ДИПЛОМНА РОБОТА
БАКАЛАВРА

ДРБ. ІП - 18.00.00.000 ПЗ

Група ІП-18-2

Єввa Павло

2022

Міністерство освіти і науки України
Івано-Франківський національний технічний університет нафти і газу

Інститут інформаційних технологій

Кафедра інженерії програмного забезпечення

УДК 004.4

ДИПЛОМНА РОБОТА БАКАЛАВРА

Тема: " Розробка програмного сервісу обробки метрологічних даних"

(назва згідно з наказом ректора)

Спеціальність – 121 "Інженерія програмного забезпечення"

ПОЯСНЮВАЛЬНА ЗАПИСКА

ДРБ.ІП - 18.00.00.000 ПЗ

(позначення)

Консультанти:

Студент

ІП-18-2 /П.Є.Єввa/
(шифр групи) (підпис) (дата) (розшифровка підпису)

Керівник роботи

асист. /Л.В.Саманів/
(посада) (підпис) (дата) (розшифровка підпису)

з питань нормоконтролю

Допускається до захисту
Завідувач кафедри

доц., к.т.н /В.Я. Піх/
(посада) (підпис) (дата)(розшифровка підпису)

д.т.н.,проф. /В.І.Шекета/
(посада) (підпис) (дата) (розшифровка підпису)

2022
(рік)

Івано-Франківський національний технічний університет нафти і газу

Інститут, факультет інформаційних технологій

Кафедра інженерії програмного забезпечення

Ступінь вищої освіти бакалавр

Спеціальність 121 – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ:
Зав. кафедрою **ІПЗ**

проф. В.І. Шекета

“ ____ ” _____2022 р.

ЗАВДАННЯ

НА ДИПЛОМНУ РОБОТУ БАКАЛАВРА СТУДЕНТОВІ

Євві Павлу Євгеновичу

(прізвище, ім’я, по-батькові)

1. Тема проекту (роботи) “ Розробка програмного сервісу обробки метрологічних даних”

керівник проекту (роботи) Саманів Любов Василівна

затверджені наказом закладу вищої освіти від **“** ___ ” травня 2022 р. № _____

2. Строк подання студентом проекту (роботи) 17 червня 2022 р.

3. Вихідні дані до проекту (роботи) Результати і матеріали отримані під час проходження переддипломної практики

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

1. Огляд сучасних методів та систем метрології
2. Розробка алгоритмічного забезпечення системи
3. Розробка програмного забезпечення програмного рішення
4. Представлення інтерфейсу програмного сервісу
5. Тестування програмного продукту

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

1. Форма запуску програми повірки лічильників газу
2. Форма початкової ініціалізації програми
3. Форма вводу параметрів лічильника
4. Основна форма програми
5. Форма встановлення витрати

6. Консультанти розділів проекту (роботи)

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання **25 травня 2022 р.**

Керівник _____
(підпис)

Завдання прийняв до виконання_____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту	Примітка
1	Огляд сучасних методів та систем метрології побутового газу повірки газових лічильників	27.05.2022	виконано
2	Розробка алгоритмічного забезпечення системи	30.05.2022	виконано
3	Розробка програмного забезпечення для повірки газових лічильників	02.06.2022	виконано
4	Опис інтерфейсу розробленого програмного продукту	06.06.2022	виконано
5	Виконання тестування розробленого програмного забезпечення	11.06.2022	виконано
6	Оформлення пояснювальної записки дипломної роботи завідувачем кафедри	17.06.2022	виконано

Студент – дипломник _____
(підпис)

Керівник роботи _____
(підпис)

АНОТАЦІЯ

В дипломній роботі було розроблено програмне забезпечення на основі ресурсів метрології побутового газу для процесу комп’ютерної повірки промислових лічильників газу, яке дозволяє полегшити процес повірки шляхом керування апаратною частиною системи, формуванням та друком протоколів повірки. Дана програмна розроблена за допомогою програмного пакету C++ Builder.

КЛЮЧОВІ СЛОВА: ГАЗОВИЙ ЛІЧИЛЬНИК, МНЕМО-СХЕМА, ПРОГРАМНА СИСТЕМА, АЛГОРИТМ РОБОТИ, КАЛІБРУВАННЯ, ЕТАЛОННИЙ ЛІЧИЛЬНИК.

ANNOTATION

In the thesis the software on the basis of resources of metrology of domestic gas was developed for the process of computer verification of industrial gas meters, which facilitates the verification process by controlling the hardware part of the system, forming and printing verification protocols. This software was developed using the C ++ Builder software package.

KEYWORDS: GAS METER, MNEMONIC SCHEME, SOFTWARE SYSTEM, ALGORITHM OF OPERATION, CALIBRATION, REFERENCE COUNTER.

ЗМІСТ

ПЕРЕЛІК ОСНОВНИХ ПОЗНАЧЕНЬ І СКОРОЧЕНЬ ……..... 7

ВСТУП…… 8

1 ОГЛЯД СУЧАСНИХ МЕТОДІВ ТА СИСТЕМ МЕТРОЛОГІЇ ТА ПОВІРКИ

1.1 Актуальність програмної розробки…… 9

1.2 Опис дзвонових витратовимірювальних установок ……..... 11

1.3 Переваги та недоліки витратовимірювальних установок на основі еталонних лічильників ……. 13

1.4 Опис та структура робочої схеми метрологічної повірки ……... 14

1.5 Аналіз систем повірки лічильників газу та інформаційно-метрологічних комплексів …….. 17

1.6 Класифікація програмного забезпечення обробки метрологічних даних.. 29

2 РОЗРОБКА АЛГОРИТМІЧНОГО ЗАБЕЗПЕЧЕННЯ СИСТЕМИ

2.1 Формування вимог до алгоритму……..... 34

2.2 Розробка алгоритму функціонування програми……..... 34

3 РОЗРОБКА ПРОГРАМНОГО СЕРВІСУ ОБРОБКИ МЕТРОЛОГІЧНИХ ДАНИХ

3.1 Вибір середовища розробки програмного сервісу..…..……. 43

3.2 Реалізація блоків алгоритму……..... 46

ВИСНОВКИ…….. 60

ПЕРЕЛІК ПОСИЛАНЬ НА ДЖЕРЕЛА…… 61

ДОДАТКИ

БІБЛІОГРАФІЧНА ДОВІДКА

					ДРБ. ІП - 18.00.00.000 ПЗ			
Змн.	*Лист*	*№ документа*	*Підпис*	*Дата*				
Розробив		Єввa П.Є			Розробка програмного сервісу обробки метрологічних даних	Літ.	Арк.	Акрушів
Перевірив		Саманів Л.В.				Н	6	
Реценз.					**Пояснювальна записка**	ІФНТУНГ ІП-18-2		
Н. Контр.		Піх В.Я.						
Затверд.		Шекета В.І.						

ПЕРЕЛІК ОСНОВНИХ ПОЗНАЧЕНЬ І СКОРОЧЕНЬ

ВУ - витратовимірювальна установка,

ЛГ - лічильник газу,

ПЗ - програмне забезпечення,

ПЛГ – побутовий лічильник газу.

ВСТУП

Природний газ — є одним з видів стратегічної сировини, а тому є об'єктом постійної уваги Уряду. Ще в роки розбудови України Кабінетом Міністрів України була прийнята Постанова від квітня 1993р. "Про заходи щодо зменшення обсягів споживання паливо–енергетичних ресурсів", а згодом була затверджена „Концепція створення єдиної системи обліку природного газу" серпень 2001 р. Цими документами зобов'язано удосконалити систему обліку газу на всіх її рівнях.

В результаті дипломної роботи розроблена діюча програма, яка може бути частиною програмно-апаратного комплексу повірки промислових лічильників газу.

Програма повірки промислових лічильників газу призначена для управління, контролю установки, а також для обробки і зберігання даних, отриманих в процесі експлуатації Установки.

Програмне забезпечення повірки промислових лічильників газу повинне виконувати наступні функції:

- збір, реєстрацію, обробку і відображення даних з давачів температури;
- збір, реєстрацію, обробку і відображення даних з давачів тиску;
- реєстрацію та підрахунок імпульсів з робочих еталонів;
- реєстрацію та підрахунок імпульсів з лічильників, що повіряються;
- перетворення підрахованих імпульсів в об'єми;
- комутацію необхідних для поточної роботи давачів;
- формування сигналів управління регулятором витрати та запірною арматурою;
- індикації положення запірної арматури та регулятора витрати;
- обчислення похибки;
- формування, збереження та друк протоколів.

1 ОГЛЯД СУЧАСНИХ МЕТОДІВ ТА СИСТЕМ МЕТРОЛОГІЇ ТА ПОВІРКИ

1.1. Актуальність програмної розробки

Аналіз причин втрат природного газу і розробка шляхів його цільового економного використання були і залишаються об'єктом прискіпливої уваги багатьох дослідників, про що свідчать численні наукові публікації, у тому числі у провідному журналі нафтогазової галузі.

Детальний аналіз структури газоспоживання по регіонах України з урахуванням напрямків використання газу показав, що досягненню мети більш раціонального використання ресурсів природного газу та зменшенню його втрат слугували такі організаційні заходи:

- посилення контролю за цільовим використанням природного газу;
- встановлення нових та заміна застарілих побутових лічильників природного газу;
- удосконалення приладного обліку газу;
- впровадження нормативної бази для обрахування питомих втрат та виробничо-технологічних витрат газу;
- зменшення фізичних втрат природного газу в розподільних газопроводах.

Відомо, що одним з найбільш енергоємних і одночасно найбільш недисциплінованих в питаннях споживання газу та його плати є підприємства комунальної теплоенергетики. В зв'язку з цим було встановлено суворий режим контролю за поставками газу цим організаціям. Ліміти споживання газу для підприємств теплокомуненерго почали встановлювати не відразу на весь звітний місяць, а окремо на кожну його декаду в залежності від того, як іде процес оплати поставленого газу. Такий захід виявився дієвим відносно тепловиків в пошуках засобів для проведення своєчасних розрахунків за спожитий газ.

Зупинимось детально на актуальності застосувань побутових лічильників газу.

Одним з основних критеріїв дієвості такого заходу, як встановлення побутових лічильників природного газу, крім суто суб'єктивних причин, пов'язаних з намаганнями самого абонента скоротити свої витрати газу, є зменшення так званих комерційних втрат газу.

Згаданий вид втрат спостерігається при поставках природного газу населенню і пояснюється виникненням від'ємної різниці між обсягами газу, обчисленими та оплаченими за затвердженими нормами споживання та використаними фактично.

Водночас останніми роками все більшої актуальності набуває питання повірки побутових лічильників газу (ПЛГ), так як багато з них вже відпрацювали регламентований чинними документами України термін (5 років) до моменту обов'язкової перевірки органами Держстандарту їх метрологічних характеристик. Однак практична реалізація перевірки вимагає значних матеріальних затрат на монтаж-демонтаж приладів і часових затрат на їх проведення, а кількість перевірочних установок не задовольняє потребам споживачів. Водночас у виробничих і наукових колах з все більшою гостротою дискутується питання про плинність метрологічних характеристик ПЛГ в проміжку п'ятирічного міжперевірочного терміну, так як ця обставина є одним із чинників комерційних втрат газопостачальних організацій і безпосередньо впливає на раціональне використання природного газу. Тому актуальним є питання контролю метрологічних характеристик ПЛГ в будь-який момент експлуатації безпосередньо у споживачів, яке можна вирішити, наприклад їх діагностуванням.

Незважаючи про відсутність в даний час нормативного документа на проведення технічного діагностування загальновідомими є як технічні Принципові розробки і напрацювання, так і нормативна база. Однак, практичної реалізації набув тільки метод, регламентований „Інструкцією щодо обслуговування та експрес-контролю побутових лічильників газу...".

Функціональна схема таких установок для експрес-контролю ПЛГ передбачає послідовне під'єднання з перевірюваним ПЛГ контрольного приладу за допомогою спеціального технологічного устаткування, яке встановлюється при демонтажі осердя перекривного вентиля підвідного газопроводу індивідуального газоспоживача. Згідно цієї схеми роль джерела витрати виконує лінія газопостачання, а відлічений перевірюваним і контрольним (еталонним) приладами природний газ спалюється газоспоживачами. Решта вузлів перевірочної установки відповідають схемі випробувань і додатково конструктивно можуть передбачати контроль температури і тиску газу у регламентованих точках та вентильний задавач відтворюваних витрат. При перевірці здійснюється відлік і порівняння виміряного об'єму газу двома лічильниками на витратах, які регламентовані відповідними нормативними документами.

Головним недоліком практичної реалізації цього методу є складність спеціального технологічного перекривного обладнання замість вхідного перекривного вентиля, яке повинне забезпечувати подачу газу після контрольного лічильника в магістралі перед газоспоживачем. Водночас ця схема характеризується складністю регулювання відтворюваних через ПЛГ контрольованих витрат газу і, як правило, відсутньою практичною можливістю проведення експрес-контролю на великих витратах газу в силу обмеженої продуктивності щодо спалювання газу обладнанням газоспоживача.

1.2 Опис дзвонових витратовимірювальних установок

Еталонні установки дзвонового типу використовуються для градуювання та повірки витратомірів та лічильників газу, для точного відтворення і вимірювання об’ємної витрати газу.

Будова випробувальної установки дзвонового типу показана на рисунку 1.1

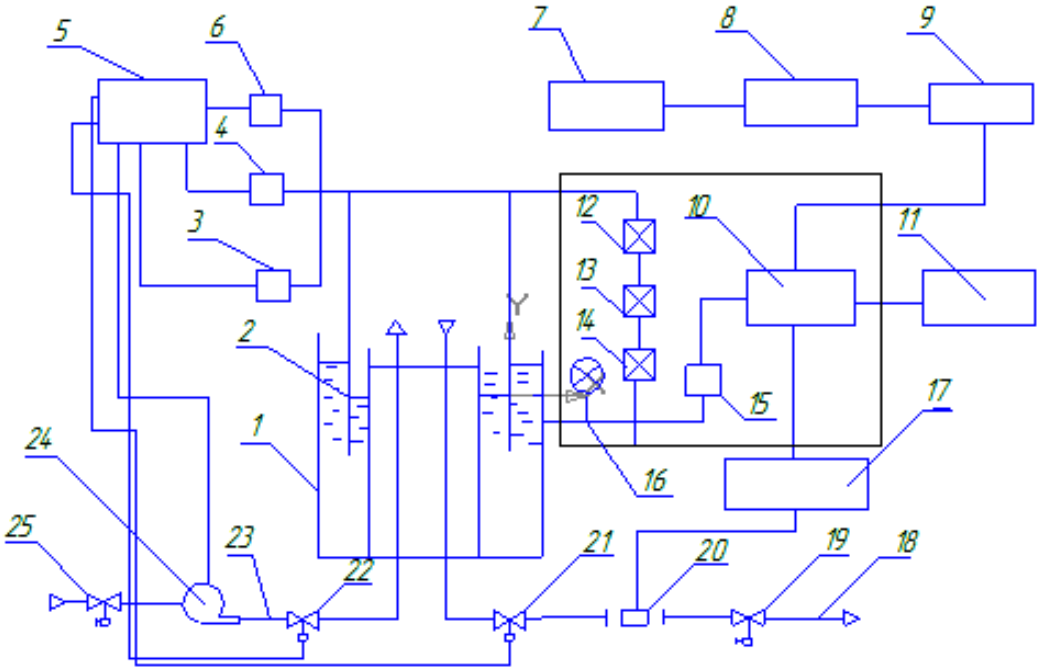


Рисунок 1.1 – Будова дзвонової випробувальної установки (ВУ)

До складу дзвонової ВУ входить:

а) витіснювач;

б) дзвоновий мірник;

в) засоби контролю та пристрої керування роботою;

г) система обробки даних;

д) пристрій індикації лінійного переміщення дзвонового мірника;

е) пристрій, що атестується;

є) регулювальні засувки;

ж) трубопроводи;

з) нагнітальний пристрій.

