(deviceId, i);
    }

    private async void OnDevicePresenceChanged(object? sender,
DevicePresenceChangedEventArgs<ulong> args)
    {
        if (args.Status == PresenceStatus.Connected)
            DeviceConnected?.Invoke(this, new DeviceConnectedEventArgs(await
GetDevice(args.DeviceId)));
        else
            DeviceDisconnected?.Invoke(this, new
DeviceDisconnectedEventArgs(args.DeviceId));
    }
}
```

Файл LocalizationService.cs

```
public class LocalizationService : ILocalizationService
{
    private readonly IUnitOfWork _uow;
    private readonly IUserProfileService _userProfileService;

    public LocalizationService(IUnitOfWork uow, IUserProfileService
userProfileService)
    {
        _uow = uow;
        _userProfileService = userProfileService;
    }

    public async Task<IEnumerable<Locale>> GetAvailableLocalizations()
    {
        var localizations = await _uow.LanguagesRepository.GetListAsync();

        var locales = localizations.Select(localization =>
            new Locale(localization.Id, localization.NativeName,
localization.EnglishName, localization.Code));

        return locales;
    }

    public async Task<string> GetCurrentLanguageCode()
    {
        var currentProfileId = _userProfileService.GetProfileId();
        var spec = new ProfileWithLanguageSpecification(currentProfileId);
        var profile = await _uow.ProfilesRepository.GetBySpecAsync(spec);
        if (profile is null)
```

```
            throw new ArgumentException("Profile doesn`t exist");

        return profile.Language.Code;
    }

    public async Task SaveSelectedLocalization(uint languageId)
    {
        var currentProfileId = _userProfileService.GetProfileId();
        var profile = await
_uow.ProfilesRepository.GetByIdAsync(currentProfileId);
        if (profile is null)
            throw new ArgumentException("Profile doesn`t exist");
        profile.LanguageId = languageId;
        _uow.ProfilesRepository.Update(profile);
        await _uow.SaveChanges();
    }
}
```

Файл TemperatureControllerService.cs

```
public class TemperatureControllerService : ITemperatureControllerService
{
    private readonly IDeviceManager _deviceManager;

    public TemperatureControllerService(IDeviceManager deviceManager)
    {
        _deviceManager = deviceManager;
    }

    public async Task<TemperatureController>
GetTemperatureController(TemperatureControllerKey controllerKey)
    {
        var currentTemperature = await GetCurrentTemperature(controllerKey);
        var desiredTemperature = await GetDesiredTemperature(controllerKey);

        return new TemperatureController(controllerKey, currentTemperature,
desiredTemperature);
    }

    public async Task SetDesiredTemperature(TemperatureControllerKey
controllerKey, ushort value)
    {
        var handle = GetHandle(controllerKey.DeviceId);
        await handle.SetDesiredTemperature(controllerKey.ChannelId, value);
    }

    private async Task<ushort> GetCurrentTemperature(TemperatureControllerKey
controllerKey)
```

```
    {
        var handle = GetHandle(controllerKey.DeviceId);
        return await handle.GetCurrentTemperature(controllerKey.ChannelId);
    }

    private async Task<ushort> GetDesiredTemperature(TemperatureControllerKey
controllerKey)
    {
        var handle = GetHandle(controllerKey.DeviceId);
        return await handle.GetDesiredTemperature(controllerKey.ChannelId);
    }

    private ITemperatureControllerCapability GetHandle(ulong deviceId)
    {
        var handle =
_deviceManager.TryGetDeviceCapability<ITemperatureControllerCapability>(devic
eId);

        if (handle == null)
            throw new CapabilityIsNotSupportedException();

        return handle;
    }
}
```

Файл PidTemperatureControllerService.cs

```
public class PidTemperatureControllerService : TemperatureControllerService,
IPidTemperatureControllerService
{
    private readonly IDeviceManager _deviceManager;

    public PidTemperatureControllerService(IDeviceManager deviceManager) :
base(deviceManager)
    {
        _deviceManager = deviceManager;
    }

    public async Task<PidTemperatureController>
GetPidTemperatureController(TemperatureControllerKey controllerKey)
    {
        var temperatureController = await
GetTemperatureController(controllerKey);
```

```
        var coefficients = await GetCoefficients(controllerKey);

        return new PidTemperatureController(temperatureController,
coefficients);
    }

    public async Task<PidControllerCoefficients>
GetCoefficients(TemperatureControllerKey controllerKey)
    {
        var handle = GetHandle(controllerKey.DeviceId);
        var pidCoefficients = await
handle.GetCoefficients(controllerKey.ChannelId);

        return new PidControllerCoefficients(pidCoefficients.Kp,
pidCoefficients.Ki, pidCoefficients.Kd);
    }

    public async Task SetPidKp(TemperatureControllerKey controllerKey, float
value)
    {
        var handle = GetHandle(controllerKey.DeviceId);
        await handle.SetKpCoefficient(controllerKey.ChannelId, value);
    }

    public async Task SetPidKi(TemperatureControllerKey controllerKey, float
value)
    {
        var handle = GetHandle(controllerKey.DeviceId);
        await handle.SetKiCoefficient(controllerKey.ChannelId, value);
    }

    public async Task SetPidKd(TemperatureControllerKey controllerKey, float
value)
    {
        var handle = GetHandle(controllerKey.DeviceId);
        await handle.SetKdCoefficient(controllerKey.ChannelId, value);
    }

    private IPidControllerCapability GetHandle(ulong deviceId)
    {
        var handle =
_deviceManager.TryGetDeviceCapability<IPidControllerCapability>(deviceId);
```

```
        if (handle == null)
            throw new Exception();

        return handle;
    }
}
```

Файл SerialPortService.cs

```
public class SerialPortsService : ISerialPortsService
{
    private readonly IDeviceManager _deviceManager;
    private readonly IHardwareDetector<SerialConnectionParameters>
_hardwareDetector;
    private readonly ISerialPortsSettingsService _serialPortsSettingsService;

    private readonly ConcurrentDictionary<string, SerialPortInfo>
_serialPorts = new();
    private readonly ISerialPortsMonitor _serialPortsMonitor;

    public SerialPortsService(IDeviceManager deviceManager,
        IHardwareDetector<SerialConnectionParameters> hardwareDetector,
        ISerialPortsMonitor serialPortsMonitor,
        ISerialPortsSettingsService serialPortsSettingsService)
    {
        _deviceManager = deviceManager;
        _hardwareDetector = hardwareDetector;
        _serialPortsMonitor = serialPortsMonitor;
        _serialPortsSettingsService = serialPortsSettingsService;

        SubscribeEvents();
        _serialPortsMonitor.Start(1000);
    }

    public IEnumerable<SerialPortInfoDto> SerialPorts =>
_serialPorts.Values.Select(info => PortInfoToDto(info));

    public event EventHandler<SerialPortInfoEventArgs>? PortAdded;
    public event EventHandler<SerialPortRemovedEventArgs>? PortRemoved;
    public event EventHandler<SerialPortInfoEventArgs>? PortInfoUpdateEvent;

    public async Task Connect(string portName)
    {
```

```
        var serialPortSettings = await
_serialPortsSettingsService.GetByPortName(portName) ?? new
SerialPortSettings(portName);

        var portExists = _serialPorts.TryGetValue(portName, out var port);

        if (!portExists || port?.ConnectedDeviceId != 0)
            return;

        var id = await
_hardwareDetector.ConnectDeviceWithIdentification(Map(serialPortSettings));
        UpdatePortInfo(portName, id);
    }

    public void Disconnect(string portName)
    {
        if (!_serialPorts.TryGetValue(portName, out var port))
            return;

        _deviceManager.RemoveDevice(port.ConnectedDeviceId);
        UpdatePortInfo(portName, 0);
    }

    private void SubscribeEvents()
    {
        _serialPortsMonitor.PortPresenceStatusChanged +=
OnPortPresenceChanged;
    }

    private void UpdatePortInfo(string portName, ulong id)
    {
        if (!_serialPorts.TryGetValue(portName, out var portInfo))
            return;

        portInfo.ConnectedDeviceId = id;
        PortInfoUpdateEvent?.Invoke(this, new
SerialPortInfoEventArgs(PortInfoToDto(portInfo)));
    }

    private void OnPortPresenceChanged(object? sender,
SerialPortPresenceEventArgs eventArgs)
    {
        switch (eventArgs.Status)
```

```
        {
            case PresenceStatus.Connected:
                var portInfo = new SerialPortInfo(eventArgs.PortName, 0);
                _serialPorts.TryAdd(eventArgs.PortName, portInfo);
                PortAdded?.Invoke(this, new
SerialPortInfoEventArgs(PortInfoToDto(portInfo)));
                break;
            case PresenceStatus.Disconnected:
                if (_serialPorts.TryGetValue(eventArgs.PortName, out
portInfo) && portInfo.ConnectedDeviceId != 0)
                    _deviceManager.RemoveDevice(portInfo.ConnectedDeviceId);
                _serialPorts.Remove(eventArgs.PortName, out _);
                PortRemoved?.Invoke(this, new
SerialPortRemovedEventArgs(eventArgs.PortName));
                break;
        }
    }

    private SerialPortInfoDto PortInfoToDto(SerialPortInfo portInfo)
    {
        var isConnected = portInfo.ConnectedDeviceId != 0;
        return new SerialPortInfoDto(portInfo.SerialPortName, isConnected);
    }

    private SerialPortSettings Map(SerialConnectionParameters parameters)
    {
        return new SerialPortSettings(
            parameters.PortName, parameters.BaudRate, parameters.Parity,
parameters.DataBits, parameters.StopBits);
    }

    private SerialConnectionParameters Map(SerialPortSettings parameters)
    {
        return new SerialConnectionParameters(
            parameters.PortName, parameters.BaudRate, parameters.Parity,
parameters.DataBits, parameters.StopBits);
    }
}
```

Файл SerialPortSettingsService.cs

```
public class SerialPortsSettingsService : ISerialPortsSettingsService
{
    private readonly IUnitOfWork _uow;
    private readonly IUserProfileService _userProfileService;

    public SerialPortsSettingsService(IUnitOfWork uow, IUserProfileService
userProfileService)
    {
        _uow = uow;
        _userProfileService = userProfileService;
    }

    public async Task<SerialPortSettings?> GetByPortName(string portName)
    {
        var userProfileId = _userProfileService.GetProfileId();
        var spec = new
SerialConnectionParametersByPortNameSpecification(userProfileId, portName);
        var serialConnectionParameters = await
_uow.SerialConnectionParametersRepository.GetBySpecAsync(spec);

        return serialConnectionParameters != null ?
Map(serialConnectionParameters) : null;
    }

    public async Task Add(SerialPortSettings portSettings)
    {
        _uow.SerialConnectionParametersRepository.Add(Map(portSettings));
        await _uow.SaveChanges();
    }

    public async Task Remove(string portName)
    {
        var userProfileId = _userProfileService.GetProfileId();
        var spec = new
SerialConnectionParametersByPortNameSpecification(userProfileId, portName);
        var serialConnectionParameters = await
_uow.SerialConnectionParametersRepository.GetBySpecAsync(spec);
        if (serialConnectionParameters != null)
        {
```

```
_uow.SerialConnectionParametersRepository.Delete(serialConnectionParameters.I
d);
            await _uow.SaveChanges();
        }
    }

    public async Task Update(SerialPortSettings portSettings)
    {
        _uow.SerialConnectionParametersRepository.Update(Map(portSettings));
        await _uow.SaveChanges();
    }

    private SerialPortSettings Map(SerialConnectionParametersEntity
parameters)
    {
        return new SerialPortSettings(
            parameters.PortName, parameters.BaudRate,
(Parity)parameters.Parity, parameters.DataBits,
(StopBits)parameters.StopBits);
    }

    private SerialConnectionParametersEntity Map(SerialPortSettings
parameters)
    {
        var profileId = _userProfileService.GetProfileId();
        return new SerialConnectionParametersEntity(
            profileId, parameters.PortName, parameters.BaudRate,
            (int)parameters.Parity, parameters.DataBits,
(int)parameters.StopBits);
    }
}
```

Файл ThermalProfileService.cs

```
public class ThermalProfileService : IThermalProfileService
{
    private readonly IUnitOfWork _uow;
    private readonly IUserProfileService _userProfileService;

    public ThermalProfileService(IUnitOfWork uow, IUserProfileService
userProfileService)
    {
```

```
        _uow = uow;
        _userProfileService = userProfileService;
    }

    public async Task<IEnumerable<ThermalProfile>> GetAll()
    {
        var userProfileId = _userProfileService.GetProfileId();
        var spec = new
ThermalProfileWithAllDataByUserSpecification(userProfileId);
        var result = await
_uow.ThermalProfilesRepository.GetListBySpecAsync(spec);
        return result.Select(Map);
    }

    public async Task<ThermalProfile> GetById(uint thermalProfileId)
    {
        var spec = new
ThermalProfileWithAllDataSpecification(thermalProfileId);
        var result = await
_uow.ThermalProfilesRepository.GetBySpecAsync(spec);
        if (result == null)
            throw new ArgumentNullException(nameof(thermalProfileId));

        return Map(result);
    }

    public async Task Add(ThermalProfile thermalProfile)
    {
        var thermalProfileEntity = Map(thermalProfile);
        _uow.ThermalProfilesRepository.Add(thermalProfileEntity);
        await _uow.SaveChanges();
    }

    public async Task Remove(uint thermalProfileId)
    {
        _uow.ThermalProfilesRepository.Delete(thermalProfileId);
        await _uow.SaveChanges();
    }

    public async Task Update(ThermalProfile thermalProfile)
    {
        _uow.ThermalProfilesRepository.Update(Map(thermalProfile));
        await _uow.SaveChanges();
```

```
    }


    private ThermalProfile Map(ThermalProfileEntity entity)
    {
        var controllerThermalProfiles = entity.Parts.Select(Map).ToList();
        return new ThermalProfile(entity.Id, entity.Name,
controllerThermalProfiles);
    }

    private ThermalProfileEntity Map(ThermalProfile model)
    {
        var profileId = _userProfileService.GetProfileId();
        var controllerThermalProfilesParts =
model.ControllersThermalProfiles.Select(Map).ToList();
        return new ThermalProfileEntity(model.Id, model.Name,
controllerThermalProfilesParts, profileId);
    }

    private ControllerThermalProfile Map(ThermalProfilePartEntity entity)
    {
        var curve = entity.TemperatureCurve.Select(Map).ToList();
        return new ControllerThermalProfile(entity.Id, entity.Name,
entity.Color, curve);
    }

    private ThermalProfilePartEntity Map(ControllerThermalProfile model)
    {
        var curve = model.TemperatureMeasurements.Select(Map).ToList();
        return new ThermalProfilePartEntity(model.Id, model.Name,
model.ArgbColor, curve);
    }

    private TemperatureMeasurementPoint Map(TemperatureMeasurementPointEntity
entity)
    {
        return new TemperatureMeasurementPoint(entity.Id,
entity.SecondsElapsed, entity.Temperature);
    }

    private TemperatureMeasurementPointEntity Map(TemperatureMeasurementPoint
model)
    {
```

```
        return new TemperatureMeasurementPointEntity(model.Id,
model.SecondsElapsed, model.Temperature);
    }
}
```

Файл SerialPortsMonitor.cs

```
public class SerialPortsMonitor : ISerialPortsMonitor
{
    private readonly ISerialPortsProvider _serialPortsProvider;
    private readonly ITimer _timer;
    private bool _isDisposed;

    private List<string> _lastDetectedPortNames = new();

    public SerialPortsMonitor(ITimer timer, ISerialPortsProvider
serialPortsProvider)
    {
        _timer = timer;
        _serialPortsProvider = serialPortsProvider;
    }

    public IReadOnlyList<string> AllPortNames =>
_lastDetectedPortNames.AsReadOnly();
    public event EventHandler<SerialPortPresenceEventArgs>?
PortPresenceStatusChanged;

    public void Start(uint scanPeriodMs)
    {
        _timer.TimerIntervalElapsed += CheckPortsStatus;
        _timer.Interval = scanPeriodMs;
        _timer.Start();
    }

    public void Stop()
    {
        _timer.TimerIntervalElapsed -= CheckPortsStatus;
        _timer.Stop();
    }

    public void Dispose()
    {
        Dispose(true);
```

```
    }

    private void CheckPortsStatus(object sender, DateTime dateTime)
    {
        var availablePorts =
_serialPortsProvider.GetAvailablePorts().ToList();

        var connectedPorts = availablePorts.Except(_lastDetectedPortNames);
        var disconnectedPorts =
_lastDetectedPortNames.Except(availablePorts);

        foreach (var port in connectedPorts)
            PortPresenceStatusChanged?.Invoke(this, new
SerialPortPresenceEventArgs(port, PresenceStatus.Connected));

        foreach (var port in disconnectedPorts)
            PortPresenceStatusChanged?.Invoke(this, new
SerialPortPresenceEventArgs(port, PresenceStatus.Disconnected));

        _lastDetectedPortNames = availablePorts;
    }

    protected virtual void Dispose(bool disposing)
    {
        if (_isDisposed)
            return;

        if (disposing)
            _timer.Dispose();

        _isDisposed = true;
    }
}
```

Файл SerialPortsProvider.cs

```
public class SerialPortsProvider : ISerialPortsProvider
{
    public IEnumerable<string> GetAvailablePorts()
    {
        return SerialPort.GetPortNames();
    }
}
```

Файл TemperatureHistoryTracker.cs

```
public class TemperatureHistoryTracker : ITemperatureHistoryTracker
{
    private readonly ConcurrentDictionary<TemperatureControllerKey,
Collection<TemperatureMeasurement>>
        _temperatureHistory =
            new();

    public IEnumerable<TemperatureMeasurement>
GetControllerHistory(TemperatureControllerKey controllerKey)
    {
        if (!_temperatureHistory.TryGetValue(controllerKey, out var
controllerTemperatureHistory))
            yield break;

        foreach (var measurement in controllerTemperatureHistory)
            yield return measurement;
    }

    public void AddMeasurement(TemperatureControllerKey controllerKey,
TemperatureMeasurement temperatureMeasurement)
    {
        if (_temperatureHistory.TryGetValue(controllerKey, out var
controllerTemperatureHistory))
        {
            controllerTemperatureHistory.Add(temperatureMeasurement);
        }
        else
        {
            var newControllerHistory = new Collection<TemperatureMeasurement>
{ temperatureMeasurement };
            _temperatureHistory.TryAdd(controllerKey, newControllerHistory);
        }
    }

    public void RemoveControllerHistory(TemperatureControllerKey
controllerKey)
    {
        _temperatureHistory.Remove(controllerKey, out _);
    }

    public void Clear()
```

```
    {
        _temperatureHistory.Clear();
    }
}
```

Файл ThermalProfileWithAllDataByUserSpecification.cs

```
public sealed class ThermalProfileWithAllDataByUserSpecification :
Specification<ThermalProfileEntity>
{
    public ThermalProfileWithAllDataByUserSpecification(uint profileId)
    {
        Query
            .Where(entity => entity.ProfileId == profileId)
            .Include(entity => entity.Parts)
            .ThenInclude(partEntity => partEntity.TemperatureCurve);
    }
}
```

Файл ThermalProfileWithAllDataByUserSpecification.cs

```
public sealed class ThermalProfileWithAllDataByUserSpecification :
Specification<ThermalProfileEntity>
{
    public ThermalProfileWithAllDataByUserSpecification(uint profileId)
    {
        Query
            .Where(entity => entity.ProfileId == profileId)
            .Include(entity => entity.Parts)
            .ThenInclude(partEntity => partEntity.TemperatureCurve);
    }
}
```

Файл JobStateService.cs

```
public class JobStateService
{
    public IJob? ActiveJob { get; private set; }

    public event EventHandler<JobStartedEventArgs> JobStarted;

    public void AddJob(IJob job)
    {
```

```
        if (ActiveJob != null)
            throw new JobException("You can`t run second job at the same
time!");

        SubscribeToJobEvents(job);

        ActiveJob = job;
        JobStarted?.Invoke(this, new JobStartedEventArgs(job.JobType));
    }

    private void SubscribeToJobEvents(IJob job)
    {
        job.StateChanged += JobOnStateChanged;
    }

    private void UnsubscribeFromJobEvents(IJob job)
    {
        job.StateChanged -= JobOnStateChanged;
    }

    private void JobOnStateChanged(object? sender, JobStateChangedEventArgs
e)
    {
        if (sender is null)
            return;

        var job = (IJob)sender;
        if (!e.JobState.IsCompleted())
            return;

        ActiveJob = null;
        UnsubscribeFromJobEvents(job);
    }
}
```

Файл SerialCommunicationClient.cs

```
public class SerialCommunicationClient : ISerialCommunicationClient
{
    private const int MaxDataLength = 250;
    private readonly ICrc16Calculator _crc16Calculator;
    private readonly object _lock = new();
```

```
    private readonly int _responseTimeout;
    private readonly IStructSerializer _serializer;
    private readonly ISerialPort _serialPort;

    private bool _disposedValue;

    public SerialCommunicationClient(ISerialPort serialPort,
IStructSerializer serializer,
        ICrc16Calculator crc16Calculator,
        SerialPortSettings settings, int responseTimeout = 500)
    {
        _serializer = serializer;
        _serialPort = serialPort;
        _crc16Calculator = crc16Calculator;
        _responseTimeout = responseTimeout;
        PortSettings = settings;

        _serialPort.PortName = PortSettings.PortName;
        _serialPort.BaudRate = PortSettings.BaudRate;
        _serialPort.Parity = PortSettings.Parity;
        _serialPort.DataBits = PortSettings.DataBits;
        _serialPort.StopBits = PortSettings.StopBits;
    }

    public SerialPortSettings PortSettings { get; }

    public void Connect()
    {
        if (_serialPort.IsOpen)
            throw new ConnectionException("Serial port is already opened!");

        try
        {
            _serialPort.Open();
        }
        catch (Exception e)
        {
            throw new ConnectionException("Serial port connection can`t be
established", e);
        }
    }

    public void Close()
```

```
    {
        Dispose();
    }

    public void ExecuteCommand<TParameters>(ushort commandCode, TParameters
parameters)
        where TParameters : struct
    {
        var commandParams = _serializer.Serialize(parameters);
        var data = ExecuteCommand(commandCode, commandParams);
        if (data.Length != 0)
            throw new UnexpectedResponseException("Unexpected response. It
must not contain any data.");
    }

    public TResult ExecuteCommand<TParameters, TResult>(ushort commandCode,
        TParameters parameters)
        where TParameters : struct where TResult : struct
    {
        var commandParams = _serializer.Serialize(parameters);
        var data = ExecuteCommand(commandCode, commandParams);

        var response = _serializer.Deserialize<TResult>(data);

        if (response == null)
            throw new UnexpectedResponseException("Unexpected response. Data
is not present or it does not fits structure.");

        return response.Value;
    }

    public byte[] ExecuteCommand(ushort command, byte[] parameters)
    {
        try
        {
            lock (_lock)
            {
                SendPacket(command, parameters);
                var result = ReceivePacket();

                return result;
            }
        }
```

```
        catch (Exception ex) when (ex is InvalidOperationException or
TimeoutException)
        {
            throw new ConnectionException("Connection was lost");
        }
    }

    public async Task ExecuteCommandAsync<TParameters>(ushort commandCode,
TParameters parameters)
        where TParameters : struct
    {
        await Task.Run(() => ExecuteCommand(commandCode, parameters));
    }

    public async Task<TResult> ExecuteCommandAsync<TParameters,
TResult>(ushort commandCode,
        TParameters parameters)
        where TParameters : struct where TResult : struct
    {
        return await Task.Run(() => ExecuteCommand<TParameters,
TResult>(commandCode, parameters));
    }

    public async Task<byte[]> ExecuteCommandAsync(ushort command, byte[]
parameters)
    {
        return await Task.Run(() => ExecuteCommand(command, parameters));
    }

    public void Dispose()
    {
        if (!_disposedValue)
        {
            _serialPort.Dispose();
            _disposedValue = true;
            GC.SuppressFinalize(this);
        }
    }

    private void SendPacket(ushort command, byte[] parameters)
    {
        if (parameters is null)
            throw new ArgumentNullException(nameof(parameters));
```