Дана установка складається з дзвону 2, який нижньою частиною занурений в резервуар 1 (витіснювач). Спочатку випробувань здійснюється подача під дзвін при відкритій арматурі 22 від стороннього джерела витрати 24, тому дзвін піднімається. При відкритті 21 в трубопроводі, що з'єднує під дзвоновий простір з приладом 20, що повіряється, відбувається витіснення

газу через цей прилад. Об’єм газу відраховується за лінійкою 12-16. Після цього 21 закривають. За різницею показів відлікового пристрою та відомим контрольованим об’ємом, визначають похибку.

При опусканні мірника в рідину за рахунок того, що дзвін має об’єм, проходить підняття рівня рідини в просторі , причому сума об’ємів рідини, визначається:

$$(S_1+S_2)\Delta H_1=S\Delta H_1=\frac{\Pi}{4}(D^2_n - D^2_b)\,\Delta H \qquad (1.1)$$

Відомо, що вимірювальна установка – сукупність функціонально об'єднаних засобів вимірювань (вимірювальних приладів, вимірювальних перетворювачів) та допоміжних пристроїв; призначена для вироблення сигналів вимірювальної інформації у формі, зручній для безпосереднього сприйняття спостерігачем, розташована в одному місці.

Особливості:

- робота приладу відбувається по дискретно-динамічному методу;
- висока точність;
- великий діапазон об’ємів (до 1:1000).

1.3 Переваги та недоліки витратовимірювальних установок на основі еталонних лічильників

Відомо, що еталонні установки використовуються для повірки витратомірів та лічильників газу.

Будова таких установок показана на рисунку 1.2.

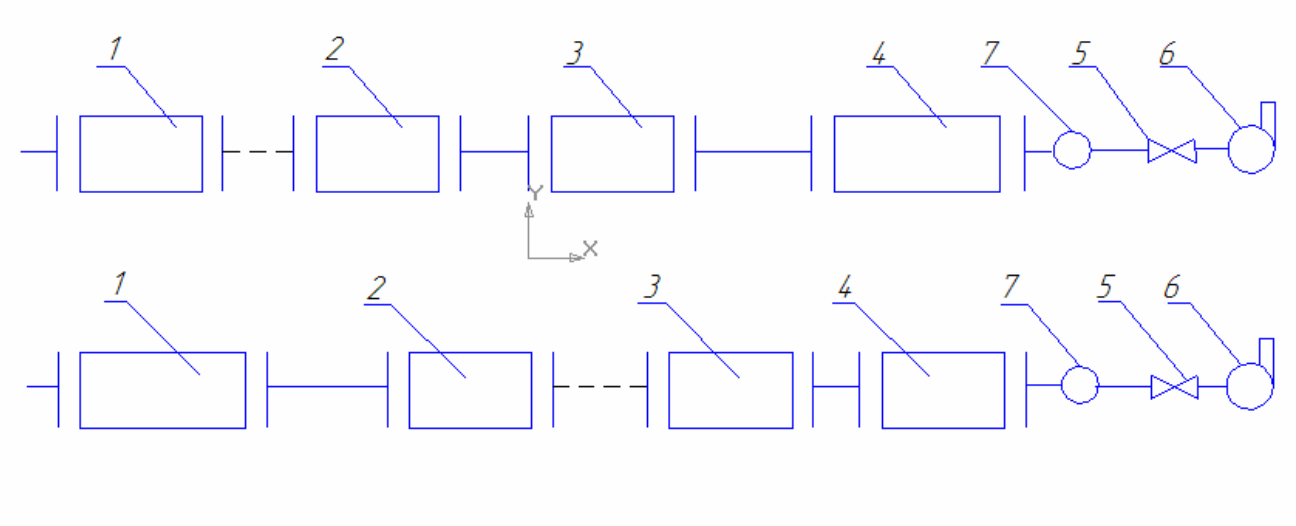


Рисунок 1.2 – Будова еталонних установок з робочими еталонами

До складу таких установок входять:

а) пристрій, що повіряється;

б) зразковий пристрій;

в) регулювальний клапан;

г) джерело витрати;

д) дросель.

Недоліком таких установок є мала точність відтворення об’єму.

Особливості :

- широкий діапазон відтворювальних об’ємів;
- можливість виконання процесу оптимізації циклу;
- точність вимірювань;
- простота використання.

1.4 Опис та структура робочої схеми метрологічної повірки

Типова структурна схема автоматизованої установки показана на рисунку 1.3

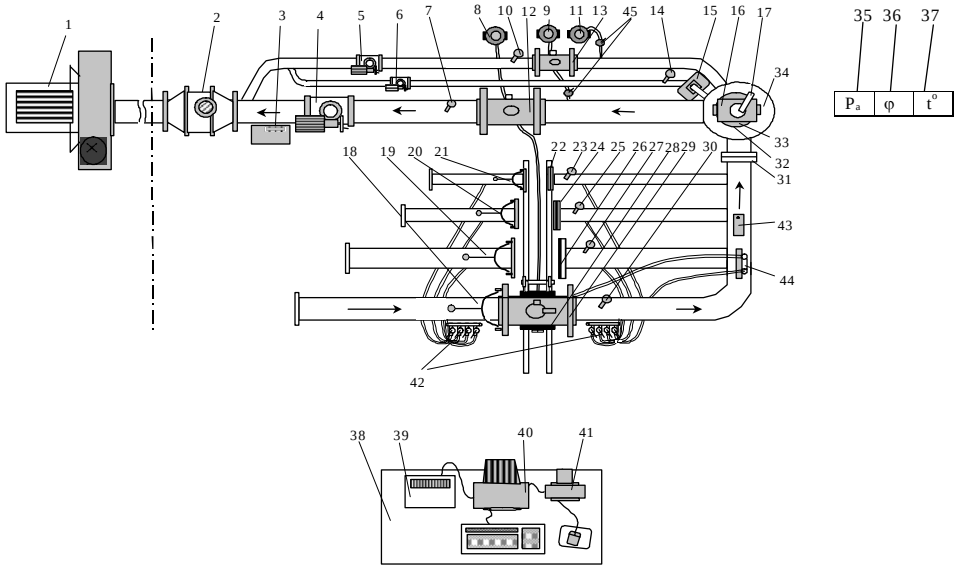


Рисунок 1.3 - Структурна схема автоматизованої установки повірки газових лічильників

де 1 – пристрій створення витрати;

2 – регулятор витрати;

3 – силовий блок управління

4,5,6 – запірна арматура;

7, 10, 14, 23, 25, 27, 30 – давач температури;

8 – давач для вимірювання різниці тиску між лічильником та навколишнім середовищем (атмосферою);

9, 11 – давач для вимірювання різниці тиску між робочими еталонами та навколишнім середовищем (атмосферою);

12, 13, 15 – робочі еталони об'єму газу (ЛГ-200, ЛГ-80, РЛ-16) із монтованими перетворювачами переміщення елементів робочих еталонів в електричні імпульси;

16 – лічильник роторного типу (РГ);

17 – перетворювач обертів робочих елементів роторних та турбінних лічильників в електричні вихідні сигнали;

18, 19, 20, 21 – притискний механізм;

22, 24, 26 – заглушки;

28 – каретка для лічильників турбінного типу;

29 – лічильник турбінного типу (ЛГ, (ЛГ-К));

31 – фільтр 1

32 – фільтр 2;

33 – перехідник;

34 – стійка для встановлення роторних лічильників,

35 – барометр-анероїд;

36 – психрометр аспіраційний;

37 – вимірювач температури навколишнього середовища;

38 – робоче місце оператора;

39 – блок управління і контролю;

40 – ЕОМ із вмонтованими контролерами збору даних та управління запірною та регулюючою арматурою і вентилятором;

41 – друкарка;

42 – комутатор забору перепаду тиску робочих лічильників типу ЛГ(ЛГ-К);

43 – манометр;

44 – мановакууметр U-подібний;

45 – відсікаючі крани.

Установка відноситься до нестандартизованих зразкових засобів вимірювальної техніки за ДСТУ 3215 і працює із застосуванням методу звірянь за ДСТУ3383 показів лічильників газу, що випробовуються (подальше - лічильників), з показами робочих еталонів об′єму газу (подальше - робочих еталонів), що входять в комплект установки і призначена для повірки, калібрування та метрологічної атестації (подальше – повірки) промислових лічильників газу.

Установка відноситься до однофункціональних виробів конкретного призначення виду 1, що відновлюється за ГОСТ 27.003.

1.5 Аналіз систем повірки лічильників газу та інформаційно-метрологічних комплексів

1.5.1 Установка повірки лічильників газу УПСГ 6500. Виробництво "ELSTER GmbH", Німеччина.



Рисунок 1.4 - Установка повірки лічильників газу УПСГ 6500

Установка складається з наступних складових частин:

- вимірювальної частини, що складається з двох наборів еталонних сопел НС1 і Нс2 (по вісім сопел в кожному наборі), чотирьох еталонних турбінних лічильників TRZ G1000;
- чотирьох повітродувок, що забезпечують необхідні значення витрати;
- випробувальних ділянок для монтажу вивіреного лічильника;
- первинних перетворювачів тиску, перепаду тиску, температури, вологості;
- запірної та регулюючої апаратури;
- датчиків знімання сигналів з вивіреного і еталонних лічильників;
- системи електроживлення та управління установкою;
- комп'ютера;
- інтерфейсу, що забезпечує зв'язок комп'ютера з іншими елементами установки.

Програмне забезпечення має захист від несанкціонованого доступу у вигляді паролів.

Принцип роботи

За допомогою повітродувки у випробувальній магістралі створюється розрідження повітря. У результаті цього повітря з приміщення починає надходити у вхідний трубопровід установки і проходить через Вивірений лічильник і, залежно від необхідної витрати, або через еталонні сопла, або через еталонний турбінний лічильник. Вимірюючи тиск, температуру, вологість і час проходження повітря, комп'ютер розраховує значення еталонного обсягу, порівнює його з показаннями приладу, що повіряється і розраховує абсолютну похибку.

1.5.2 Установки повірочні УПГ виробництва ТОВ «Ельстер-Газелектроніка»

Установки повірочні УПГ виробництва ТОВ «Ельстер-Газелектроніка» призначені для повірки лічильників газу ротаційних RVG, турбінних СГ і TRZ, діафрагменних ВК та інших лічильників газу, що мають імпульсний вихідний сигнал, кількість імпульсів якого пропорційно вимірюваному витраті газу. Установки застосовуються на підприємствах-виробниках для проведення первинної повірки лічильників газу при випуску з виробництва або після ремонту, а також в організаціях державної метрологічної служби для проведення періодичної повірки лічильників газу, що знаходяться в експлуатації.

Установка для повірки лічильників газу УПСГ-6500 призначена для повірки лічильників газу. Установка застосовується для проведення первинної повірки лічильників газу при випуску з виробництва або після ремонту, а також при проведенні періодичної повірки лічильників, що знаходяться в експлуатації.



Рисунок 1.5 - Установка повірочна УПГ виробництва ТОВ «Ельстер-Газелектроніка»

Установка складається з наступних основних частин:

- силової рами для монтажу всіх елементів установки;
- вимірювальної частини, що складається з декількох вимірювальних ліній, що включають в себе еталонні лічильники газу;
- сполучних трубопроводів;
- запірної арматури (клапани з пневмоприводами, керовані оператором з пульта управління установкою);
- випробувального ділянки для встановлення вивіреного лічильника і під'єднання його до установки;
- пристрої для завдання витрати повітря (повітродувки, перетворювачі частоти, запірна арматура - дискові затвори з механічними редукторами);
- первинних перетворювачів температури;
- первинних перетворювачів тиску;
- пристроїв знімання сигналів з первинних перетворювачів;
- системи електроживлення установки;
- пристрої знімання сигналів з вивіреного лічильника;

- пульта управління пневматикою;
- керуючого комп'ютера;
- програмного забезпечення;
- друкувального пристрою для отримання протоколів повірки на паперовому носії;
- пульта дистанційного керування роботою установки в ручному режимі повірки.

Установки залежно від діапазонів вимірюваних витрат мають наступні модифікації: УПГ 0,015 / 16; УПГ 0,015 / 100; УПГ 0,015 / 650; УПГ 1/650; УПГ 0,015 / 1000; УПГ 1/1000; УПГ 0,015 / 1600; УПГ 1/1600.

Принцип дії установок заснований на звіренні показань лічильників газу з еталонними лічильниками газу. За допомогою повітродувки у випробувальній магістралі створюється розрідження повітря. У результаті цього повітря з приміщення починає надходити у вхідний трубопровід установки і проходить через лічильник і, залежно від значення витрати, через один з еталонних лічильників.

1.5.3 Універсальна комп'ютеризована установка "Метрогаз"

Універсальна комп'ютеризована установка "Метрогаз" використовується для повірки побутових лічильників газу методом порівняння з робочими еталонами - лічильниками барабанного і роторного типу. Атестована Укрметртестстандартом.

Основні характеристики установки:

1. Повірка лічильників різноманітних типів і типорозмірів G1, 6; G2, 5.

2. Діапазон об’ємної витрати 0,016 - 16 м ^{3} / год.

3. Межі відносної похибки ± 0,5%.

4. Можливість групової повірки - до 10 лічильників одночасно (мембранних) і до 5 лічильників одночасно (роторних).



Рисунок 1.6 - Універсальна комп'ютеризована установка "Метрогаз"

5. Можливість архівації результатів повірки лічильників, друкування протоколів повірки.

6. Комфортне програмне забезпечення для роботи на установці, що дозволяє:

- у процесі роботи безупинно контролювати величину фактичної витрати та об'єму;

- керувати роботою повітродувки що знаходиться поза зоною установки.

1.5.4 Автоматизований вимірювальний комплекс по повірці газових лічильників

Програма повірки лічильників, призначена для перевірки і градуювання промислових лічильників обсягу газу, яка використовується у складі відповідної установки.

Нижче з ілюстративної метою в короткій формі наводиться опис одного з режимів роботи системи - повірки тільки одного типу лічильників об'єму газу - турбінного (хоча програмне забезпечення призначене для двох типів лічильників: турбінних та ротаційних).

На рис. 1.7 показано основне меню програми, що з'являється на екрані після запуску програми.