```
        var dataLength = parameters.Length;

        if (dataLength > MaxDataLength)
            throw new ArgumentException($"Command parameters length can`t be
more than {MaxDataLength} bytes",
                nameof(parameters));

        var headerLength = 3;
        var crcLength = 2;
        var packet = new byte[headerLength + dataLength + crcLength]; //
header(3 bytes) + data + crc(2bytes)

        packet[0] = (byte)(command & 0xff);
        packet[1] = (byte)((command >> 8) & 0xff);
        packet[2] = (byte)dataLength;
        Array.Copy(parameters, 0, packet, 3, parameters.Length);
        var crc = _crc16Calculator.Calculate(packet, 0, 3 + dataLength);
        packet[packet.Length-2] = (byte)(crc & 0xff);
        packet[packet.Length-1] = (byte)((crc >> 8) & 0xff);

        _serialPort.Write(packet, 0, packet.Length);
        while (_serialPort.BytesToWrite > 0)
            Thread.Sleep(3);
    }

    private byte[] ReceivePacket()
    {
        var buffer = new byte[256];
        var headerLength = 2;
        var crcLength = 2;

        var isAnyResponse = IsAnyResponse();
        if (!isAnyResponse)
            throw new PackageIsCorruptedException("No response.");

        var isHeaderInBuffer = IsDataInBuffer(headerLength);
        if (!isHeaderInBuffer)
            throw new PackageIsCorruptedException("Header is not fully
present in packet");

        _serialPort.Read(buffer, 0, headerLength); // read header
```

```
        var dataLength = buffer[headerLength - 1];
        if (dataLength > MaxDataLength)
            throw new PackageIsCorruptedException("The data length value
received was too large. Data cannot be processed.");

        var dataAndChecksumLength = dataLength + crcLength;
        var isFullPacketInBuffer = IsDataInBuffer(dataAndChecksumLength);
        if (!isFullPacketInBuffer)
            throw new PackageIsCorruptedException("Received incomplete
package.");
        _serialPort.Read(buffer, 2, dataAndChecksumLength); // read to end of
packet

        var fullPacketLength = headerLength + dataAndChecksumLength;

        var calculatedCrc = _crc16Calculator.Calculate(buffer, 0,
headerLength + dataLength);
        var receivedCrc = buffer[fullPacketLength - 2] |
(buffer[fullPacketLength - 1] << 8);
        if (calculatedCrc != receivedCrc)
            throw new PackageIsCorruptedException("Packet checksum is
invalid");

        if (buffer[0] != (byte)CommandResponse.Ok)
        {
            byte maxCommandResponseCode =
(byte)Enum.GetValues(typeof(CommandResponse)).Cast<CommandResponse>().Max();

            if (buffer[0] <= maxCommandResponseCode)
                throw new
CommandResponseException((CommandResponse)buffer[0]);

            throw new UnexpectedResponseException("Unexpected error code
received");
        }

        var resultData = new byte[dataLength];
        Array.Copy(buffer, headerLength, resultData, 0, dataLength);
        return resultData;
    }

    private bool IsAnyResponse()
    {
```

```
            short timer = 0;
            while (_serialPort.BytesToRead == 0)
            {
                Thread.Sleep(1);
                timer++;
                if (timer >= _responseTimeout)
                    return false;
            }

            return true;
        }

        private bool IsDataInBuffer(int bytesNumber)
        {
            var bytesAvailable = _serialPort.BytesToRead;
            while (bytesAvailable < bytesNumber)
            {
                var symbolTime = Math.Max(1000 / (_serialPort.BaudRate / 8) * 4,
2);
                //delay == transfer speed of 1 byte multiplied by 4 if it`s more
than 2ms
                Thread.Sleep(symbolTime);
                var newBytesAvailable = _serialPort.BytesToRead;
                if (bytesAvailable == newBytesAvailable)
                    return false;

                bytesAvailable = newBytesAvailable;
            }

            return true;
        }
    }
```

Файл SerialCommunicationClientFactory.cs

```
public class SerialCommunicationClientFactory :
ISerialCommunicationClientFactory
{
    private readonly ICrc16Calculator _crc16Calculator;
    private readonly IStructSerializer _structSerializer;
```

```
    public SerialCommunicationClientFactory(IStructSerializer
structSerializer,
        ICrc16Calculator crc16Calculator)
    {
        _structSerializer = structSerializer;
        _crc16Calculator = crc16Calculator;
    }

    public ISerialCommunicationClient Create(SerialPortSettings portSettings)
    {
        var serialPort = new SerialPortDefault();
        return new SerialCommunicationClient(serialPort, _structSerializer,
_crc16Calculator, portSettings);
    }

    public ISerialCommunicationClient Create(SerialPortSettings portSettings,
int responseTimeout)
    {
        var serialPort = new SerialPortDefault();
        return new SerialCommunicationClient(serialPort, _structSerializer,
_crc16Calculator, portSettings,
            responseTimeout);
    }
}
```

Файл serial_communication.c

```
static int timeout = 2;
static unsigned char buffer[BUFFER_LENGTH];
static unsigned char buffer_counter = 0;
static unsigned char transmitting_counter = 0;
static unsigned long last_buffer_counter_update_time = 0;

static SERIAL_PROCESS_STATE state = UNINITIALIZED;
static command_hander_t Command_handler;

void serial_transmit_callback()
{
    if(transmitting_counter < buffer_counter)
    {
        usart_send_byte(buffer[transmitting_counter]);
        transmitting_counter++;
```

```
    }
    else
    {
        usart_transmit_byte_interrupt_disable();
        buffer_counter = 0;
        transmitting_counter = 0;
        last_buffer_counter_update_time = 0;
        state = RECEIVING_HEADER;
        usart_receive_byte_interrupt_enable();
    }

}

void serial_receive_callback(char symbol)
{
    last_buffer_counter_update_time = millis();
    buffer[buffer_counter] = symbol;
    buffer_counter++;

    if(buffer_counter == 255)
        usart_receive_byte_interrupt_disable();
}

void serial_process()
{
    static unsigned char packet_length = 0;
    static unsigned char data_length = 0;

    switch(state)
    {
        case RECEIVING_HEADER:
        {
            if(buffer_counter >= 3 && buffer[2] <= MAX_DATA_LENGTH)
            {
                data_length = buffer[2];
                packet_length = data_length + 5;
                state = RECEIVING_DATA_AND_CRC;
            }
            else if(buffer_counter != 0 &&
millis_timeout_check(&last_buffer_counter_update_time, timeout))
            {
                usart_receive_byte_interrupt_disable();
                serial_write_response(CORRUPTED_PACKET, 0, 0);
```

```
                state = TRANSMITTING_RESPONSE;
                usart_transmit_byte_interrupt_enable();
            }

            break;
        }
        case RECEIVING_DATA_AND_CRC:
        {
            if(buffer_counter >= packet_length)
            {
                usart_receive_byte_interrupt_disable();
                unsigned short caluclated_crc = CRC16(buffer, packet_length-
2);
                unsigned short received_crc = buffer[buffer_counter-2] |
(buffer[buffer_counter-1] << 8);
                if(caluclated_crc == received_crc)
                {
                    unsigned short cmd = buffer[0] | (buffer[1] << 8);
                    command_response_t result = Command_handler(cmd,
&buffer[3], data_length);
                    serial_write_response(result.response_code, result.data,
result.data_length);
                }
                else
                {
                    serial_write_response(CORRUPTED_PACKET, 0, 0);
                }
                state = TRANSMITTING_RESPONSE;
                usart_transmit_byte_interrupt_enable();
            }
            else if(millis_timeout_check(&last_buffer_counter_update_time,
timeout))
            {
                usart_receive_byte_interrupt_disable();
                serial_write_response(CORRUPTED_PACKET, 0, 0);
                state = TRANSMITTING_RESPONSE;
                usart_transmit_byte_interrupt_enable();
            }
            break;
        }
        case TRANSMITTING_RESPONSE:
            break;
    }
```

```
}

void serial_write_response(unsigned char response, unsigned char* data,
unsigned char data_length)
{
    unsigned char packet_length = data_length + 4;
    buffer[0] = response;
    buffer[1] = data_length;

    for(unsigned char i = 0; i < (data_length); i++)
        buffer[i+2] = data[i];

    unsigned short caluclated_crc = CRC16(buffer, (packet_length-2));
    buffer[packet_length-2] = caluclated_crc & 0xff;
    buffer[packet_length-1] = (caluclated_crc >> 8) & 0xff;

    buffer_counter = packet_length;
}

void serial_communication_init(unsigned int usartBaudrate, command_hander_t
command_handler)
{
    transmit_callback_t transmit_callback =
(transmit_callback_t)serial_transmit_callback;
    receive_callback_t receive_callback =
(receive_callback_t)serial_receive_callback;
    Command_handler = command_handler;
    timeout = ((float)1000/(usartBaudrate/8)*4)+1;
    if(timeout < 2)
        timeout = 2;
    usart_init(usartBaudrate, transmit_callback, receive_callback);

    state = RECEIVING_HEADER;
    buffer_counter = 0;
    usart_receive_byte_interrupt_enable();
}
```

Файл heater.c

```
static Heater_t* heaters[MAX_HEATERS];
static int heatersNumber = 0;
```

```
static void configure_timer()
{
    cli();
#ifdef __AVR_ATmega328P__
    TCNT2 = 0;                                  // Clear TCNT0
    OCR2A = F_CPU/100/1024-1;                   // 100 Hz
(F_CPU/((OCR2A+1)*1024))
    TCCR2A = (1 << WGM21);                      // CTC
    TCCR2B = (1 << CS22) | (1 << CS20);         // Prescaler 1024
    TIMSK2 = (1 << OCIE2A);                     // Output Compare Match A
Interrupt Enable
#elif defined __AVR_ATmega8A__
    const unsigned short sds = F_CPU/100/64-1;
    TCNT1 = 0;                                              // Clear TCNT0
    OCR1A = F_CPU/100/64-1;                                 // 100 Hz
(F_CPU/((OCR0A+1)*64))
    TCCR1A = 0;                                             // CTC
    TCCR1B = (1 << WGM12) | (1 << CS11) | (1 << CS10);      // CTC & Prescaler 64
    TIMSK |= (1 << OCIE1A);                                 // Output Compare
Match A Interrupt Enable
#endif
    sei();
}

void heater_set_power_percentage(Heater_t* heater, int powerPercentage)
{
    if(powerPercentage >= 0 && powerPercentage <= 100)
        heater->powerPercentage = powerPercentage;
}

int heater_get_current_power_percentage(Heater_t* heater)
{
    return heater->powerPercentage;
}

void heater_disable(Heater_t* heater)
{
    heater->isEnabled = 0;
}

void heater_enable(Heater_t* heater)
{
    heater->isEnabled = 1;
```

```
}

void heater_init(Heater_t* heaterPtr, GPIO_t* gpio)
{
    Heater_t heater = {
        .heaterGpio = gpio,
        .isEnabled = 0
    };

    gpio_configure_as_output(heater.heaterGpio);
    gpio_set_high(heater.heaterGpio);
    heater_set_power_percentage(&heater, 0);

    heaters[heatersNumber] = heaterPtr;
    heatersNumber++;
    *heaterPtr = heater;

    static bool firstInit = 1;
    if(firstInit)
    {
        configure_timer();
        firstInit = 0;
    }
}