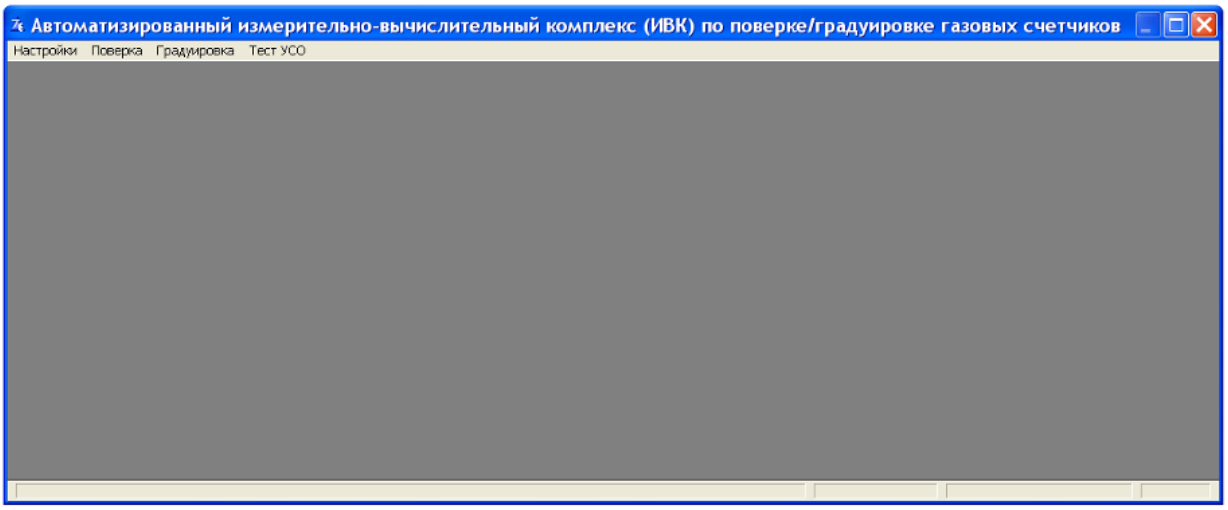


Рисунок 1.7 – Основне меню програми

Перш ніж приступити до роботи за безпосередньої повірку або градуюванні газових лічильників, необхідно задати вихідні параметри датчиків аналогових сигналів, що вимірюють температуру і тиск (номінальні значення вимірювальних резисторів і номінальні (максимальні) значення вимірюваних фізичних величин) і підключаються до аналогових входів установки (рис. 1.8, рис. 1.9), сформувати базу даних аеродинамічних параметрів використовуваних сопел (рис. 1.10, рис. 1.11), вибрати параметри порту зв'язку з комп'ютером (рис. 1.12).

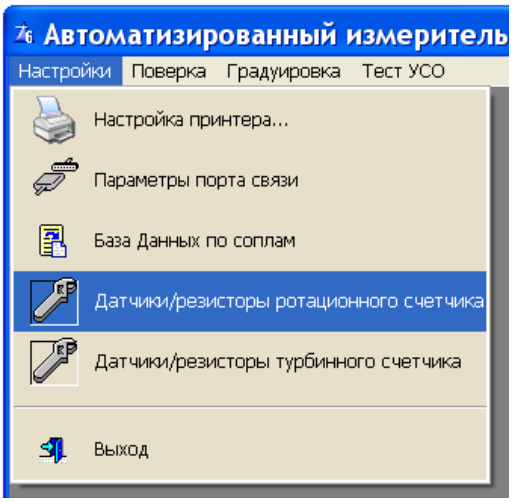


Рисунок 1.8 – Послідовність вибору меню налаштувань коефіцієнтів турбінного лічильника



Рисунок 1.9 – Меню налаштувань коефіцієнтів для турбінного лічильника газу

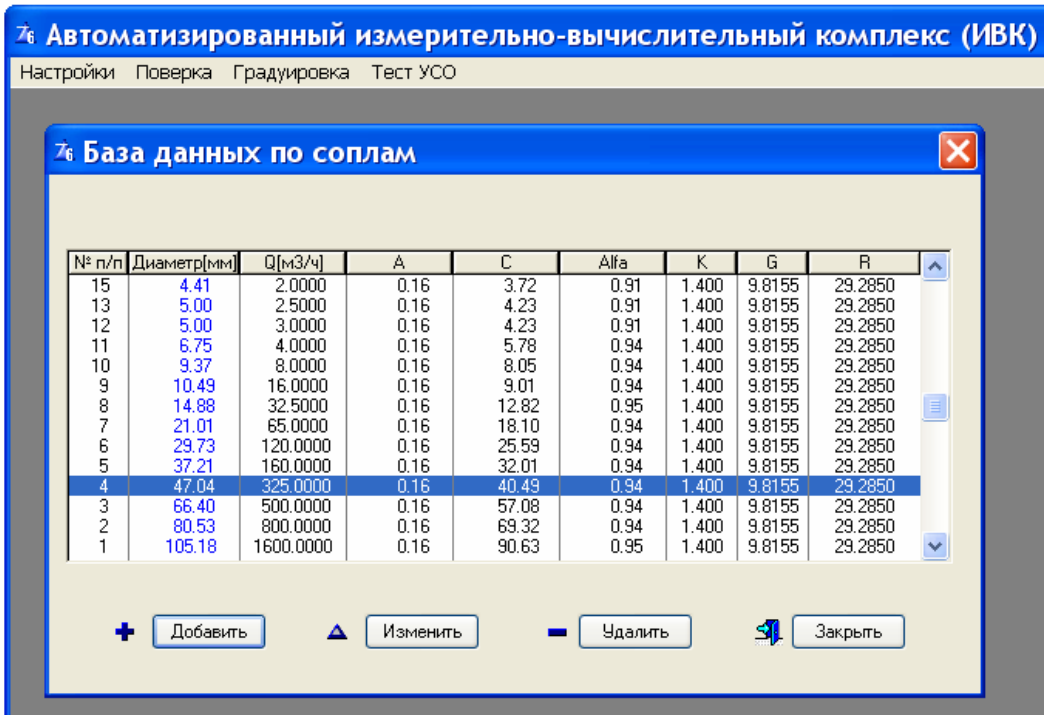


Рисунок 1.10 – База даних по соплах

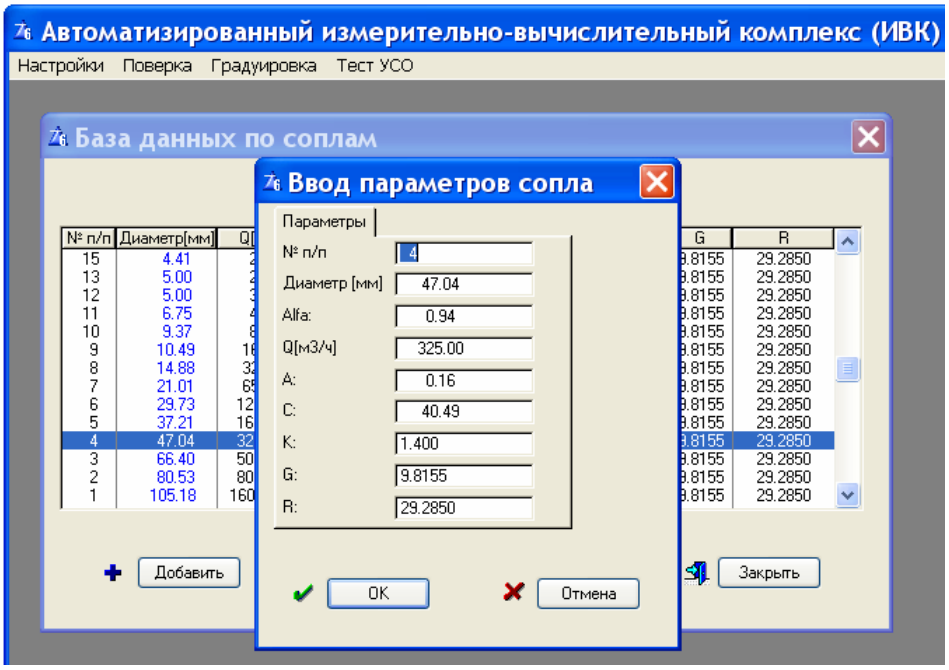


Рисунок 1.11 – Коректування параметрів запису №4 в базі даних

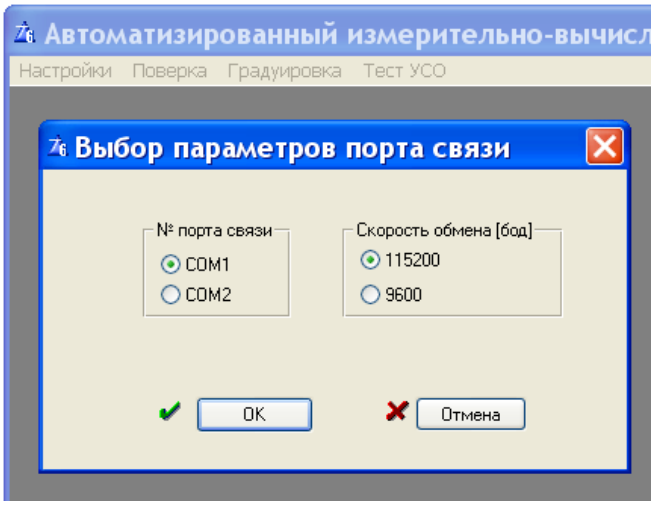


Рисунок 1.12 – Вибір параметрів порта зв’язку з комп’ютером

Після цього необхідно вибрати необхідний режим роботи системи (наприклад, Повірка), тип лічильника (наприклад, Турбінний), тип його електричного виходу (наприклад, С ВЧ-виходом) і спосіб запуску (наприклад, Запуск від комп'ютера) - рис. 1.13.

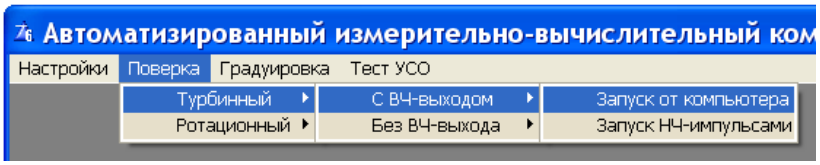


Рисунок 1.13 – Вибір режиму повірки турбінного лічильника

Після закінчення налаштування витрати і натиснення кнопки Вимірювання починаються вимірювання для повірки лічильника (рис. 1.14).

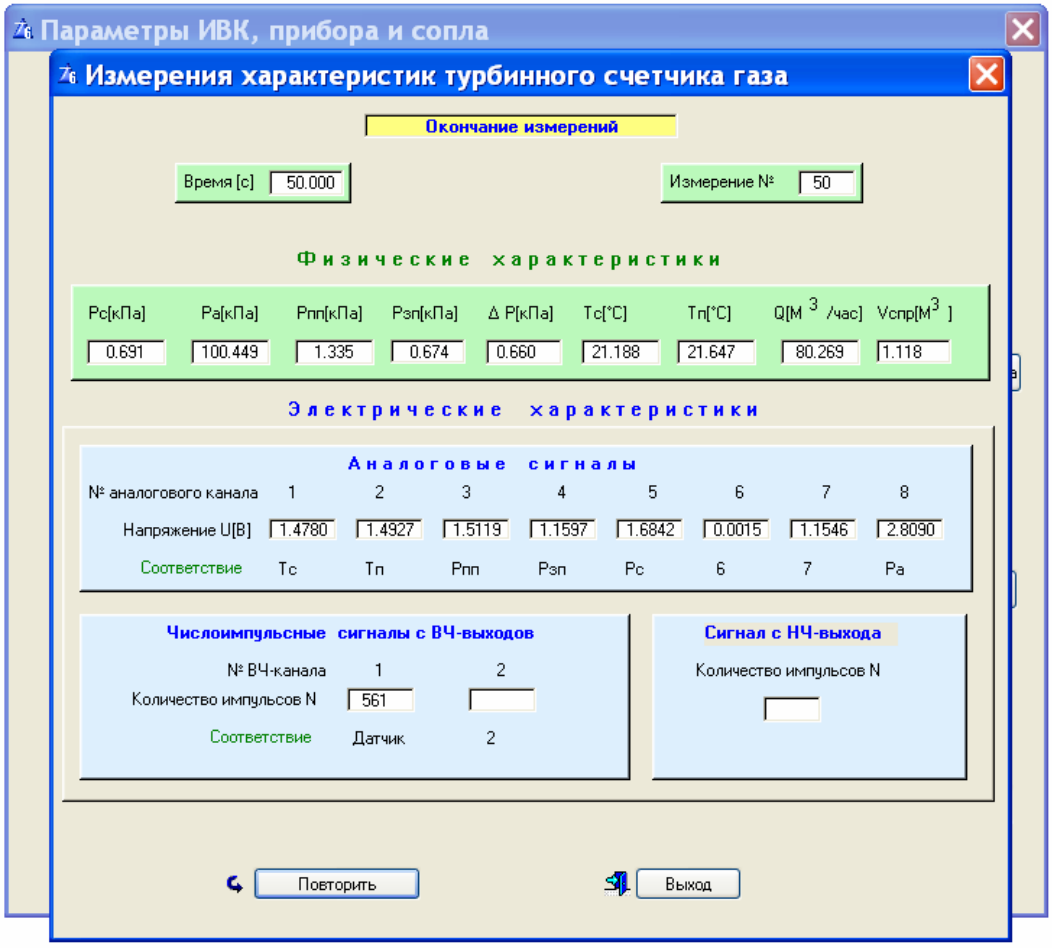


Рисунок 1.14 – Вимірювання характеристик лічильника

Після цього на екран виводиться вікно, що відображає протокол з результатами попередніх вимірювань, на якому показано графік залежності похибки приладу від витрати і область, яка відображає діапазон похибок, при попаданні похибки в який лічильник вважається придатним.

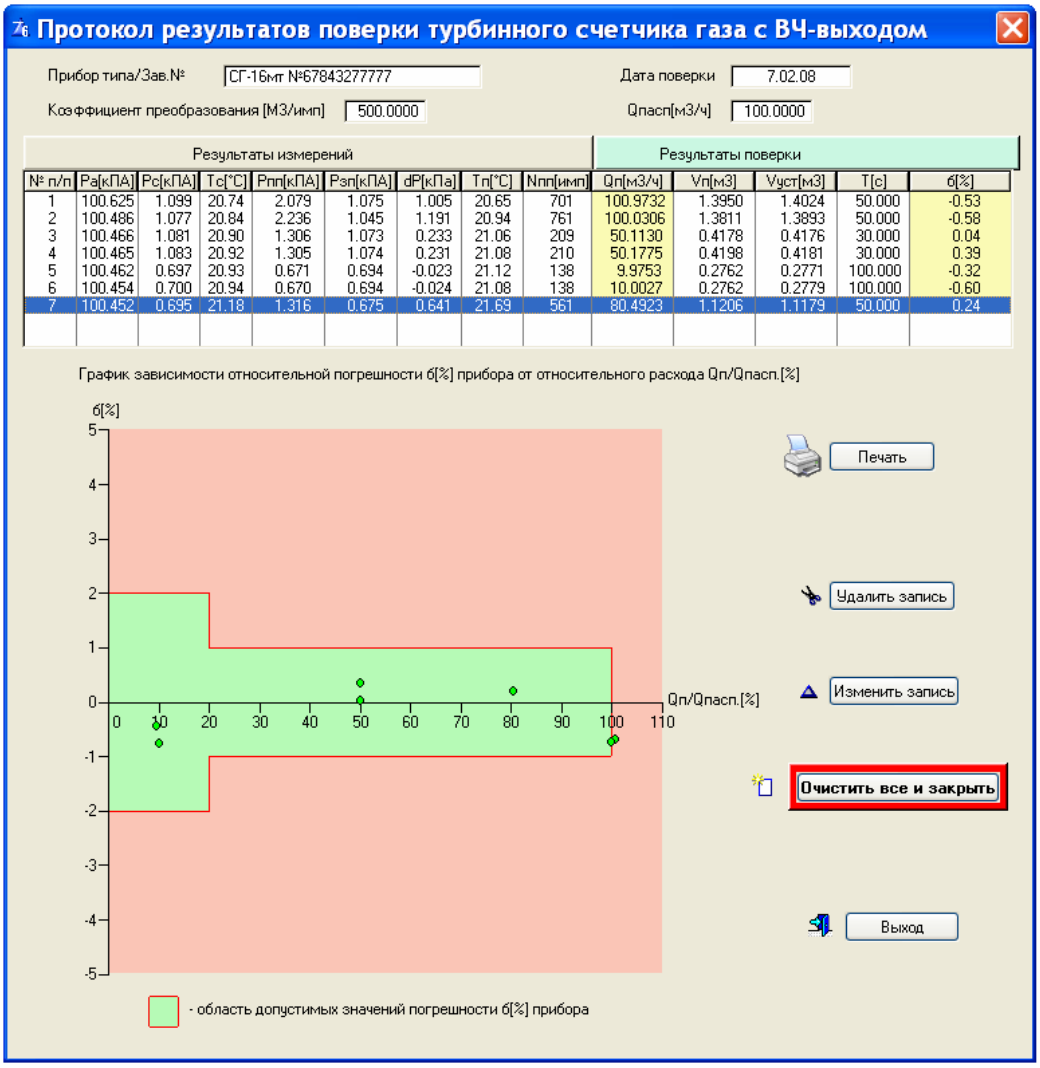


Рисунок 1.15 – Протокол результатів повірки лічильника.