#ifdef __AVR_ATmega328P__
ISR(TIMER2_COMPA_vect) {
#elif defined __AVR_ATmega8A__
ISR(TIMER1_COMPA_vect) {
#endif
    for(int i=0;i<heatersNumber;i++)
    {
        Heater_t* currentHeater = heaters[i];

        bool isActive = currentHeater->currentHalfWave < currentHeater-
>powerPercentage;

        if(currentHeater->isEnabled && isActive &&
gpio_is_high(currentHeater->heaterGpio))
            gpio_set_low(currentHeater->heaterGpio);
        else if(!isActive && gpio_is_low(currentHeater->heaterGpio))
            gpio_set_high(currentHeater->heaterGpio);
```

```
        if(currentHeater->currentHalfWave < 99)
            currentHeater->currentHalfWave++;
        else
            currentHeater->currentHalfWave = 0;
    }
}
```

Файл pid.c

```
void pid_process(PID_t* pid)
{
    unsigned short temp = thermoCouple_get_temperature(pid->thermocouple);
    if(temp == TEMP_ERROR)
    {
        //Overheating protection
        pid->currentTemperature = 0;
        pid->expectedTemperature = 0;
    }
    else
        pid->currentTemperature = temp;

    float error = (pid->expectedTemperature)-(pid->currentTemperature);
//P

    pid->integral += error * pid->dt;
//I
    float integral = pid->integral;
    if(integral < 0)
        integral = 0;
    else if(integral > 100)
        integral = 100;

    float differential = (error-pid->previousError) / (pid->dt);
//D
    pid->previousError = error;

    int result = ((error * pid->kp));

    if(result < 0)
        result = 0;
    else if(result > 100)
        result = 100;
```

```
    heater_set_power_percentage(pid->heater, result);
}

void pid_init(PID_t* pidPtr, float kp, float ki, float kd, float dt,
Thermocouple_t* thermocouple, Heater_t* heater)
{
    PID_t pid =
    {
        .thermocouple = thermocouple,
        .heater = heater,
        .currentTemperature = 0,
        .expectedTemperature = 0,
        .kp = kp,
        .ki = ki,
        .kd = kd,
        .dt = dt,
        .previousError = 0,
        .integral = 0
    };

    *pidPtr = pid;
    heater_enable(pid.heater);
}
```

Файл command_handler.c

```
static void handle_get_pid_devices_number_cmd(unsigned char* data, unsigned
char data_length, command_response_t* response)
{
    if(data_length != 0)
        (*response).response_code = WRONG_ARGUMENTS_LENGTH;

    get_pid_devices_number_response responseModel = { .number = pidsNumber };
    char modelSize = sizeof(responseModel);

    (*response).response_code = OK;
    (*response).data_length = modelSize;
    memcpy((*response).data, &responseModel, modelSize);
}
```

```
static void handle_get_pid_current_temperature_cmd(unsigned char* data,
unsigned char data_length, command_response_t* response)
{
    if(data_length != sizeof(get_pid_current_temperature_request))
        (*response).response_code = WRONG_ARGUMENTS_LENGTH;

    get_pid_current_temperature_request* request =
(get_pid_current_temperature_request*)data;

    if(request->pid_id >= pidsNumber)
        (*response).response_code = WRONG_ARGUMENT_VALUE;

    get_pid_current_temperature_response responseModel = { .temperature =
pids[request->pid_id].currentTemperature };
    char modelSize = sizeof(responseModel);

    (*response).response_code = OK;
    (*response).data_length = modelSize;
    memcpy((*response).data, &responseModel, modelSize);
}

static void handle_get_pid_expected_temperature_cmd(unsigned char* data,
unsigned char data_length, command_response_t* response)
{
    if(data_length != sizeof(get_pid_expected_temperature_request))
        (*response).response_code = WRONG_ARGUMENTS_LENGTH;

    get_pid_expected_temperature_request* request =
(get_pid_expected_temperature_request*)data;

    if(request->pid_id >= pidsNumber)
        (*response).response_code = WRONG_ARGUMENT_VALUE;

    get_pid_expected_temperature_response responseModel = { .temperature =
pids[request->pid_id].expectedTemperature };
    char modelSize = sizeof(responseModel);

    (*response).response_code = OK;
    (*response).data_length = modelSize;
    memcpy((*response).data, &responseModel, modelSize);
}
```

```
static void handle_get_pid_coefficients_cmd(unsigned char* data, unsigned
char data_length, command_response_t* response)
{
    if(data_length != sizeof(get_pid_coefs_request))
        (*response).response_code = WRONG_ARGUMENTS_LENGTH;

    get_pid_coefs_request* request = (get_pid_coefs_request*)data;

    if(request->pid_id >= pidsNumber)
        (*response).response_code = WRONG_ARGUMENT_VALUE;

    get_pid_coefs_response responseModel =
    {
        .kp = pids[request->pid_id].kp,
        .ki = pids[request->pid_id].ki,
        .kd = pids[request->pid_id].kd
    };
    char modelSize = sizeof(responseModel);

    (*response).response_code = OK;
    (*response).data_length = modelSize;
    memcpy((*response).data, &responseModel, modelSize);
}

static void handle_set_pid_expected_temperature_cmd(unsigned char* data,
unsigned char data_length, command_response_t* response)
{
    if(data_length != sizeof(set_expected_temperature_request))
        (*response).response_code = WRONG_ARGUMENTS_LENGTH;

    set_expected_temperature_request* request =
(set_expected_temperature_request*)data;

    if(request->pid_id >= pidsNumber || request->temperature < 0 || request-
>temperature > 512)
        (*response).response_code = WRONG_ARGUMENT_VALUE;


    pids[request->pid_id].expectedTemperature = request->temperature;

    (*response).response_code = OK;
    (*response).data_length = 0;
}
```

```
static void handle_set_pid_kp_coef_cmd(unsigned char* data, unsigned char
data_length, command_response_t* response)
{
    if(data_length != sizeof(set_pid_kp_coef_request))
        (*response).response_code = WRONG_ARGUMENTS_LENGTH;

    set_pid_kp_coef_request* request = (set_pid_kp_coef_request*)data;

    if(request->pid_id >= pidsNumber || request->kp < 0 || request->kp > 100)
        (*response).response_code = WRONG_ARGUMENT_VALUE;

    set_pid_coefficient(PID_KP_ADDRESS, request->pid_id, request->kp);
    pids[request->pid_id].kp = request->kp;

    (*response).response_code = OK;
    (*response).data_length = 0;
}

static void handle_set_pid_ki_coef_cmd(unsigned char* data, unsigned char
data_length, command_response_t* response)
{
    if(data_length != sizeof(set_pid_ki_coef_request))
        (*response).response_code = WRONG_ARGUMENTS_LENGTH;

    set_pid_ki_coef_request* request = (set_pid_ki_coef_request*)data;

    if(request->pid_id >= pidsNumber || request->ki < 0 || request->ki > 100)
        (*response).response_code = WRONG_ARGUMENT_VALUE;

    set_pid_coefficient(PID_KI_ADDRESS, request->pid_id, request->ki);
    pids[request->pid_id].ki = request->ki;

    (*response).response_code = OK;
    (*response).data_length = 0;
}

static void handle_set_pid_kd_coef_cmd(unsigned char* data, unsigned char
data_length, command_response_t* response)
{
    if(data_length != sizeof(set_pid_kd_coef_request))
        (*response).response_code = WRONG_ARGUMENTS_LENGTH;
```

```
    set_pid_kd_coef_request* request = (set_pid_kd_coef_request*)data;

    if(request->pid_id >= pidsNumber || request->kd < 0 || request->kd > 100)
        (*response).response_code = WRONG_ARGUMENT_VALUE;

    set_pid_coefficient(PID_KD_ADDRESS, request->pid_id, request->kd);
    pids[request->pid_id].kd = request->kd;

    (*response).response_code = OK;
    (*response).data_length = 0;
}

command_response_t handle_command(unsigned short cmd, unsigned char* data,
unsigned char data_length)
{
    command_response_t response = { .response_code = WRONG_COMMAND,
.data_length = 0 };

    switch(cmd)
    {
        case GET_PID_DEVICES_NUMBER:
            handle_get_pid_devices_number_cmd(data, data_length, &response);
            break;
        case GET_PID_CURRENT_TEMPERATURE:
            handle_get_pid_current_temperature_cmd(data, data_length,
&response);
            break;
        case GET_PID_EXPECTED_TEMPERATURE:
            handle_get_pid_expected_temperature_cmd(data, data_length,
&response);
            break;
        case GET_PID_COEFS:
            handle_get_pid_coefficients_cmd(data, data_length, &response);
            break;
        case SET_PID_EXPECTED_TEMPERATURE:
            handle_set_pid_expected_temperature_cmd(data, data_length,
&response);
            break;
        case SET_PID_KP_COEF:
            handle_set_pid_kp_coef_cmd(data, data_length, &response);
            break;
        case SET_PID_KI_COEF:
            handle_set_pid_ki_coef_cmd(data, data_length, &response);
```

```
            break;
        case SET_PID_KD_COEF:
            handle_set_pid_kd_coef_cmd(data, data_length, &response);
            break;
        default:
        {
            response.response_code = WRONG_COMMAND;
            response.data_length = 0;
        }
    }

    return response;
}

void command_handler_init(PID_t* pidArray, unsigned char length)
{
    pids = pidArray;
    pidsNumber = length;
}
```

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2023 р.

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ ПІДТРИМКИ ДІЯЛЬНОСТІ ПАЯЛЬНОЇ СТАНЦІЇ

Програма та методика тестування

КПІ.ІТ-8105в.045490.04.51

“ПОГОДЖЕНО”
Керівник проєкту:
_____ Олена ХАЛУС

Нормоконтроль:
_____ Максим ГОЛОВЧЕНКО

Виконавець:
_____ Владислав ВЛАСЮК

Київ – 2023

ЗМІСТ

1 ОБ’ЄКТ ВИПРОБУВАНЬ

Об’єктом випробування є клієнтське програмне забезпечення для паяльної станції створене на платформі .NET з використанням фреймворку AvaloniaUI.

2 МЕТА ТЕСТУВАННЯ

Метою тестування є наступне:

– перевірка правильності роботи програмного забезпечення відповідно до функціональних вимог;

– перевірка збереження даних;

– перевірка сумісності застосунку з різними операційними системами (Windows, Linux);

– знаходження проблем, помилок і недоліків з метою їх усунення;

– перевірка зручності графічного інтерфейсу.

3 МЕТОДИ ТЕСТУВАННЯ

Для тестування програмного забезпечення використовуються такі методи:

– динамічне тестування – застосовується в процесі виконання програми. Коректність програмного засобу перевіряється на певній кількості тестів. При прогоні кожного з них збираються та аналізуються дані про проблеми та помилки в роботі програми;

– функціональне тестування – полягає у перевірці відповідності реальної поведінки програмного забезпечення очікуваній;

– системне тестування – перевіряється усе програмне забезпечення в цілому;

– мануальне тестування – тестування без використання автоматизації, тест-кейси пише особа, що тестує програмне забезпечення;

– тестування «чорної скриньки» – об’єктом тестування тут є функції присутні у програмі. Перевіряється коректність вихідних даних при заданих вхідних;

4 ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ

Тестування виконується мануально з використанням наскрізного тестування (End-to-end, E2E) з метою знаходження помилок та недоліків як у функціональній частині програмного забезпечення так і в зручності користування. Для того, щоб перевірити працездатність та відмовостійкість застосунку, необхідно провести наступні тестування:

– динамічне тестування на відповідність функціональним вимогам;

– тестування на персональних комп’ютерах з різною роздільно здатністю екрану;

– тестування на персональних комп’ютерах з різною операційною системою;

– тестування на виведення повідомлень про помилку, коли це необхідно;

– тестування інтерфейсу користувача на відповідність вимогам;

– тестування зручності використання;

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2023 р.

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ ПІДТРИМКИ ДІЯЛЬНОСТІ ПАЯЛЬНОЇ СТАНЦІЇ

Керівництво користувача

КПІ.ІТ-8105в.045490.05.34

“ПОГОДЖЕНО”
Керівник проєкту:
_____ Олена ХАЛУС

Нормоконтроль:
_____ Максим ГОЛОВЧЕНКО

Виконавець:
_____ Владислав ВЛАСЮК

Київ – 2023

ЗМІСТ

1 ПРИЗНАЧЕННЯ ПРОГРАМИ