1.5.5. Пакет програм "Інформаційно-вимірювальний комплекс" (ТзОВ "Івано-Франківське спеціальне конструкторське бюро засобів автоматизації")

Пакет програм "Інформаційно-вимірювальний комплекс" дозволяє:

- перегляд технологічного процесу в режимі реального часу на мнемосхемі, слідкування за параметрами у вигляді гістограм та часових діаграм при під'єднанні до пристрою збору та обробки даних;
- імпорт даних з пристрою збору та обробки даних;
- перегляд та вивід результатів роботи на друкуючий пристрій (прінтер) у вигляді таблиць, часових діаграм (за годину, за добу і за заданий час);

- ведення протоколу аварійних ситуацій при виході контрольованих параметрів за встановлені межі, а також друк даного протоколу (за годину, за добу і за заданий час);
- редагування мнемосхем, конфігурування параметрів;
- синхронізація дати і часу на комп'ютері і в пристрої збору та обробки даних.

Передача даних між "Інформаційно-вимірювальним комплексом" і пристроєм збору та обробки даних здійснюється по протоколу MODBUS(ASCII). Збереження даних на комп'ютері можливе протягом 10 років безперервної роботи.

Програми, які входять в "Інформаційно-вимірювальний комплекс":

1. Програма "Інформаційно-вимірювальний комплекс. Конфігурація"

Тут встановлюється послідовний порт, за допомогою якого здійснюється зв'язок з пристроєм збору та обробки даних, та його швидкість передачі даних.

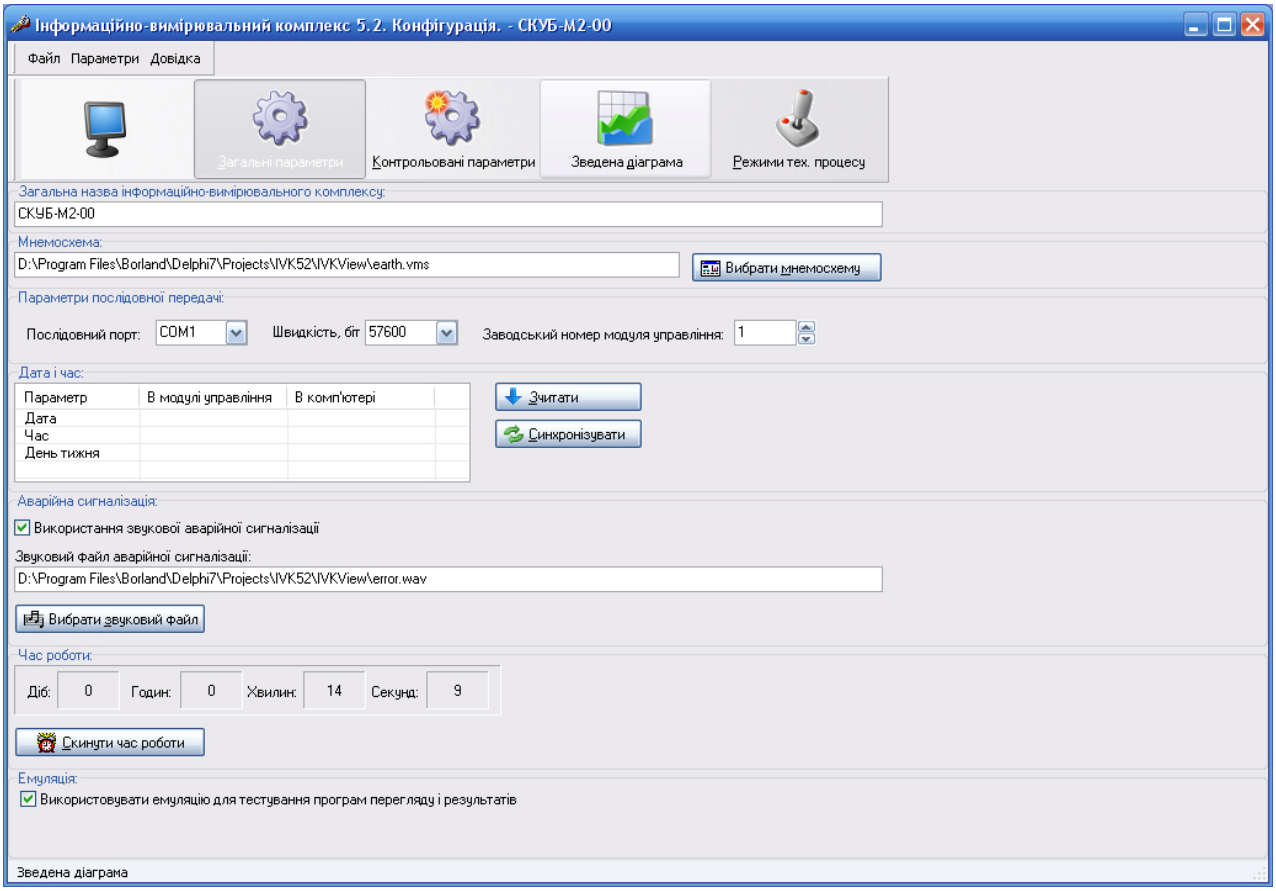


Рисунок 1.16 - "Інформаційно-вимірювальний комплекс". Конфігурація

Далі вибір мнемосхеми, звукової сигналізації, внесення в таблицю контрольованих параметрів, вибір контрольованих параметрів для зведеної діаграми (до 4-х параметрів), введення списку режимів технологічного процесу. Також програма синхронізує дату і час в комп'ютері і в приладі збору і обробки даних.

2. Програма "Інформаційно-вимірювальний комплекс. Перегляд"

Програма допомагає слідкувати за технологічним процесом на мнемосхемі та здійснювати перегляд контрольованих параметрів у вигляді гістограм та часових діаграм.

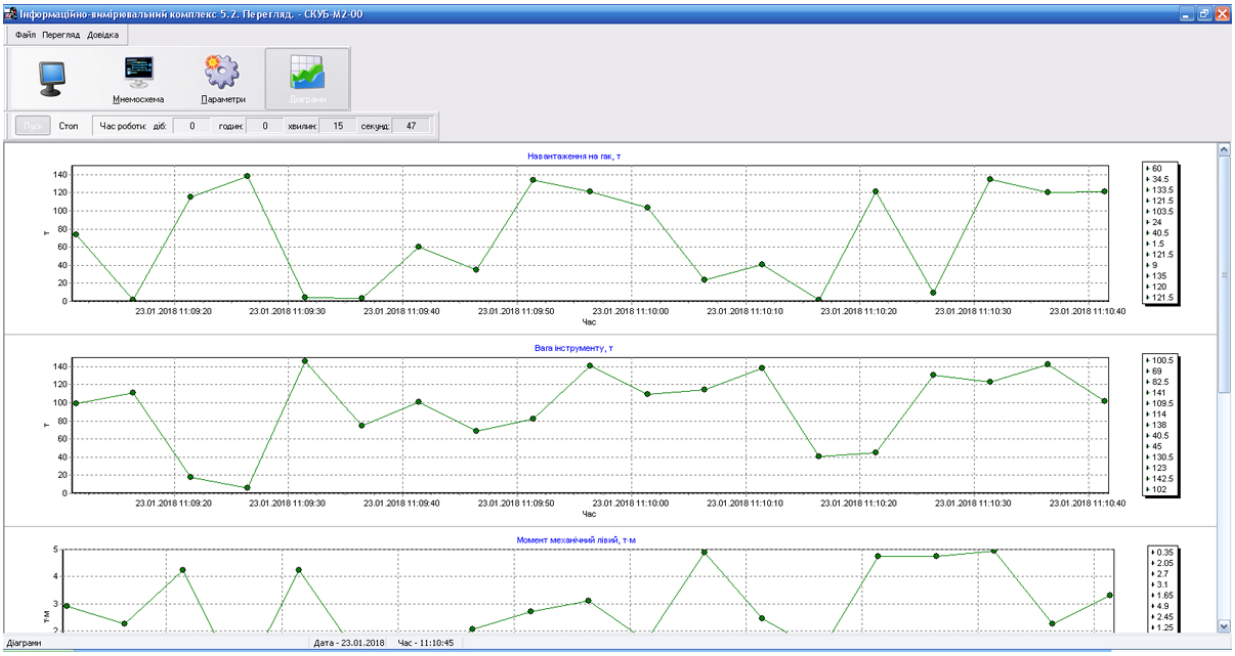


Рисунок 1.17 - "Інформаційно-вимірювальний комплекс". Перегляд

3. Програма "Інформаційно-вимірювальний комплекс. Результати"

Програма дозволяє робити перегляд та вивід результатів роботи на друкуючий пристрій (принтер) у вигляді таблиць, часових діаграм (за годину, за добу і за заданий час), ведення протоколу аварійних ситуацій при виході контрольованих параметрів за встановлені межі, а також друк даного протоколу (за годину, за добу і за заданий час).

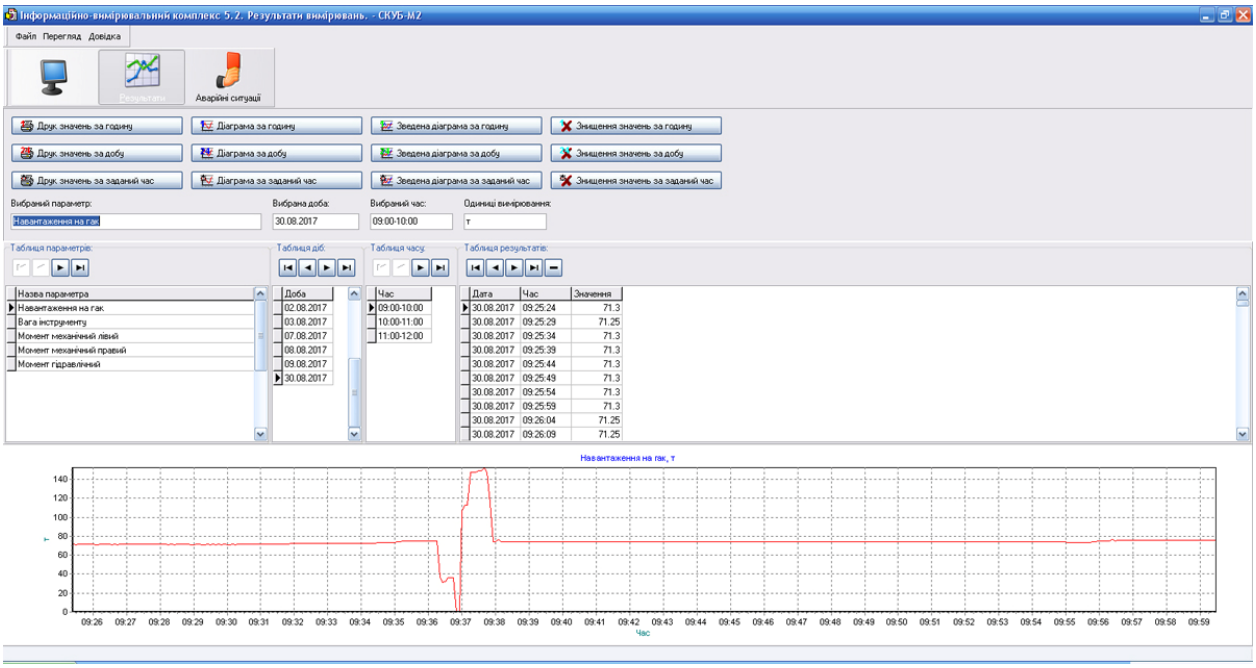


Рисунок 1. 18 - "Інформаційно-вимірювальний комплекс". Результати

4. Програма "Інформаційно-вимірювальний комплекс. Станція збору даних з робочих станцій"

Дозволяє переглянути результати вимірювання і аварійні ситуації на всіх робочих станціях, які під'єднані до станції збору даних з робочих станцій, вивести на друкуючий пристрій (принтер) результати вимірювання, діаграми та аварійні ситуації за заданий час. Є окремою програмою. Надається одна програма на декілька робочих станцій. Все програмне забезпечення надається з ключем захисту від несанкціонованого доступу. Пакет програм працює в операційній системі Windows.

1.6 Класифікація програмного забезпечення обробки метрологічних даних

Область застосування ПЗ в метрології з кожним роком захоплює все ширші сфери. Виникає необхідність, вибору вимог до ПЗ і завдань його метрологічної атестації. Так як вони відрізняються між собою в залежності від розв'язуваної задачі, виникає потреба зробити спробу класифікації наявних

завдань метрологічної атестації програмного забезпечення, використовуваного в метрології.

Класифікувати ПЗ можна за наступними ознаками:

За ступенем прив’язки до конкретних засобів вимірювань і вимірювальних систем для ПЗ можна розрізняти такі категорії:

- прив'язані до конкретних екземплярів (або типам засобів вимірювань);
- відсутність залежності від конкретних засобів вимірювання.
- моделювання вимірювальних експериментів

За ступенем автономності та щодо можливого ПЗ ділиться на:

- жорстко прив'язане до засобів для вирішення поставлених вимірювальних завдань і не допускає його виділення як самостійного об'єкта метрологічної атестації(вбудоване або призначене для користувача ПЗ);
- допускає використовувати його частину як самостійний об'єкт метрологічної атестації(модифіковане комерційне програмне забезпечення зі зміненим кодом (LabView, і ін.));
- автономне комерційне ПЗ, яке куплено готовим і використовується без модифікації (Microsoft Excel, MATLAB та інші пакети прикладних програм).