«Soldering Station Client» - це настільний застосунок для підтримки діяльності паяльної станції, який дозволяє налаштувати її та вміє автоматично змінювати температуру нагрівачів станції за температурним профілем, що створив та обрав сам користувач програми.

Програмне забезпечення поширюється у вигляді виконуваних файлів для операційних систем Windows та Linux, має зручний та адаптивний інтерфейс, а також базу даних для збереження налаштувань та температурних профілів.

2 ПІДГОТОВКА ДО РОБОТИ З ПРОГРАМНИМ ЗАБЕЗПЕЧЕННЯМ

2.1 Системні вимоги для коректної роботи

Для успішної роботи даного застосунку необхідне виконання наступних мінімальних вимог:

– наявність комп'ютера або ноутбука з операційною системою на базі Windows 10-11, або Debian 9 та вище, або Ubuntu 16.04 та вище;

– не менше 300 мб вільного місця на жорсткому диску;

– не менше 4 Гб оперативної пам’яті;

– наявність інтегрованої або дискретної графіки;

– наявність монітору з роздільною здатністю не менше 1280x720.

Для комфортного використання застосунку необхідне виконання наступних вимог:

– наявність комп'ютера або ноутбука з операційною системою на базі Windows 10-11, або Debian 9 та вище, або Ubuntu 16.04 та вище;

– не менше 8 Гб оперативної пам’яті;

– не менше 1 Гб вільного місця на жорсткому диску;

– наявність монітору з роздільною здатністю не менше 1920x1080;

2.2 Завантаження застосунку

На даний момент застосунок можна встановити власноруч, використовуючи відповідний виконуваний файл. Для цього спершу необхідно його завантажити на комп’ютер та запустити.

2.3 Перевірка коректної роботи

Після завершення завантаження додатка його необхідно перемістити у зручну папку, в ній після запуску буде автоматично створено файл бази даних. Користувач має змогу запустити додаток, клацнувши на виконуваний файл. Після натискання повинно відобразитися головне вікно застосунку.

3 ВИКОНАННЯ ПРОГРАМИ

При запуску клієнтського програмного застосунку користувачу буде відображено головне вікно програми, в якому знаходиться блок з вибором та налаштуванням порту для підключення, графіком температур, кнопкою для управління температурними профілями та кнопкою відкриття вікна вибору та запуску режиму роботи за температурним профілем (рисунок 3.1).

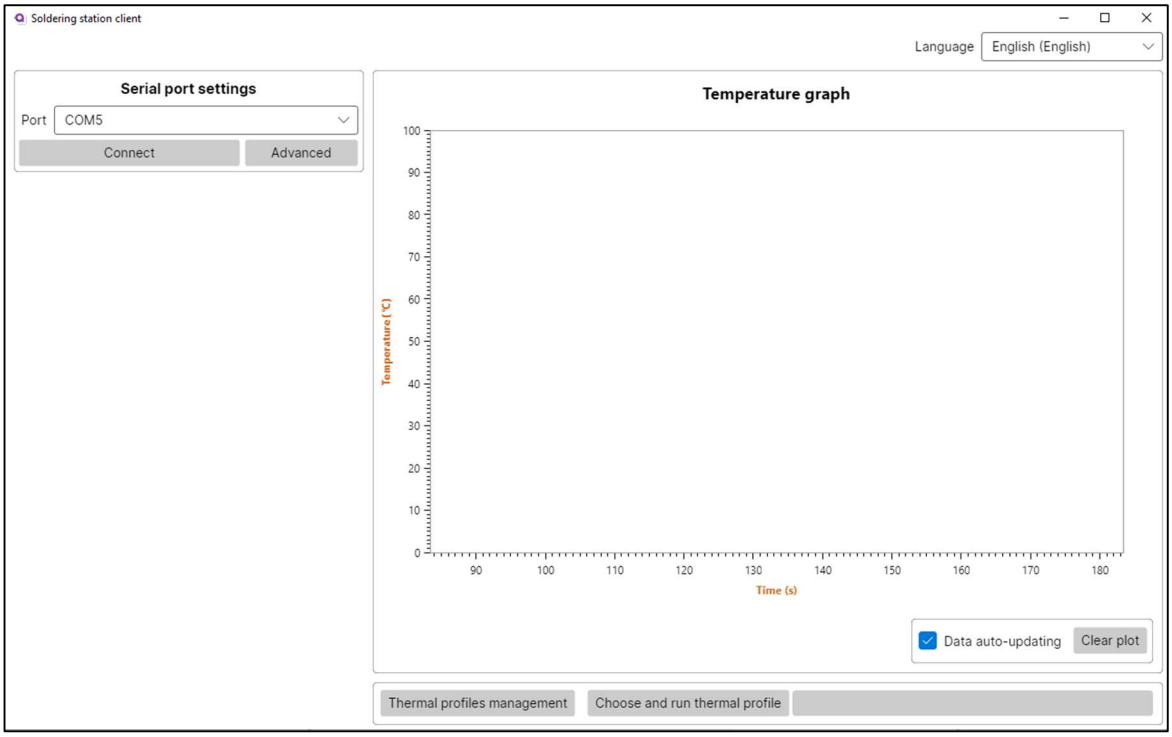


Рисунок 3.1 – Головне вікно додатку

Користувач має змогу обрати порт з випадаючого списку та натиснути кнопку «Connect». Якщо пристроїв в списку немає, кнопка залишається неактивною (рисунок 3.2). Після підключення нового порту (це може бути USB-TTL перетворювач) вікно знову стане активним.

Рисунок 3.2 – Підключених портів немає, кнопки неактивні

Після появи порту в системі можна або підключитися до пристрою, або перейти до додаткових налаштувань порту. На рисунку 3.2 зображено

налаштування для контролера, який розроблявся разом з клієнтським програмним забезпеченням у даній роботі. Після натискання кнопки «Apply» усі налаштування порту будуть збережені в базу даних, і при підключенні цього ж пристрою до цього ж порту не доведеться більше змінювати налаштування.

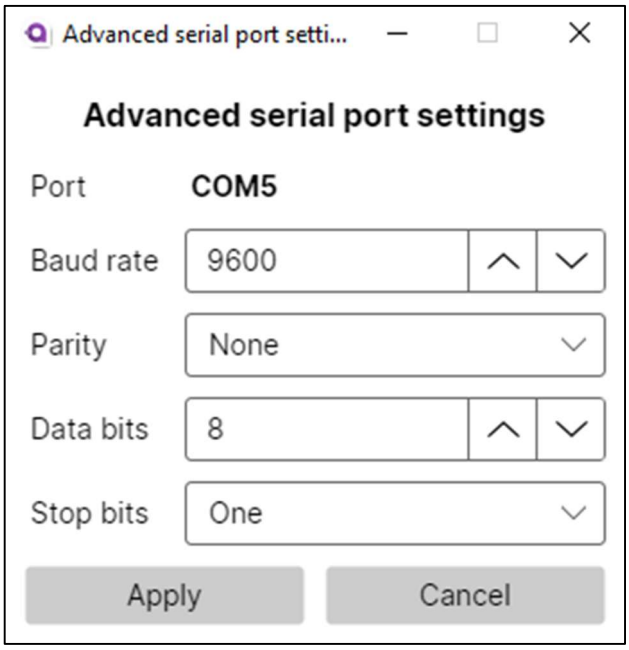