За ступенем використання засобів обчислювальної техніки ПЗ класифікується як:

- розроблене цільовим чином для систем вимірювальної техніки(що застосовуються в конкретному прикладі);
- використовується для персонального універсального комп'ютера багатоцільового призначення.

За ступенем захищеності або класу ризику фальсифікації даних ПЗ підрозділяється на:

- захищене від випадкового неправильного використання;
- захищене від шахрайства, що включає підкачування програм,

використання не документованих команд, що надходять через інтерфейс користувача, захист контрольованих параметрів, які забезпечуються фізичним пломбуванням, а також електронними та криптографічними засобами.

По можливості виділення метрологічно значущою і законодавчо контрольованої частини використовуваного програмного забезпечення:

- поділ ПЗ з виділенням метрологічно значущою і законодавчо контрольованої частини можливо;
- поділ ПЗ з виділенням метрологічно значущою і законодавчо контрольованої частини неможливо.

Особливості цих категорій наведені в таблиці 3.1.

Таблиця 1.1 - Категорії програмного забезпечення

Категорія	**Типи**	**Групи програм**
1 (ППЗ)	Операційні системи	Операційні системи
2 (ППЗ)	«Жорстке» програмне забезпечення	«Зашиті» або вбудоване програмне забезпечення
3 (ППЗ)	Стандартні пакети програм	Програми електронної пошти, текстові процесори.
4 (МПЗ)	Конфігуровані і модифіковані пакети програмного забезпечення	Програми як засіб програмування, необхідне пристосування перед використанням.
5 (СПЗ)	Призначені для користувача, замовні або «саморобні» програми	Програмне забезпечення, розроблене для користувача і використовує програмні інструменти.

По можливості завантаження модифікованих версій ПЗ:

- оновлення з перевіркою;
- простежується оновлення.

Типи ПЗ можна розбити на 5 різних категорій:

- COTS (Commercial off-the-shelf), тобто готове (покупне) ПЗ - ППЗ;
- MOTS (Modified off-the-shelf), модифіковане покупне ПЗ- МПЗ;
- CUSTOM, виготовлене на замовлення або «саморобне » ПЗ- СПЗ.

Основними вимогами Директиви до ПЗ є:

1) Те, що технічна документація повинна включати необхідні для оцінки та ідентифікації типу засоби вимірювань. Якщо прийнятно, опис електронних приладів з кресленнями, діаграмами, діаграмами руху логістики та загальною інформацією про програмне забезпечення, що пояснюють їх характеристики і роботу;

2) Якщо засіб вимірювань має додаткове програмне забезпечення, яке забезпечує і інші функції, крім вимірювальних, то програмний продукт, який є в цьому випадку критичним для метрологічних характеристик, повинен бути розпізнаваним і не повинен піддаватися неприпустимому впливу додаткового програмного забезпечення.

3) Програмне забезпечення, що впливає на метрологічні характеристики, повинна бути ідентифікована як таке і захищене. Його ідентифікація повинна бути легко забезпечена засобом вимірювання. Докази втручання повинні бути доступні протягом обґрунтованого періоду часу.

4) Дані вимірювань, програмне забезпечення, що впливає на метрологічні характеристики, а також на важливі метрологічні параметри, застосовуються чи передані, повинні відповідним чином бути захищені від випадкового або навмисного втручання.

Тому для виконання цієї Директиви в світі було розробили наступні документи:

- Європейська кооперація з законодавчої метрології розробила Настанову з програмного забезпечення

- Міжнародна організація законодавчої метрології паралельно підготувала Документ МОЗМ «Основні вимоги до програмно керованим засобам вимірювань».

- Регіональна Організація співпраці державних метрологічних установ країн Центральної та Східної Європи випустила рекомендацію КООМЕТ «Програмне забезпечення засобів вимірювальної техніки. Загальні технічні вимоги ».

На рисунку 1.19 відображено, які набори вимог існують.

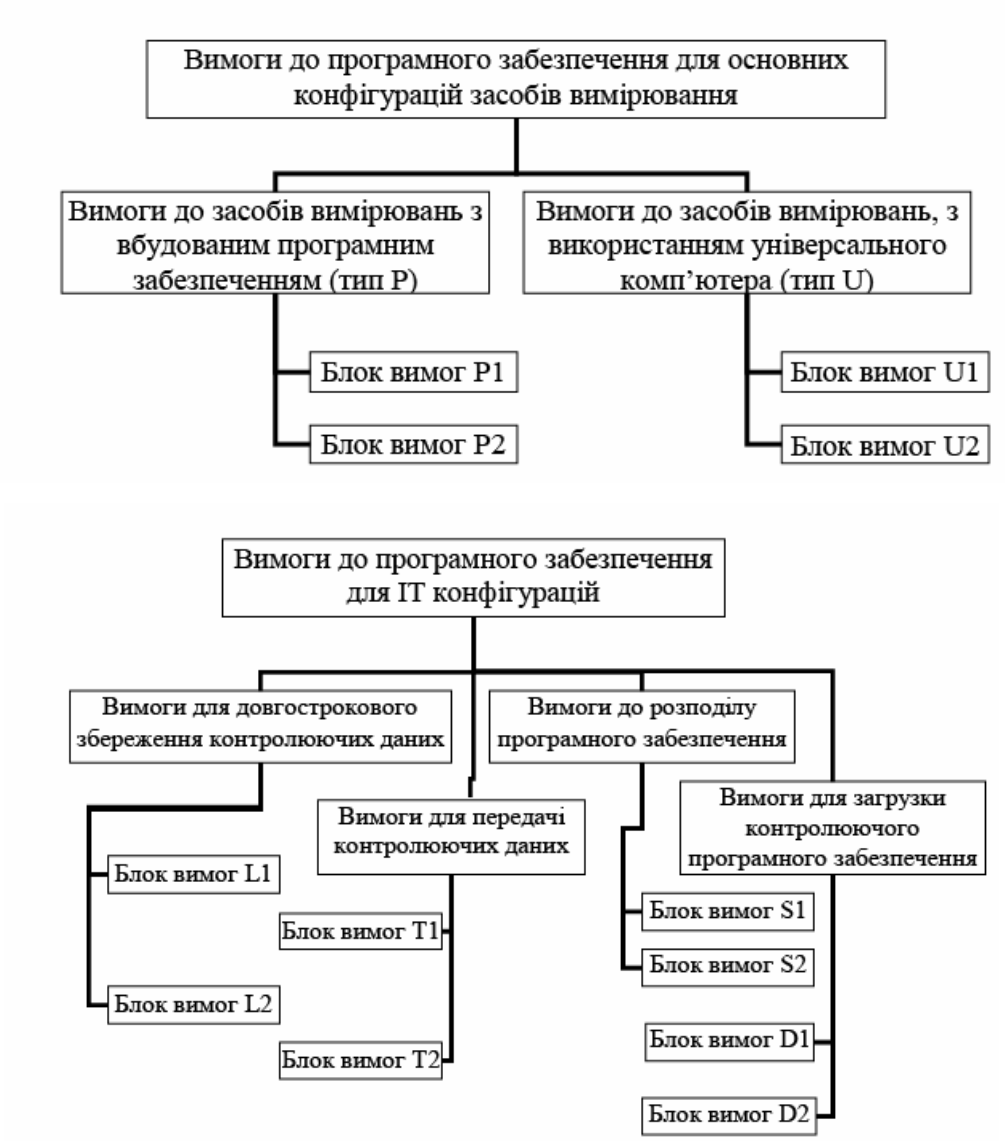


Рисунок 1.19 - Набори вимог

Для кожного ЗВ повинна бути проведена оцінка класу ризику, тому що конкретні вимоги до ПЗ визначаються, перш за все, класом ризику, притаманним приладу.

2 РОЗРОБКА АЛГОРИТМІЧНОГО ЗАБЕЗПЕЧЕННЯ СИСТЕМИ

2.1 Формування вимог до алгоритму

Програма повірки промислових лічильників газу призначена для управління, контролю установки, а також для обробки і зберігання даних, отриманих в процесі експлуатації Установки. Програма в комплексі з ЕОМ із спеціалізованими контролерами, блоком управління і контролю, силовим блоком управління, набором давачів, системи з'єднувальних кабелів, що є невід'ємною частиною установки утворюють комплекс прогамно-апаратних засобів, необхідних для функціонування установки.

Програмне забезпечення повірки промислових лічильників газу повинне виконувати наступні функції:

- збір, реєстрацію, обробку і відображення даних з давачів температури;
- збір, реєстрацію, обробку і відображення даних з давачів тиску;
- реєстрацію та підрахунок імпульсів з робочих еталонів;
- реєстрацію та підрахунок імпульсів з лічильників, що повіряються;
- перетворення підрахованих імпульсів в об'єми;
- комутацію необхідних для поточної роботи давачів;
- індикації положення запірної арматури та регулятора витрати;
- обчислення похибки;
- формування, збереження та друк протоколів

2.2 Розробка алгоритму функціонування програми

Для вирішення поставленого завдання було розроблено структурну схему програми повірки промислових лічильників газу, що зображена на рисунку 2.1.

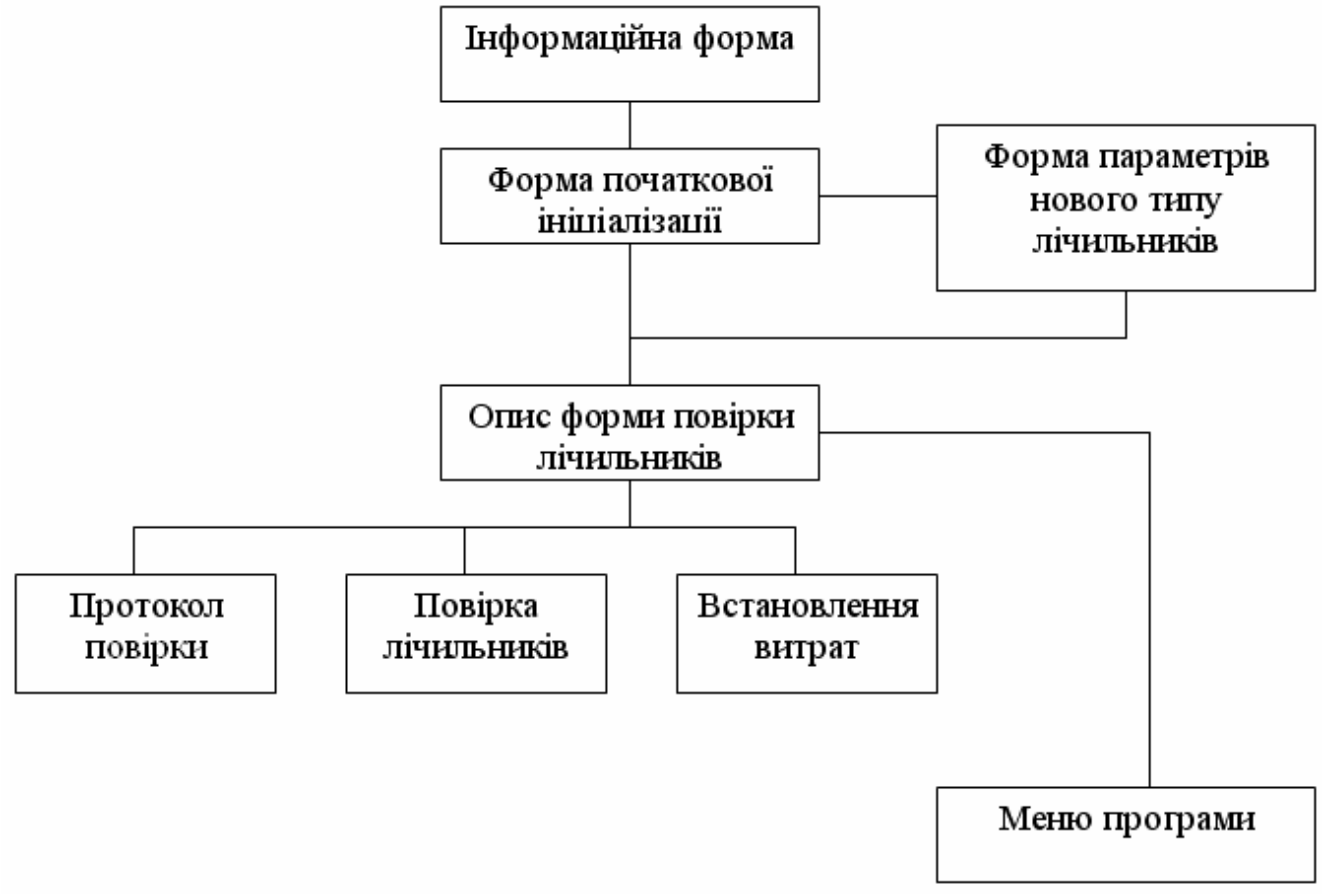


Рисунок 2.1 - Структурна схема повірки промислових лічильників газу

Після запуску програми на екрані з'являється інформаційна форма в якій відображено інформацію про організацію і розробника програми. Дана форма містить кнопки «ОК» і «Вийти». Натискання кнопки «Вийти» завершує роботу Програми, а кнопки "ОК" – викликає форму початкової ініціалізації програми, структурна схема якої зображена на рисунку 2.2.



Рисунок 2.2 - Структурна схема початкової форми ініціалізації

Дана форма дозволяє вибрати за допомогою контекстного меню (з бази лічильників) тип робочого лічильника, що повіряється, відзначити наявність коректора для можливості його подальшої перевірки, або проведення повірки лічильника за допомогою коректора, ввести для відображення в протоколі початкові покази лічильників, атмосферний тиск, відносну вологість, температуру навколишнього середовища та заводський номер лічильника, що повіряється. Крім того, в даній формі передбачено можливість виведення похибки після початку вимірювання при включенні даної опції. Якщо в стандартному списку немає потрібного лічильника в контекстному меню необхідно вибрати пункт « Інший… », тоді на екрані з’явиться форма структурна схема якої наведена в рисунку 2.3

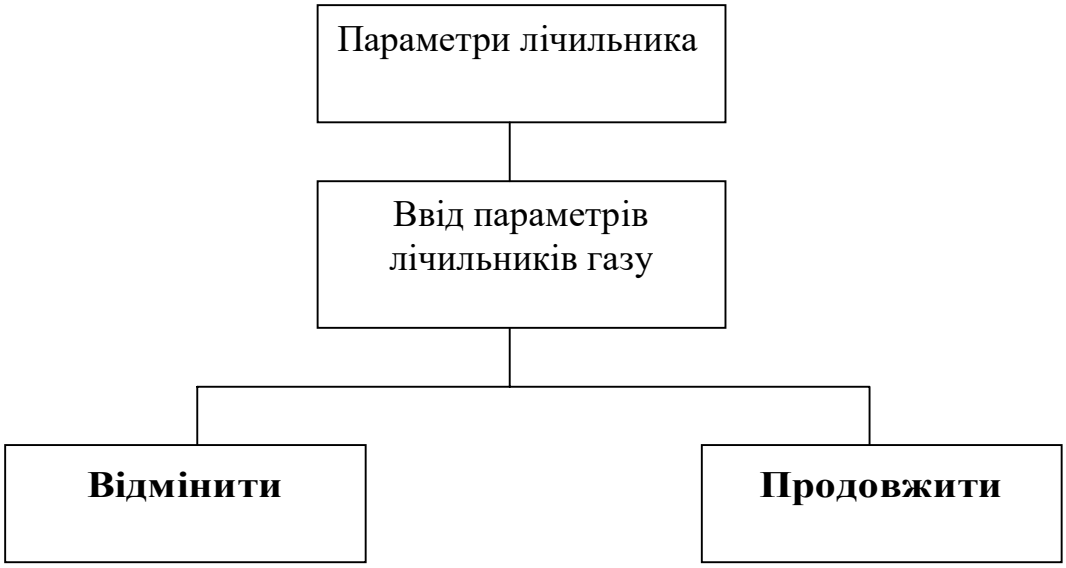


Рисунок 2.3 - Структурна схема форми параметрів лічильника

На цій формі необхідно ввести назву лічильника, вибрати по якій лінії буде проводитися повірка, ввести мінімальну і максимальну витрату лічильника, його коефіцієнт перетворення (кількість імпульсів на 1м^3), а також максимально-допустимі значення похибки в діапазоні від Qmin до 0.2Qmax, та від 0.2Qmax до Qmax і натиснути кнопку «Продовжити» для збереження введених параметрів лічильника. Натиснення кнопки «відмінити» повертає на попередню форму без збереження введених параметрів лічильника.

Для зручності роботи з програмою повірки промислових лічильників газу основні функції програми організовані на трьох формах-вкладках головної форми програми. На першій вкладці розміщені функції повірки лічильників, на другій – функції встановлення витрати та управління (комутація) запірної арматури, на третій – функції перегляду, збереження та друку загального протоколу повірки.

Структурна схема вікна повірки лічильників зображена на рисунку 2.4.

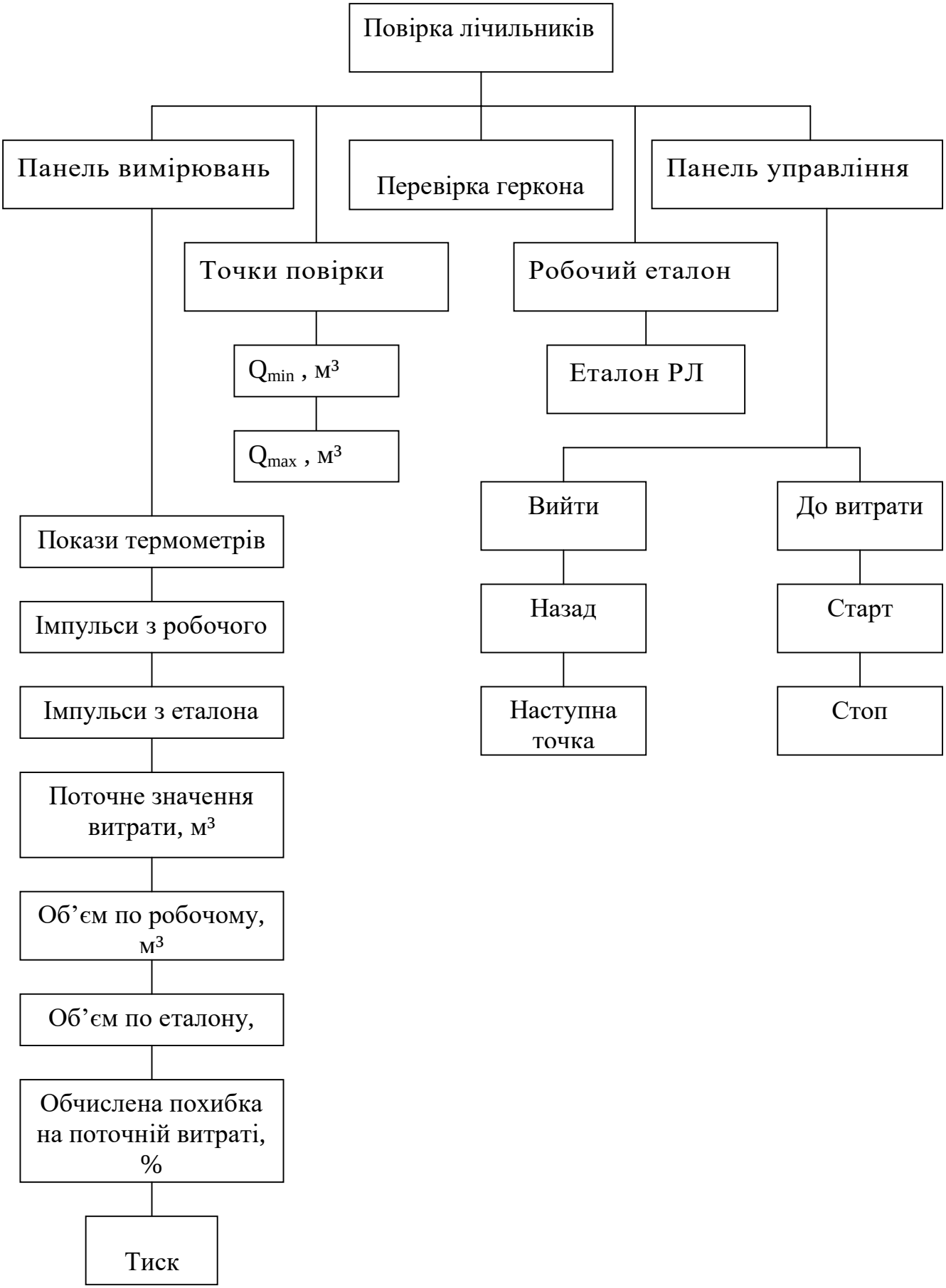


Рисунок 2.4 - Структурна схема вікна повірки

У вікні повірки лічильників розміщені наступні панелі:

- «*Панель вимірювань*», яка відображає інформацію з давачів системи.
- панель «*Робочий лічильник*», на якій показано тип і заводський номер лічильника, що повіряється;
- панель «*Точки повірки*» відображає точки на яких проводиться повірка вибраного лічильника. Вказівник показує, на якій точці проводиться повірка в даний момент. Витрати, на яких відбулася повірка відмічаються відповідними значками;
- панель «*Робочий еталон*» показує який еталон використовується при повірці на поточній витраті;
- «*Панель управління*» в якій розміщені кнопки «Вийти», «Назад», «Наступна точка», «До витрати», «Старт» і «Стоп».

Кнопка «*Вийти*» – вихід з програми.

Кнопка «*Назад*» – повернення до форми «*Початкова ініціалізація*» для зміни типу лічильника.

Кнопка «*Наступна точка*» – переводить програму до наступної точки повірки. В автоматичному режимі відбувається переключення (при необхідності) робочого еталона (що демонструється на панелі) і обчислення його коефіцієнта перетворення на цій точці.

Кнопка «*До витрати*» переводить користувача до сторінки «*Встановлення витрати*».

Кнопка «*Старт*» – починає процес повірки на вибраній точці. При цьому відбувається обнулення лічильників імпульсів (робочого і еталонного), а також обнулення результатів повірки в протоколі (якщо це повторна повірка на цій точці).

Кнопка «*Стоп*» – примусова синхронна (по фронту зростання імпульса з робочого лічильника) зупинка процесу повірки на вибраній точці.

В полі «*Задана кількість імпульсів на робочому лічильнику*» вказується кількість імпульсів з робочого лічильника, яку необхідно отримати для проведення повірки на вибраній точці. Після того, як з лічильника надійде

останній імпульс, відбувається синхронна зупинка лічильників імпульсів і обчислення похибки. Вікно встановлення витрати зображене на рисунку 2.5.

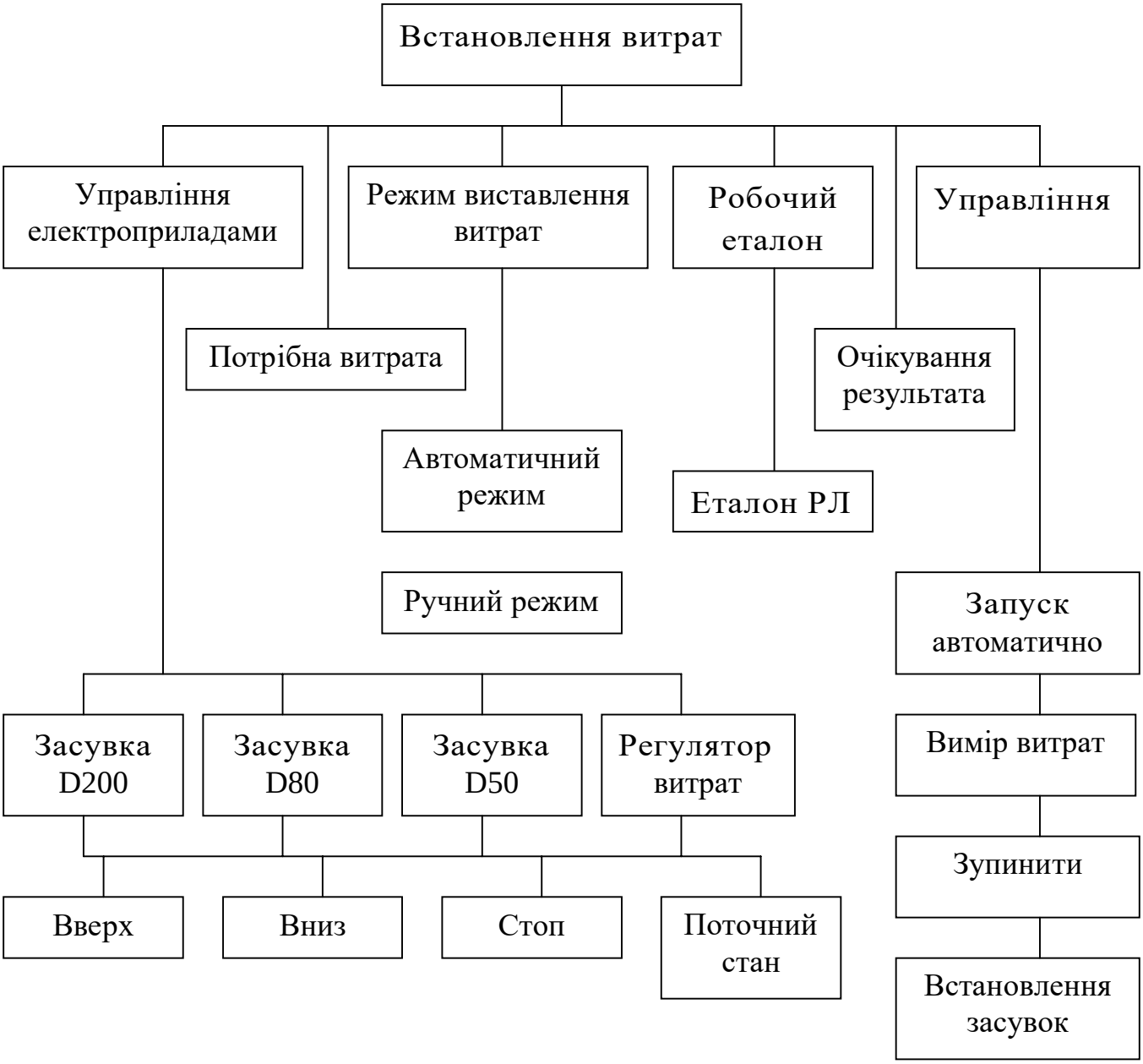


Рисунок 2.5 - Структурна схема вікна встановлення витрати

На сторінці «Встановлення витрати», яка зображена на рисунку 2.5, розміщені:

- панель управління електроприводами в ручному режимі, на якій для кожної засувки і регулятора знаходяться кнопки їх відкривання «Вверх»,

закривання «Вниз» та зупинки «Стоп». Панель активна тільки при ручному режимі виставлення витрати (див. наст. пункт);

- панель «Режими встановлення витрати» дозволяє обрати ручний чи автоматичний режими управління електроприводами;
- панель «Робочий еталон» показує який еталон працює на поточній витраті;
- панель «Управління» містить кнопки «Запуск авт.» і «Зупинка» (активні в автоматичному режимі) відповідно для запуску і зупинки процесу встановлення витрати в автоматичному режимі, а також кнопки «Вимір. Витрати» і «Встан. Засувок», які активні в ручному режимі і служать відповідно для вимірювання витрати, а також для комутації засувок у відповідне до типу робочого еталона положення;
- панель «Потрібна витрата» містить поле, в якому можна задати нестандартне значення витрати. Обчислюється також допустима похибка встановлення витрати: якщо витрата $Q=Q_{min}$ вибраного типу лічильника, то витрата повинна бути встановлена з похибкою (0 ÷ +5)%;
- повідомлення «Значення частоти, Гц», показує частоту імпульсів з активного робочого еталона, яка відповідає значенню витрати, що відображається повідомленням «Значення витрати, $м^3$/год.»;
- полоса «Обчислення витрати» показує хід обчислення витрати;
- повідомлення «Чекайте, встановлюється витрата…», «Працює регулятор» або «Стабілізується еталон» з'являється після початку встановлення витрати в автоматичному режимі;
- повідомлення «Номер ітерації» показує число відкривань/закривань регулятора витрати при встановленні поточної витрати.

У вікні «Протокол повірки» розміщені функції перегляду, збереження та друку загального протоколу повірки, який формується автоматично відповідно до даних отриманих в процесі повірки лічильника. Меню програми складається з двох частин, «Файл» та «Додаткові можливості», зображені на рисунку 2.6.

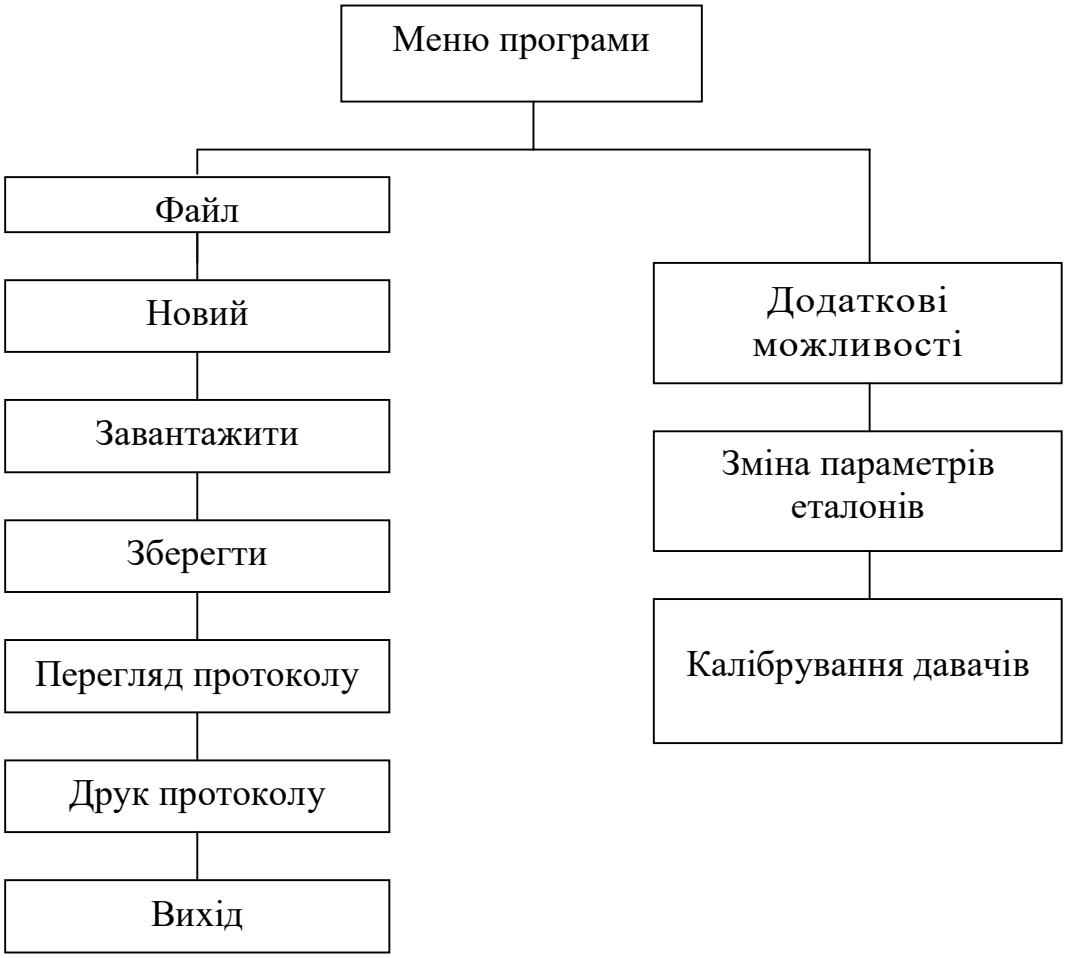


Рисунок 2.6 - Структурна схема вікна меню програми

Меню програми «Файл» містить наступні пункти:

- «Новий» – обнуляє всі дані протоколу повірки;

- «Завантажити» і «Зберегти» – служать для роботи з файлами повірки;

- «Перегляд Протоколу» і «Друк Протоколу» – відповідно попередній перегляд і друк протоколу повірки лічильника;

- «Вихід» – завершує роботу з програмою.