Рисунок 3.3 – Додаткові налаштування порту

Після натискання кнопки «Connect» може з’явитися вікно з помилкою, зображене на рисунку 3.3. Вона може означати одну з трьох речей:

– обрано невірні налаштування порту;

– до обраного порту не під’єднано контролер, який підтримується програмою;

– контролер або USB-TTL адаптер вийшов з ладу (якщо використовується).

Рисунок 3.4 – Помилка підключення

Після успішного підключення контролер станції автоматично розпізнається та з’явиться у списку пристроїв. На рисунку 3.4 видно головне вікно програми з підключеним контролером.

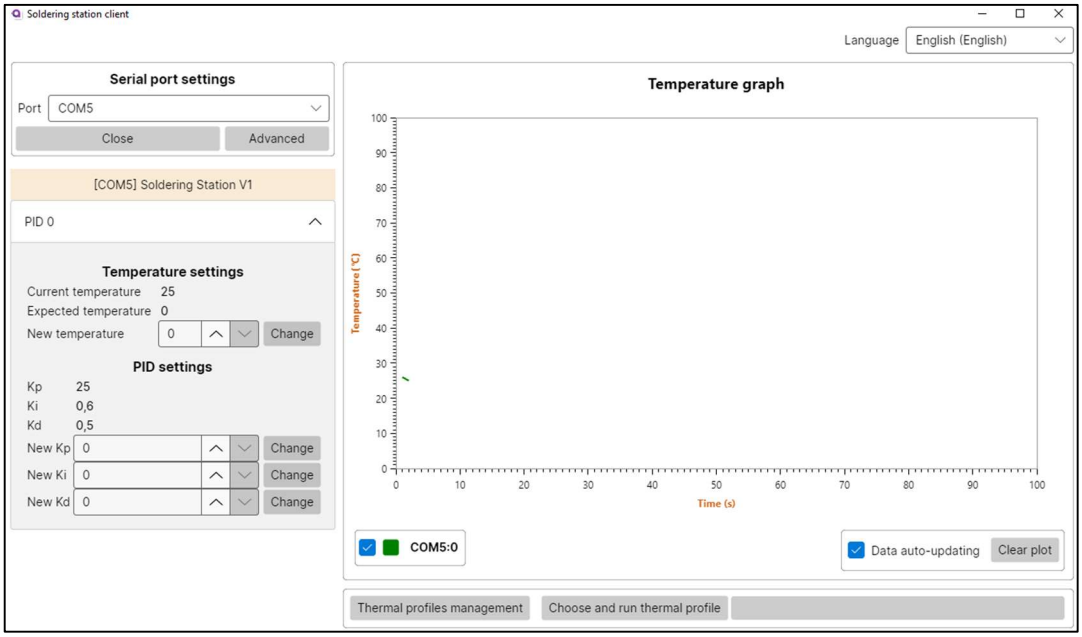


Рисунок 3.5 – Головне вікно з підключеним контролером

Після підключення можна змінити температуру або коефіцієнти ПІД регулятора, якщо підключений контролер підтримує таку функцію. Після зміни температури на графіку з’явиться показник, який це відображає (рисунок 3.5).

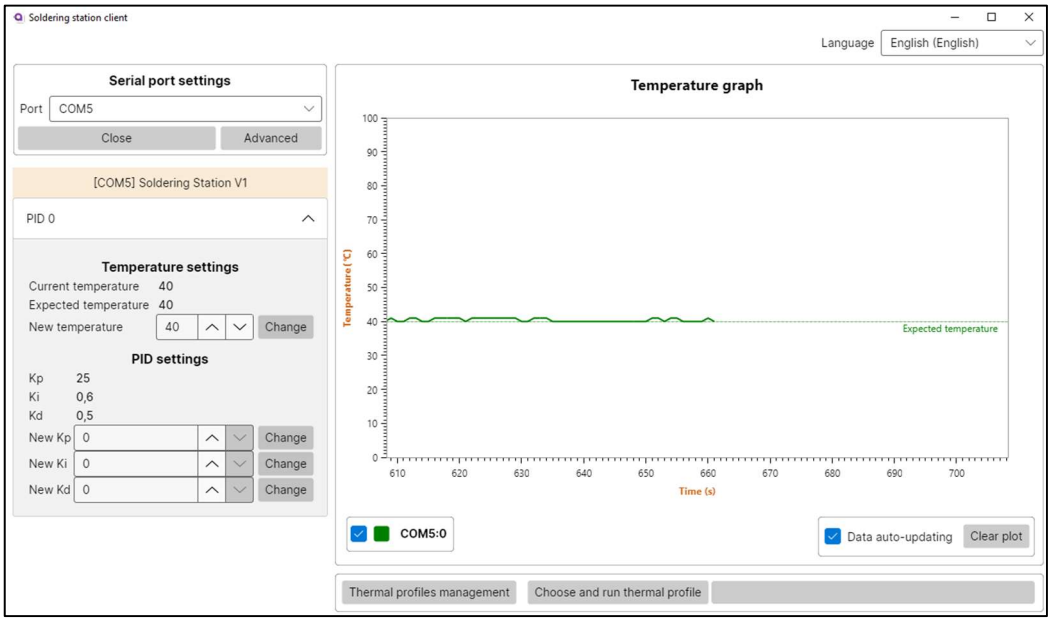


Рисунок 3.6 – Встановлено статичну температуру нагріву

Якщо ввести неправильне значення, то поле буде обведено червоним та можна буде переглянути помилку, якщо навестися на знак оклику поряд з полем (рисунок 3.6).

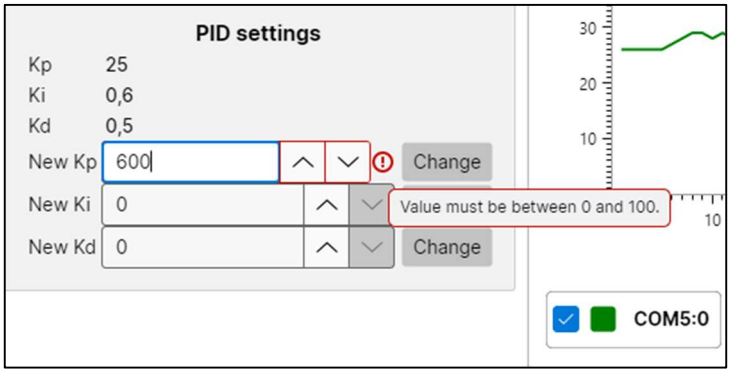


Рисунок 3.7 – Помилка введення

Для зміни кольору конкретного контролера температури необхідно натиснути на кнопку навпроти назви контролера під графіком температур (рисунок 3.7). Щоб приховати дані з контролера на графіку, необхідно зняти відмітку з чекбоксу поряд з кнопкою зміни кольору. Для того, щоб зупинити опитування контролера, необхідно зняти відповідну відмітку з чекбоксу під графіком. При встановленні її назад графік буде автоматично очищено. Для простої очистки графіку також є відповідна кнопка.

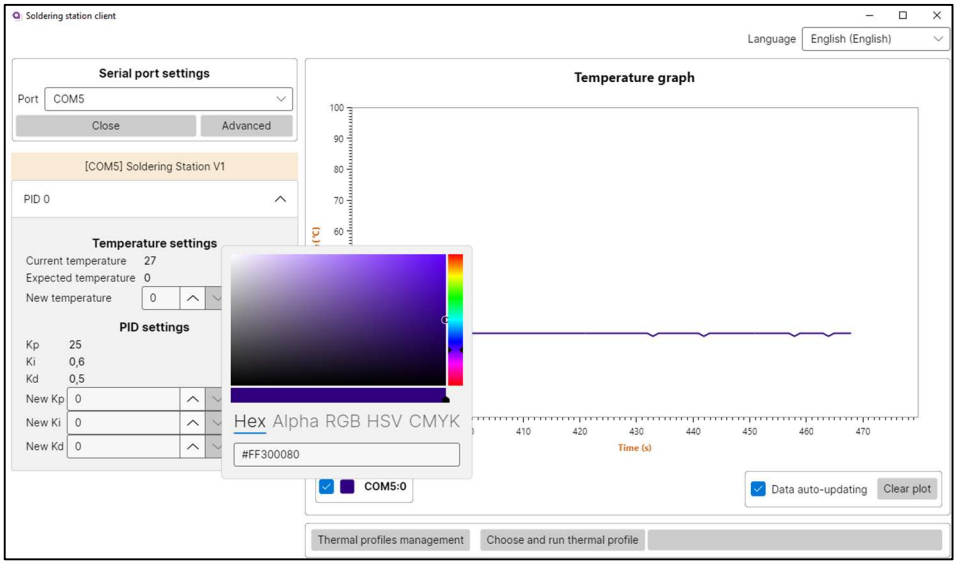


Рисунок 3.8 – Зміна кольору контролера

Програма також надає можливості для створення, редагування та видалення температурних профілів. Щоб перейти до управління

температурними профілями необхідно натиснути кнопку «Thermal profile management». Після цього відкриється вікно з редактором (рисунок 3.8).

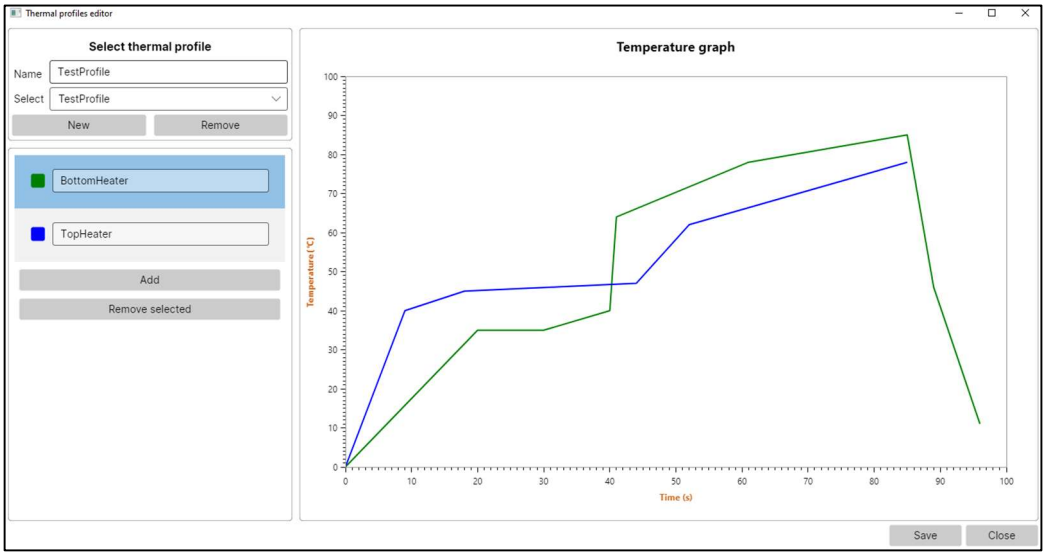


Рисунок 3.9 – Редактор температурних профілів

В панелі під назвою «Select thermal profile» можна обрати існуючий температурний профіль, створити новий або видалити обраний температурний профіль. В полі «Name» можна змінити назву обраного температурного профіля. Нижче знаходиться список усіх кривих, які можна створювати, видаляти та обирати. Кожна крива в подальшому буде прикріплена до певного контролера температури. Для зміни кольору необхідно натиснути кнопку поряд з назвою кривої, також можна змінити саму назву, відредагувавши її. Для додавання нової точки на графік потрібно натиснути лівою кнопкою миші на вільному місці на графіку. Щоб видалити точку, потрібно натиснути на неї також лівою кнопкою миші. Усі зміни зберігаються тільки після натискання кнопки «Save».