Меню програми «Додаткові можливості» складається з:

- «Зміна параметрів еталонів» служить для коректування коефіцієнтів перетворення робочих еталонів.

- «Калібрування давачів» – служить для коректування таблиці градуювання давачів температури і тиску, що необхідно після кожної їх переповірки.

3 РОЗРОБКА ПРОГРАМНОГО СЕРВІСУ ОБРОБКИ МЕТРОЛОГІЧНИХ ДАНИХ

3.1 Вибір середовища розробки програмного сервісу

Для реалізації програмного сервісу повірки промислових лічильників газу, з метою формування зручного інтерфейсу для користувача, було використано середовище програмування C++ Builder.

Borland C++ Builder, сьогодні є найбільш довершеним візуальним середовищем швидкої розробки на C++ для Windows. У її склад входить близько 200 самих різних компонентів, а створення закінченої програми вимагає мінімуму зусиль. Borland C++ Builder може працювати в середовищі операційних систем Windows. C++Builder був задуманий як максимально універсальна мова, і тому він не може використовувати нестандартний формат об'єктних файлів. Це зробило б неможливим, наприклад, створення файлів підпрограм, які могли б включатися в програми, написані на інших мовах. Можливість роздільної компіляції забезпечується в C++Builder застосуванням заголовних файлів, що приєднуються до компільованих початкових файлів на етапі препроцесорної обробки. Заголовки містять прототипи функцій, оголошення типів і іншу інформацію, необхідну для роздільної компіляції початкових модулів програми.

Хороше розуміння мови C представляється необхідним для успішного програмування на C++. C++Builder повною мірою підтримує стандарт ANSI C і, крім того, деякі інші версії мови (Керніган & Річи, Unix V). Ми в основному орієнтуватимемося на стандарт ANSI, який, до речі, є в якійсь мірі результатом “зворотної дії” мови C++Builder .

C++ Builder є SDI-застосуванням, головне вікно якого містить інструментальну панель (зліва), що настроюється, і палітру компонентів

(справа). Крім цього, за умовчанням при запуску C++ Builder з'являються вікно інспектора об'єктів (зліва) і форма нового застосування (справа). Під вікном форми застосування знаходиться вікно редактора коду.

Для ефективної розробки призначених для користувача інтерфейсів в C++ Builder нерідко виникає необхідність в маніпулюванні компонентами на формах. Більшість операцій для маніпулювання компонентами знаходяться в меню Edit: До різних опцій цього меню слід звертатися після того, як на формі розміщено один або декілька компонентів, властивості яких потрібно змінити. Якщо створене за допомогою C++ Builder застосування в процесі роботи звертається до баз даних, воно, як правило, використовує для цієї мети бібліотеку BDE (Borland Database Engine), засновану на технології IDAPI (Integrated Database Application Program Interface). Ця бібліотека встановлюється автоматично при установці C++ Builder. Для генерації звітів в C++ Builder використовується сторінка QReport на палітрі компонентів. Цей набір компонентів дозволяє проектувати звіти на основі таблиць, запитів, списків, текстових файлів, масивів, використовуючи дизайнер форм C++ Builder. Однією з найважливіших проблем при розробці замовлених програмних продуктів є остаточна збірка проекту і постачання програмних додатків. Ця проблема може бути вирішена різними способами залежно від умов постачання, числа установок, складу продукту, що поставляється. C++Builder забезпечує високу швидкодію при компіляції і збірці 32-розрядних програмних додатків для сучасних операційних систем Windows. Система навіть відображає час, витрачений на основні етапи побудови програм. Результуючі програми добре оптимізовані за швидкістю виконань і витрат пам'яті. Дизайнер форм, інспектор об'єктів і інші засоби залишаються доступними під час роботи програми, тому вносити зміни можна в процесі налаштування.

C++Builder підтримує основні принципи об'єктно-орієнтованого програмування - інкапсуляцію, поліморфізм і множинне спадкоємство, а також нововведені специфікації і ключові слова в стандарті мови C++. Хоча C++

Builder представляється вельми надійною системою, корпорації ще належить спростувати твердження, що в кожній відладженій програмі (у тому числі і в комерційній) є щонайменше одна помилка. Мабуть, саме цим прагненням пояснюється зайва, на мій погляд, поспішність з рекламуванням "покращуваної і розширеної" версії Borland C++ версії.

На закінчення відзначимо основні переваги Borland C++ Builder в порівнянні з іншими засобами розробки програмних додатків:

- швидкість розробки додатку;
- висока продуктивність розробленого додатку за рахунок отримання 32-розрядного виконуваного коду за допомогою оптимізованого компілятора;
- невисокі вимоги розробленого додатку до ресурсів комп'ютера;
- можливість повного доступу до функцій операційних систем Windows;
- можливість модернізації середовища розробки за рахунок вбудовування в неї нових компонент, інструментів, експертів;
- вдале опрацьовування ієрархії об'єктів;
- використання для написання коду одного з найпопулярніших мов програмування C++;
- можливість використання в додатках візуальних компонент, створених за допомогою Borland Delphi;
- можливість розробки нових компонент і інструментів як власними засобами, так і засобами Borland Delphi;
- наявність як на ринку, так і у вільному розповсюдженні (freeware і shareware) великої кількості візуальних компонент інших фірм,
- створених для використання в Borland C++ Builder.

Отже, Borland C++ Builder - це інструмент, що відповідає всім сучасним вимогам до засобів розробки програмних додатків, завдяки використання системи швидкої розробки (RAD-систем, Rapid Application Development — середовище швидкої розробки додатків) в основі якої лежить технологія

візуального проектування і діалогового програмування, суть якої полягає в тому, що середовище розробки бере на себе велику частину роботи по генерації коду програми, залишаючи програмістові роботу по конструюванню діалогових вікон і написанню функцій обробки подій.

3.2 Реалізація блоків алгоритму

На основі розробленої структури в середовищі Builder C++ було розроблено програму, яка реалізує поставлене завдання. Лістинг програми наведено в додатку А.

Для зручності роботи з програмою повірки промислових лічильників газу основні функції програми організовані на трьох формах-вкладках та в меню програми. На першій вкладці розміщені функції повірки лічильників, на другій – функції встановлення витрати та управління (комутація) запірної арматури, на третій – функції перегляду, збереження та друку загального протоколу повірки. В меню розміщені функції для роботи з протоколом (Новий, Завантажити, Зберегти, Перегляд протоколу, Друк протоколу), функція завершення програми «Вихід» та функції додаткових можливостей, що дозволяють змінювати параметри еталонів та виконувати калібрування давачів тиску та температури.

Розробка основного вікна повірки.

В даному розділі програми повинні виконуватися наступні пункти:

- відображення інформації з давачів системи (температури, тиску та імпульсів з робочого і еталонного лічильників);
- обчислення об’єму по робочому лічильнику;
- обчислення об’єму по еталонному лічильнику;
- обчислення і відображення поточного значення витрати;
- обчислення та відображення похибки на поточній витраті;
- відображення точок на яких проводиться повірка;
- індикація активного еталонного лічильника;

- задання кількості імпульсів на робочому лічильнику;
- відображення статусу проходження повітки;
- обчислення часу повірки на даній тотці;
- забезпечення переходу до вікна виставлення витрати;
- запуск/зупинку повірки;
- перехід до вибору лічильника;
- вихід з програми.

Для реалізації поставленого завдання на основну форму програми необхідно помістити:

- Button – стандартна кнопка Windows, яка при натисненні виконує певну функцію;
- RadioButton – радіо кнопка призначена для вибору тільки одного з декількох варіантів;
- панель GroupBox – це контейнер з рамкою і надписом, що об'єднує групу зв'язаних елементів;
- CheckBox – індикатор з прапорцем, що відмічає певну подію;
- Label – елемент призначений для відображення текстової інформації, який не міняється користувачем. В даному випадку використовувався для надписів;
- Edit – вікно редагування для введення одного рядка символів;
- BitBtn – кнопка з можливістю відображення піктограми;
- CGauge – елемент відображення ходу процесу;
- StaticText – елемент призначений для швидкозмінної текстової інформації;
- Image – елемент відображення графіки.

Запуск повірки після виставлення необхідної витрати відбувається по натисненню кнопки BitBtn3 «Старт». Функція обробки події натиснення даної кнопки викликає функцію Obchisl(), яка виконує підрахунок імпульсів до натиснення кнопки Button2 «Стоп» або при зрівнянні підрахованих імпульсів

робочого лічильника з числом імпульсів введеним в поле Edit1. Підрахунок імпульсів відбувається апаратно, а програма зчитує дані за допомогою функції Update(). Для зчитування даних з лічильників необхідно періодично їх опитувати, що в програмі реалізовано за допомогою потоку. Для читання показів лічильників Nr (робочого) і Ne (еталона) спочатку створюється призупинений потік CounterThread(CounterSuspend) і після натиснення кнопки «Старт» в потоці періодично відбувається читання підрахованої кількості імпульсів і виведення її на екран за допомогою елементів StaticText. Одночасно для обчислення похибки в потоці ADC_Thread(ADCSuspend) відбувається періодичне читання масиву даних Data[] з АЦП і виведення відповідних оцифрованих значень температури ті тиску на екран з допомогою компонентів StaticText. Значення температури виводяться через компоненти StaticText1- StaticText7, а значення тиску за допомогою StaticText9- StaticText11. Зліва від значень, що виводяться за допомогою елементів Label підписано якому давачеві відповідає відповідне виведене значення. Підрахована кількість імпульсів робочого лічильника перетворюється в об'єм за допомогою функції NtoQ(int n), яка повертає результат ділення числа імпульсів на глобальну змінну коефіцієнта перетворення Кр1. Аналогічно підрахована кількість імпульсів еталона перетворюється в об'єм за допомогою функції NtoQ(int n), яка повертає результат ділення числа імпульсів на глобальну змінну коефіцієнта перетворення Ket.

Повірка зупиняється автоматично при отриманні вказаного числа імпульсів на робочому лічильнику або при натисненні кнопки «Стоп», яка викликає функцію зупинки повірки KnopkaStop().

Обчислення похибки проводиться функцією Delta(), в якій визначається температура робочого лічильника tRobValue, температура еталонного лічильника tEtValue, тиск еталона pEt і обчислюється похибка dlt. Всі вище перелічені отримані дані виводяться за допомогою StaticText на екран. Робочий еталон, що використовується позначається за допомогою компонентів RadioButton1- RadioButton3, які об'єднані за допомогою GroupBox1.

Вибір точки повірки відбувається натиснення кнопки Button4 «Наступна точка», що індукується на екрані у за допомогою елементів Image8- Image11 (Біля вибраної точки розміщується вказівник Image).

Повернення до попередньої форми початкової ініціалізації відбувається по натисненні кнопки BitBtn1 «Назад» за допомогою функції приховання форми повірки MainFormMy->Hide(), та показу форми початкової ініціалізації за допомогою функції SecondForm->Show().

Вихід з програми здійснюється по натисненні кнопки BitBtn2 «Вийти» з використанням функції MyProgramExit().

Вікно встановлення витрати об’єднує в собі всі необхідні засоби для забезпечення встановлення необхідної витрати як в ручному, так і в автоматичному режимі.

В даному розділі програми повинні виконуватися наступні пункти:

- індикація положення засувки та регулятора витрати;
- відкриття (закриття) засувок та регулятора витрати;
- вибір режиму виставлення витрати;
- автоматичне визначення еталона по введеній витраті;
- автоматична комутація засувок;
- автоматичне виставлення заданої витрати;
- вимірювання поточної витрати;
- обчислення допустимих меж витрати, що виставляється;

Для реалізації поставленого завдання на форму вікна програми необхідно помістити:

- Button – стандартна кнопка Windows, яка при натисненні виконує певну функцію;
- RadioButton – радіо кнопка призначена для вибору тільки одного з декількох варіантів;
- Панель GroupBox – це контейнер з рамкою і надписом, що об’єднує групу зв’язаних елементів;

- Label – елемент призначений для відображення текстової інформації, який не міняється користувачем. В даному випадку використовувався для надписів;
- Edit – вікно редагування для введення одного рядка символів;
- BitBtn – кнопка з можливістю відображення піктограми;
- CGauge – елемент відображення ходу процесу;
- StaticText – елемент призначений для швидкозмінної текстової інформації;
- ProgressBar – елемент прогресу виконання операції.

Для виставлення витрати в автоматичному режимі необхідно ввести в поле Edit2 потрібне значення витрати, в панелі вибору режиму встановлення витрати вибрати автоматичний і тнатиснути кнопку «Запуск Авт.» За допомогою функції ZasuvkiSetToBe() буде автоматично скомутовано засувки у відповідності до введеної витрати. Після чого за допомогою функції VitrataSet(StrToFloat(Edit2->Text)) буде виконуватись автоматичне встановлення витрати, шляхом зупинки запущених потоків, запуск потоку визначення витрати та потоку управління регулятором. При виставленні витрати яка входить в обчислені допустимі межі потік управління регулятором зупиняється.

Виміряне значення витрати виводиться на екран через StaticText25.

При виставленні витрати в ручному режимі запускається тільки потік виміру витрати, а управління регулятором відбувається в ручному режимі шляхом формування сигналів встановленням певних бітів в одиничні рівні.

Для відкритя і закриття засувки D50 використовуються відповідно функції _50Up() і _50Down().

Для відкритя і закриття засувки D80 використовуються відповідно функції _80Up() і _80Down().

Аналогічно для відкритя і закриття засувки D200 використовуються відповідно функції _200Up() і _200Down().