Після натискання в головному вікні кнопки «Choose and run thermal profile» відкриється вікно (рисунок 3.9), в якому можна обрати температурний профіль та прикріпити контролери, які будуть працювати за цим профілем. Якщо кількість підключених контролерів недостатня, то кнопка «Apply» буде неактивною. В такому разі потрібно обрати або інший температурний профіль, або підключити необхідну кількість контролерів.

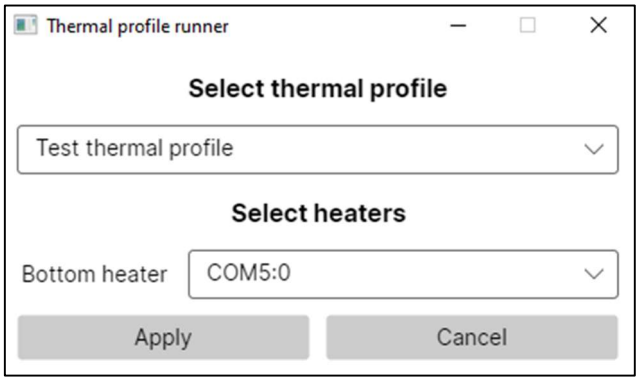


Рисунок 3.10 – Вікно запуску роботи за температурним профілем

Після запуску роботи за температурним профілем усі функції заблокуються, можна буде лише відмінити завдання. На графіку буде видно температурний профіль та контролери, які виконують завдання (рисунок 3.10).

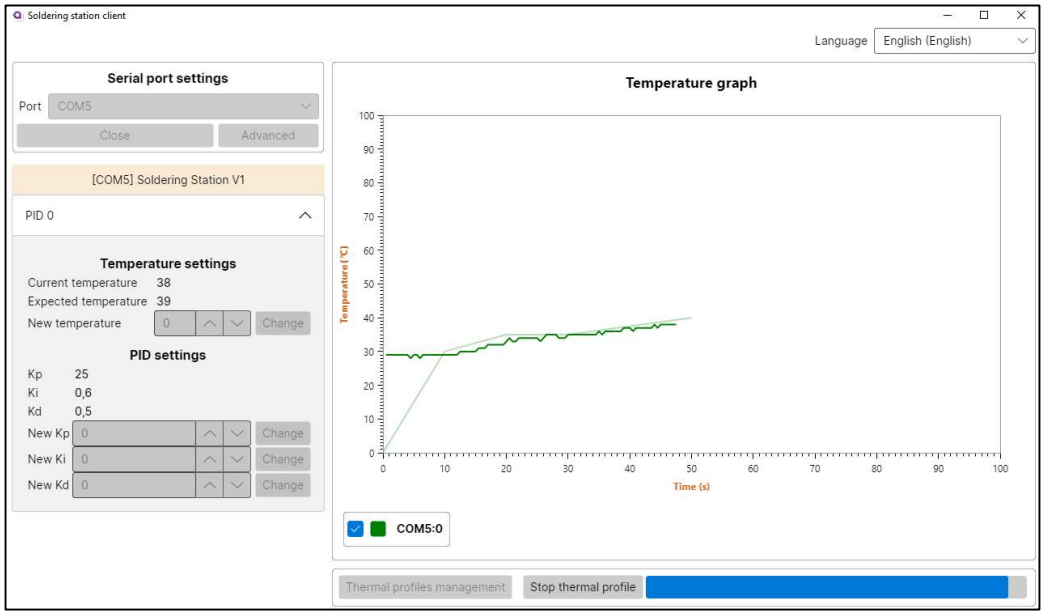


Рисунок 3.11 – Виконання роботи за температурним профілем

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2023 р.

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ ПІДТРИМКИ ДІЯЛЬНОСТІ ПАЯЛЬНОЇ СТАНЦІЇ

Графічний матеріал

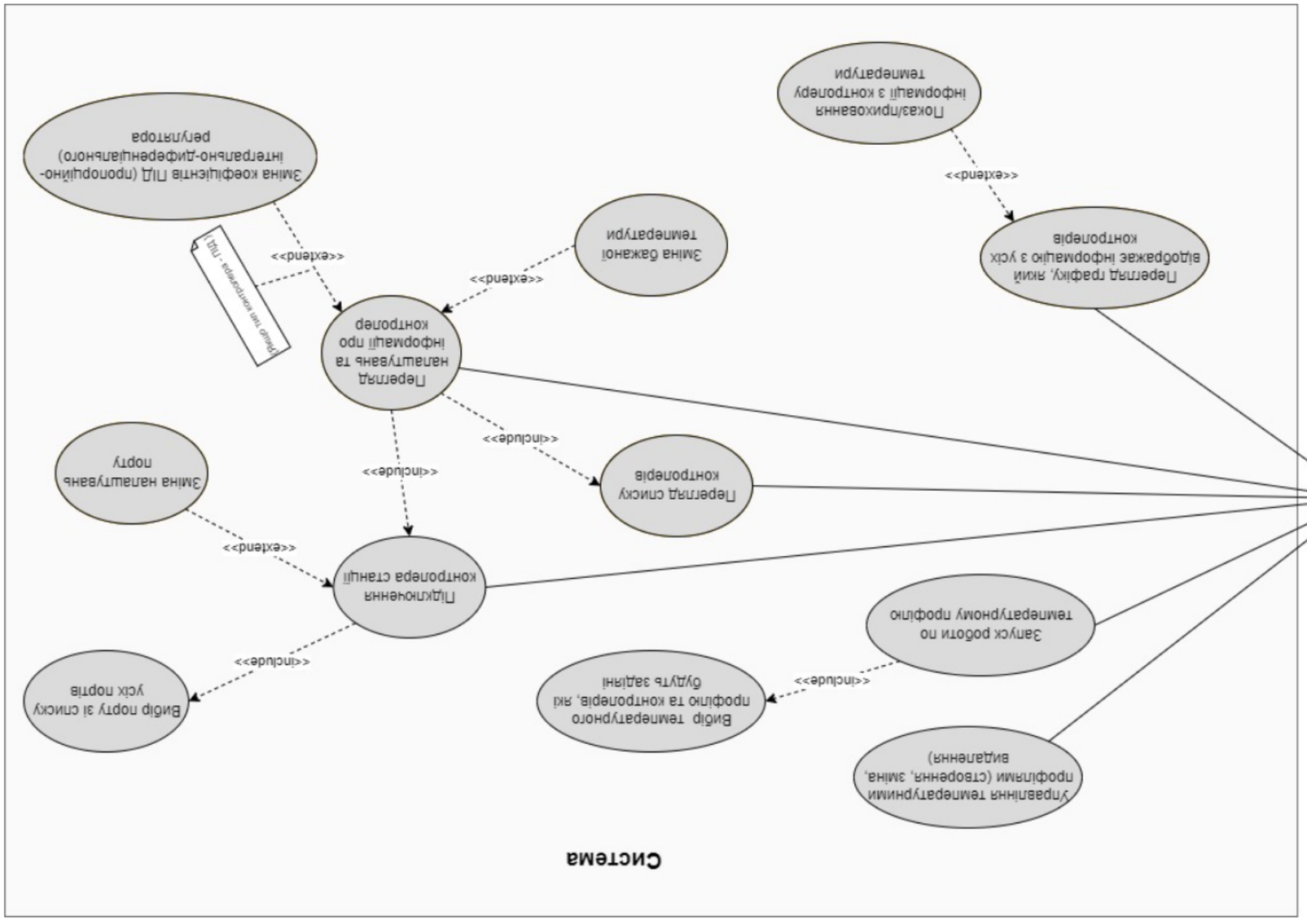
КПІ.ІТ-8105в.045490.06.99

“ПОГОДЖЕНО”
Керівник проєкту:
_____ Олена ХАЛУС

Нормоконтроль:
_____ Максим ГОЛОВЧЕНКО

Виконавець:
_____ Владислав ВЛАСЮК

Київ – 2023



Зм.	Арк.	№ документа	Підпис	Дата							
Розробив		Власюк В.В.									
Перевірив		Харус О.А.									
Т. контр.											
Н. контр.		Голоденко М.М.									
Затвердив		Жарікова Е.В.									

КПІ.ІТ-8105в.045490.06.99.ССВ

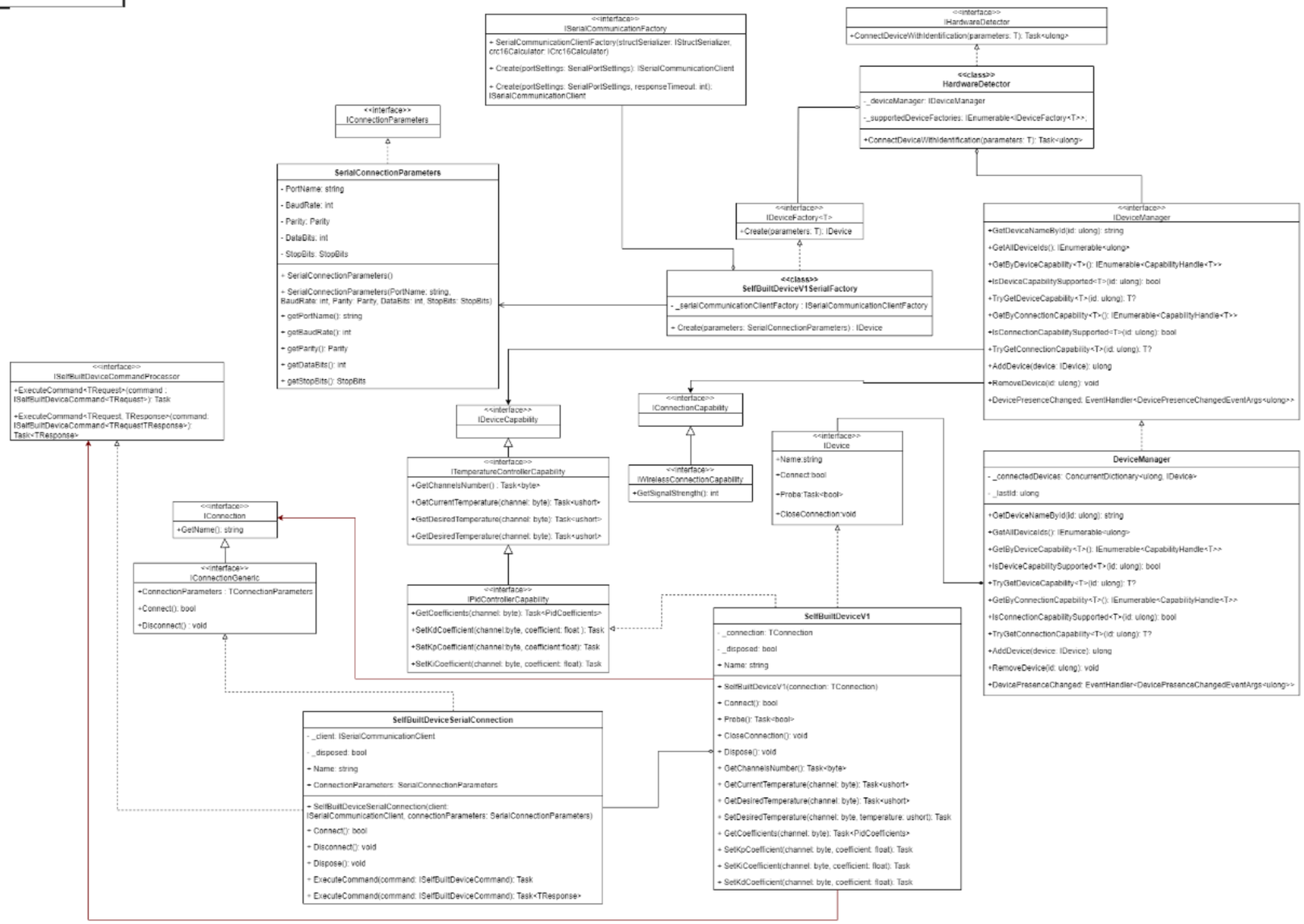
Схема структура варіантів використань

Літера	Маса	Масштаб

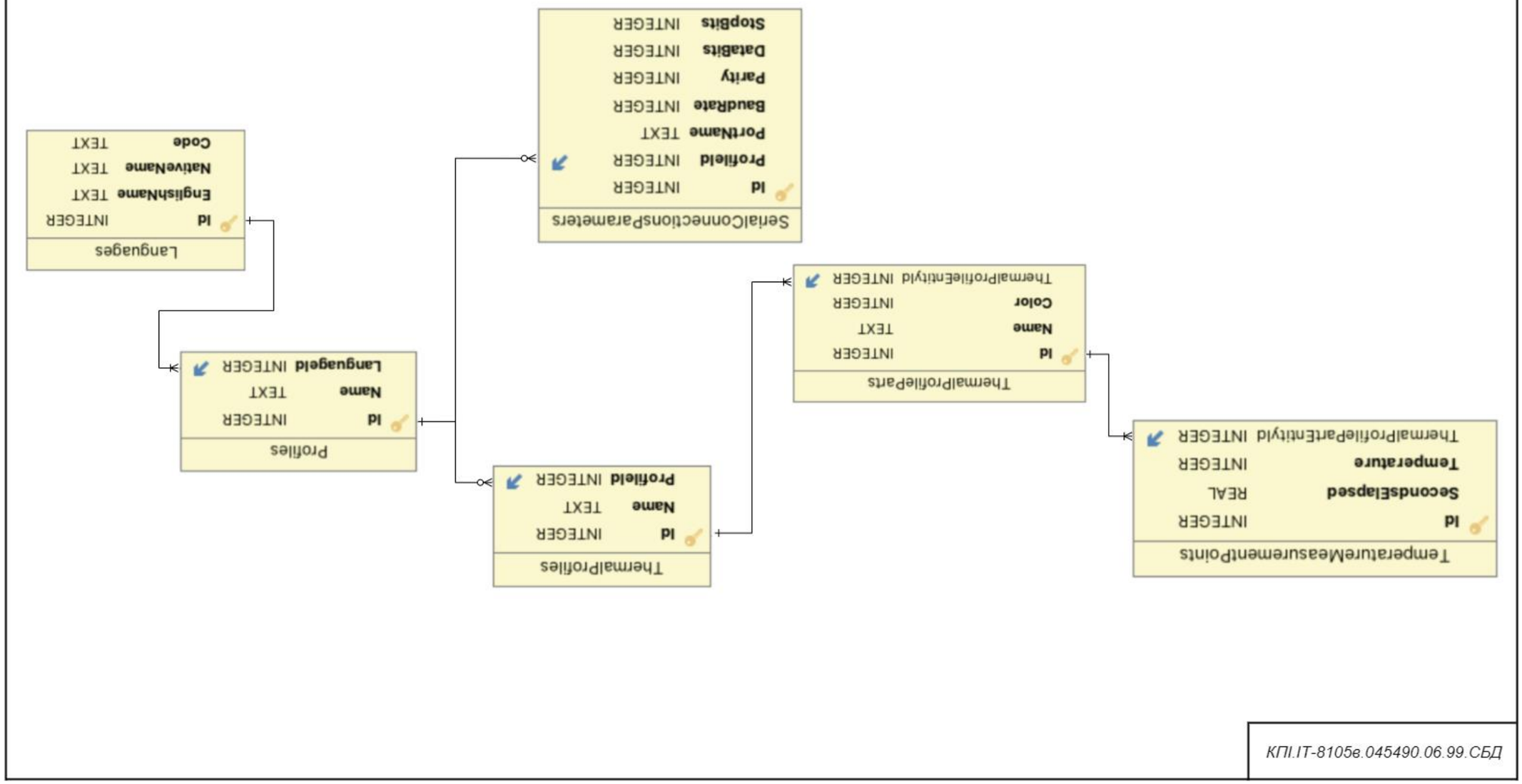
Аркуш	Аркушів

Програмне забезпечення для підтримки діяльності лабораторії станції

КПІ ім.Ігоря Сікорського
Кафедра ІПІ
гр. ІТ-81в



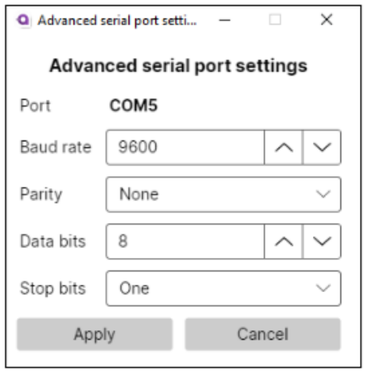
					КПІ.ІТ-8105в.045490.06.99.ССК	
						Літера / Маса / Масштаб
Зм.	Арк.	№ документа	Підпис	Дата	Схема структурна класів програмного забезпечення	
Розробив		Власюк В.В.				Аркуш / Аркушів
Перевірив		Халус О.А.				
Т. кон.					Програмне забезпечення для підтримки діяльності паяльної станції	КПІ ім.Ігоря Сікорського Кафедра ІПІ гр. ІТ-81в
Н. кон.		Головченко М.М.				
Затвердив		Жаріков Е.В.				

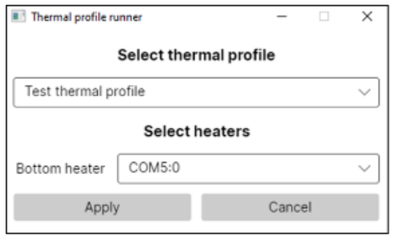


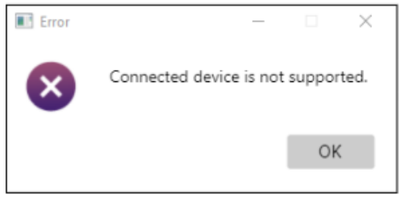
КПІ.ІТ-8105в.045490.06.99.СБД

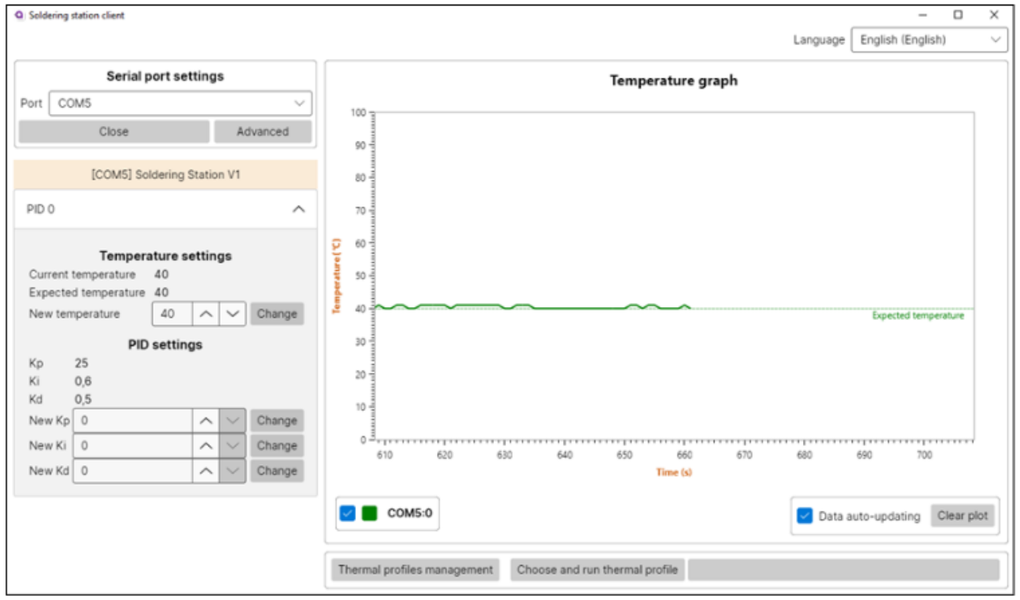
Зм.	Арк.	№ документа	Підпис	Дата	КПІ.ІТ-8105в.045490.06.99.СБД			
Розробив		Вацюк В.В.			Схема бази даних	Літера	Маса	Масштаб
Перевірив		Харус О.А.						
Т. контр.						Аркуш	Аркушів	
Н. контр.		Голоденко М.М.			Програмне забезпечення для підтримки діяльності гальванічної станції	КПІ ім.Ігоря Сікорського кафедра ІПІ гр. ІТ-818		
Затвердив		Жарікова Е.В.						

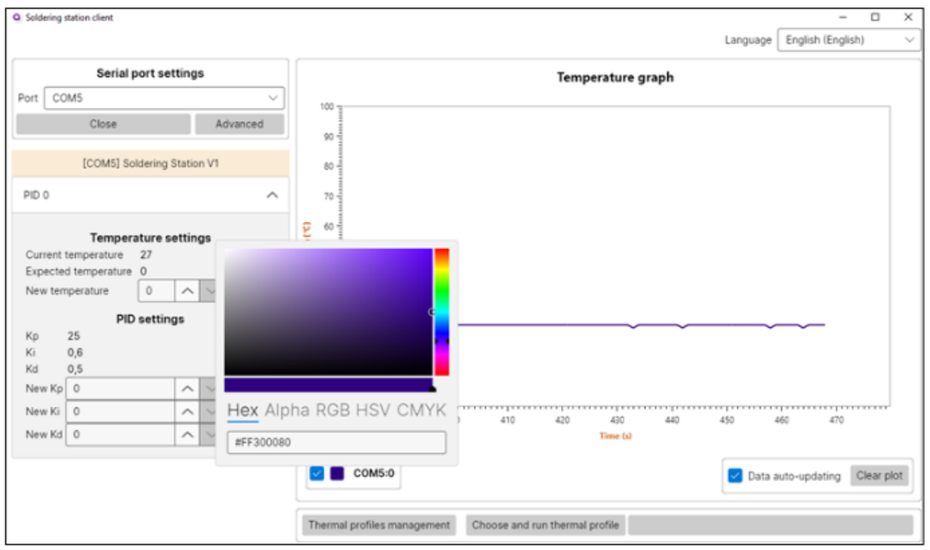
КПІ.ІТ-8105в.045490.06.99.КЕ

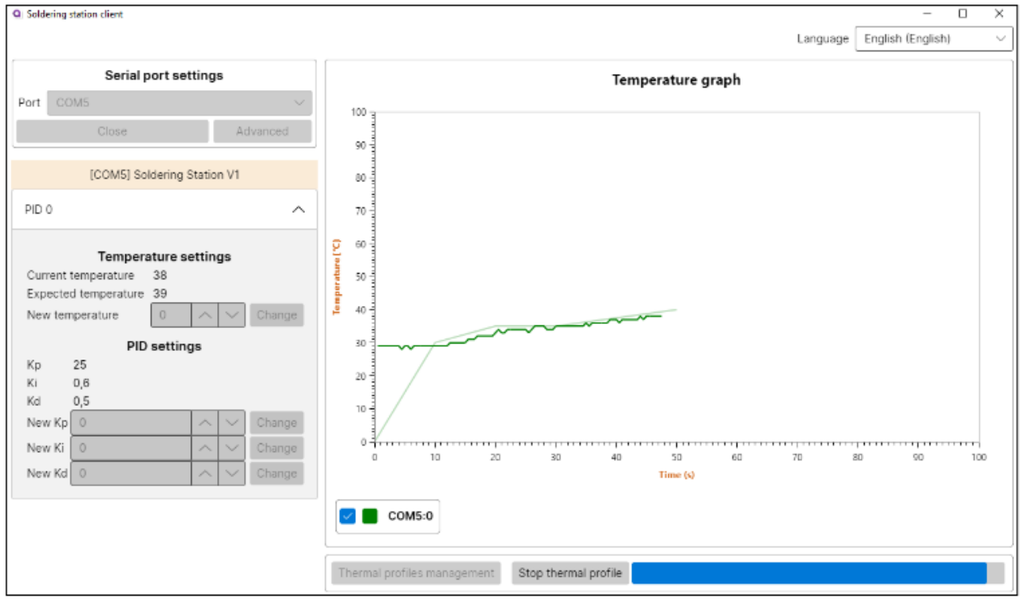


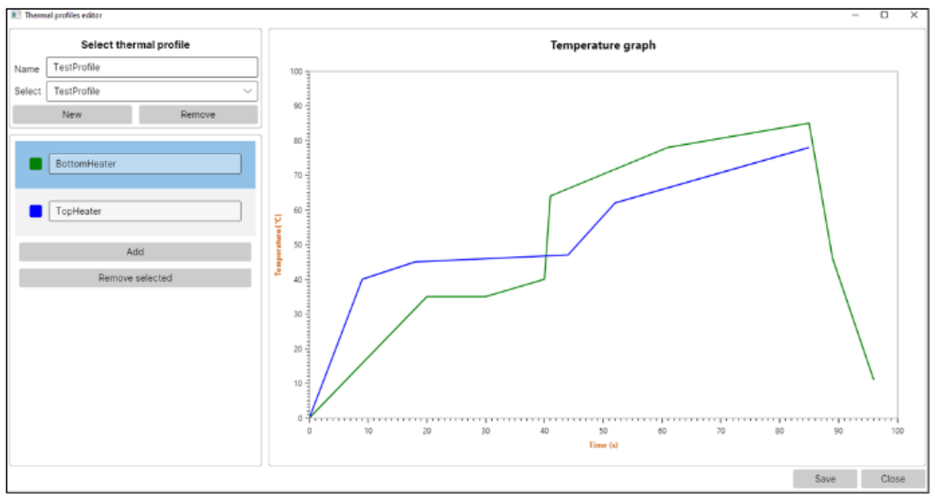












					КПІ.ІТ-8105в.045490.06.99.КЕ		
Зм.	Арк.	№ документа	Підпис	Дата	Креслення вигляду екранних форм	Літера	Маса / Масштаб
Розробив		Власюк В.В.					
Перевірив		Халус О.А.				Аркуш	Аркушів
Т. кон.							
					Програмне забезпечення для підтримки діяльності паяльної станції	КПІ ім.Ігоря Сікорського Кафедра ІПІ гр. ІТ-81в	
Н. кон.		Головченко М.М.					
Затвердив		Жаріков Е.В.					