Встановлення засувок відбувається по натисненні кнопки Button5 шляхом визначення поточного стану засувок за допомогою функції

PerevirkaZasuvok(),створенням форми роботи засувок, створенням потоку опитування засувок і самою комутацією засувок за допомогою функції PerevirkaZasuvokToBeDone(StrToFloat(Edit2->Text)). Після встановлення засувок знищується форма роботи засувок і потік опитування.

Розробка сервісних функцій меню програми

Меню програми складається з двох частин (Файл і Додаткові можливості) і містить наступні функції:

- сворення нового протоколу;
- звавантаження;
- збереження;
- перегляд;
- друк протоколу;
- завершення програми;
- зміна параметрів еталонів;
- калібрування давачів;

При виборі в меню пункту «Новий» відбувається перевірка, чи був модифікований поточний протокол, якщо так то виводиться на екран діалогова форма з запитом на збереження поточного протоколу перед створенням нового. При натисненні кнопки «Так» в діалоговому вікні за допомогою функції SaveReportFile() на екран буде викликано стандартне вікно Windows збереження файлу. Після збереження поточного протоколу функцією NewOneCounter() буде створено новий чистий бланк протоколу.

При виборі в меню пункту «Завантажити» на екрані зявиться діалогове вікно запиту на збереження поточного протоколу аналогічно попередньому описові. Після чогоза допомогою функції Execute() невізуального компоненту OpenDialog1 буде відкрито (завантажено) вибраний протокол.

Вибравши в меню «Файл» пункту «Зберегти» протокол буде збережено за допомогою функції SaveReportFile().

Для перегляду протоколу використовується функція ainFormMy->QuickRep1->Preview().

При виборі пункту «Друк» в меню «Файл» відбудеться перевірка чи введено значення втрати тиску за максимальної витрати. Якщо втрата тиску не булавведена, то на екран виводиться вікно з проханням ввести дане значення, після чого буде здійснено друк протоколу з використанням функції MainFormMy->QuickRep1->Print().

При виборі в меню «Файл» пункту «Вихід» або натисненні комбінації клавіш Ctrl+X за допомогою функції MyProgramExit() буде завершено програму.

При виборі в меню «Додаткові можливості» пункту «Зміна параметрів еталонів» буде створено форму зміни коефіцієнтів еталонних лічильників, яка буде доступна тільки після введення правильного пароля.

Розробка форми протоколу повірки лічильників

Форма протоколу повірки промислових лічильників газу побудована за допомогою компоненти поля QuickRep1 на якому за допомогою елементу QRShape створено сітку таблиці, що заповнена відповідними записами з використанням елементів для відображення текстової інформації QRLabel.

Для введення номера протоколу, типу лічильника, показів лічильника після повірки, втрату тиску на лічильнику за максимальної витрати, прізвище, ім’я і по-батькові державного повірника використано вікно редагування для введення одного рядка символів Edit.

Над даним протоколом можуть виконуватися операції збереження, завантаження, перегляду та друку за допомогою відповідних пунктів меню «Файл».

Результати роботи програми

Запуск програми здійснюється з меню Пуск->Програми-> Повірка лічильників -> Повірка лічильників. Після запуску з’являється форма яка

зображена на рисунку 3.1 що містить інформацію про версію Програми, її розробника і користувача

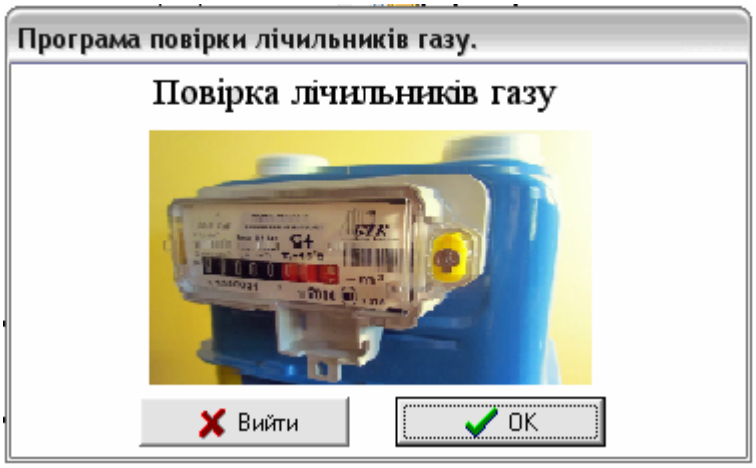


Рисунок 3.1 - Форма запуску програми повірки лічильників газу

Натискання кнопки «Вийти» завершує роботу Програми, а кнопки «ОК» – викликає форму початкової ініціалізації програми, що зображена на рисунку 3.2.

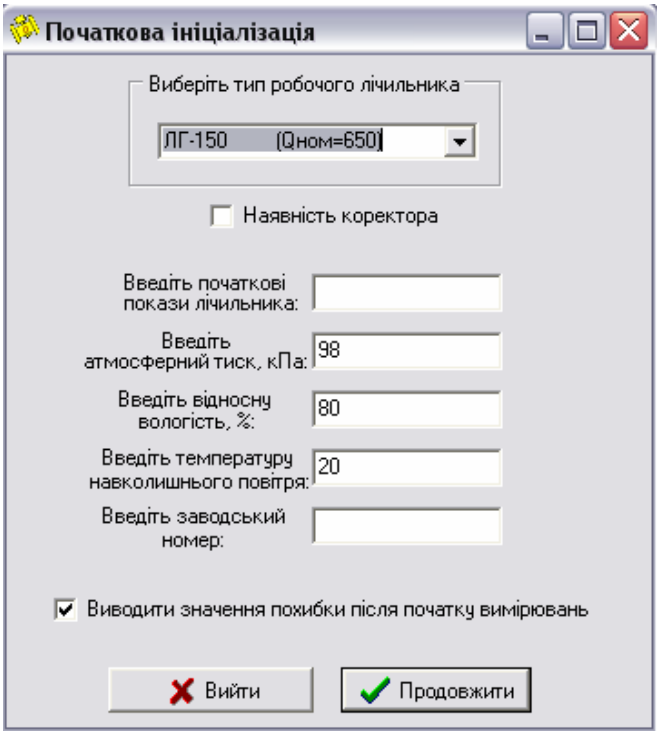


Рисунок 3.2 – Форма початкової ініціалізації програми

На цій формі за допомогою контекстного меню необхідно вибрати тип лічильника, газу, повірку якого необхідно провести. Якщо лічильник має вихід

на коректор, необхідно відмітити пункт “Наявність коректора”. Далі необхідно ввести початкові покази лічильника, атмосферний тиск, відносну вологість повітря, температуру навколишнього середовища і заводський номер лічильника. Якщо не відмічено пункт «Виводити похибку після початку вимірювання», обчислення відносної похибки лічильника буде проводитись після проходження об’єму, що відповідає 200 імпульсів з лічильника, що повіряється.

Якщо у стандартному списку типів лічильників немає потрібного, необхідно вибрати з меню пункт «Інший**…**». Тоді з’явиться форма, що зображена на рисунку 3.3.

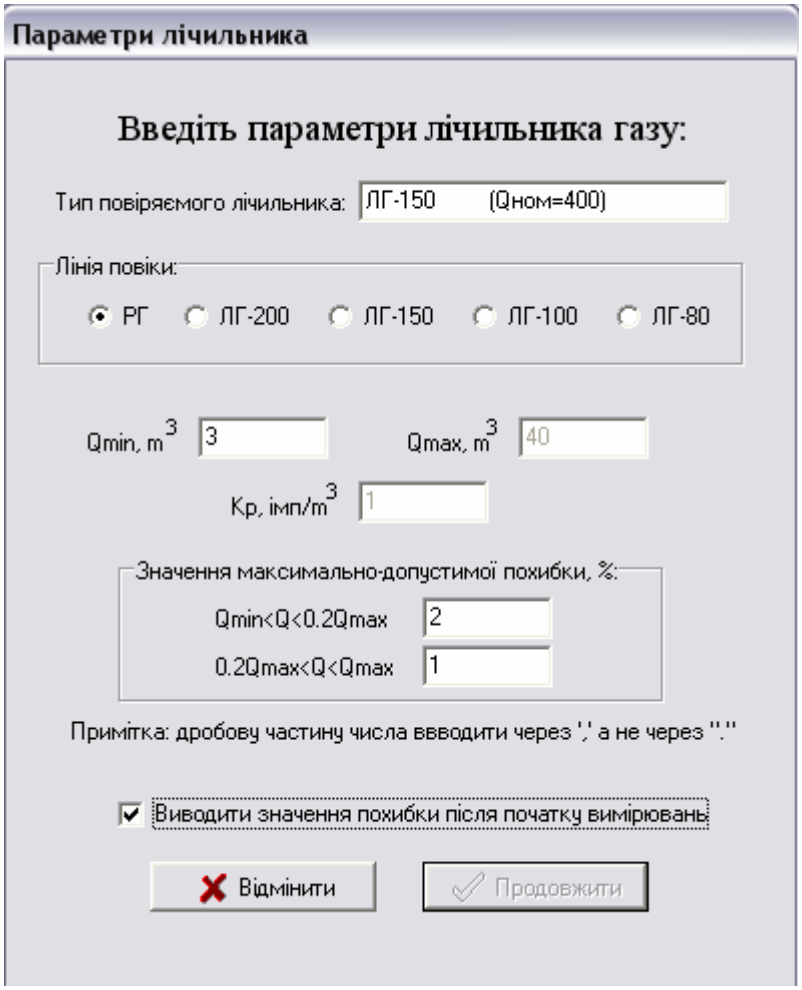


Рисунок 3.3 – Форма вводу параметрів лічильника

На цій формі необхідно ввести назву лічильника, вибрати по якій лінії буде проводитися повірка, мінімальну і максимальну витрату лічильника, його

коефіцієнт перетворення (кількість імпульсів на 1м^3), а також максимально-допустимі значення похибки в діапазоні від Qmin до 0.2Qmax, та від 0.2Qmax до Qmax. Останні значення необхідні для формування висновку про придатність чи непридатність лічильника в його протоколі повірки. Кнопка «Відмінити» повертає користувача до попередньої форми і не зберігає зміни, кнопка «Продовжити» – те саме, але із збереженням нових даних про лічильник.

Кнопка «Вийти» форми, завершує роботу програми, кнопка «Продовжити» викликає основну форму Програми.

Вона зображена на рисунку 3.4

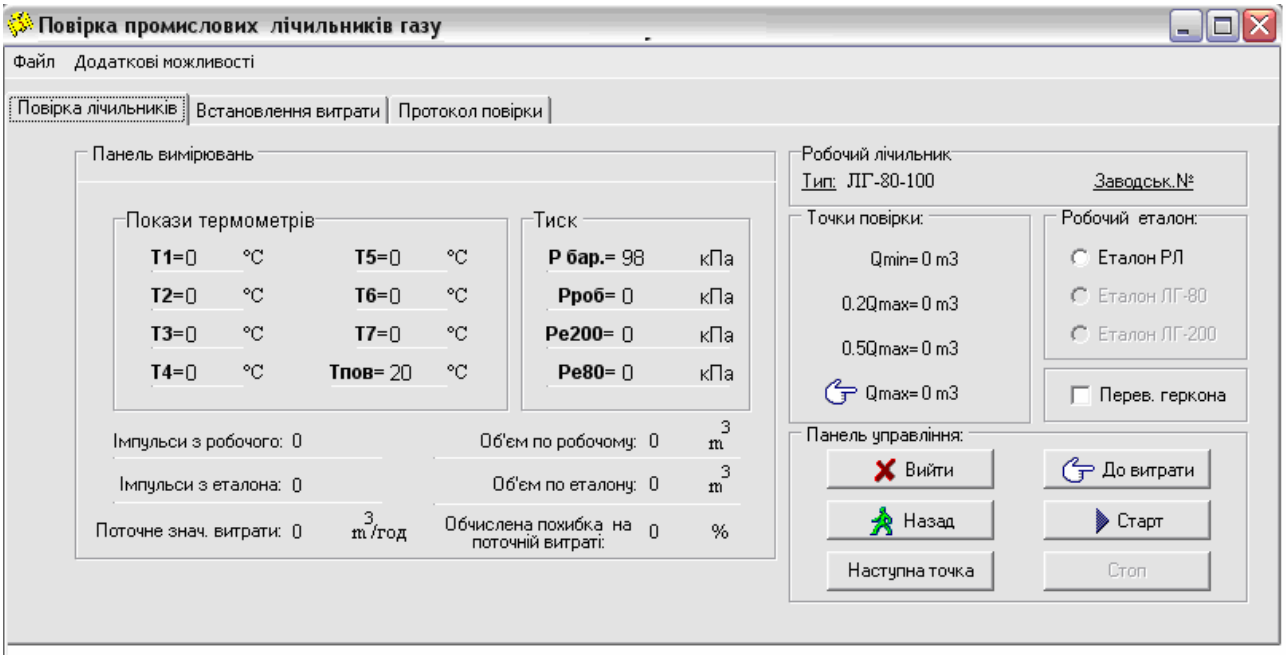


Рисунок 3.4 – Основна форма програми

Як видно з рисунка, основна форма складається з трьох сторінок-вкладок: «Повірка лічильників», «Встановлення витрати», «Протокол повірки».

На сторінці «Повірка лічильників» (рис 3.4) розміщені:

- «Панель вимірювань», яка відображає інформацію з давачів системи. Позначення давачів – згідно з рисунком, який розміщений нище цієї панелі ;

- панель «Робочий лічильник», на якій показано тип і заводський номер лічильника, що повіряється;
- панель «Точки повірки» відображає точки на яких проводиться повірка вибраного лічильника. Вказівник показує, на якій точці проводиться повірка
- в даний момент. Витрати, на яких відбулася повірка відмічаються відповідними значками;
- панель «Робочий еталон» показує який еталон використовується при повірці на поточній витраті;
- «Панель управління» містить такі кнопки:
  - «Вийти»– вихід з програми;
  - «Назад»– повернення до форми «Початкова ініціалізація». Служить для того випадку, коли потрібно змінити тип лічильника чи інші його параметри;
  - «Наступна точка» – переводить програму до наступної точки повірки. В автоматичному режимі відбувається переключення (при необхідності) робочого еталона (що демонструється на панелі) і обчислення його коефіцієнта перетворення на цій точці;
  - «До витрати» переводить користувача до сторінки «Встановлення витрати»;
  - «Старт»– починає процес повірки на вибраній точці. При цьому відбувається обнулення лічильників імпульсів (робочого і еталонного), а також обнулення результатів повірки в протоколі;
  - «Стоп» – примусова синхронна (по фронту зростання імпульса з робочого лічильника) зупинка процесу повірки на вибраній точці. Всі показники фіксуються такими, якими вони були на момент зупинки процесу;
  - в полі «Задана кількість імпульсів на робочому лічильнику» вказується кількість імпульсів з робочого лічильника, яку необхідно отримати для проведення повірки на вибраній точці. Після того, як з лічильника надійде

- останній імпульс , відбувається синхронна зупинка лічильників імпульсів і обчислення похибки на поточній витраті з врахуванням поправок по температурі і тиску. Нище розміщена полоса, яка в відсотках відображає хід виконання процесу повірки на вибраній точці. Час, що залишився до закінчення повірки на точці індикується на панелі, що знаходиться нище полоси;
- на панелі параметри лічильників показано коефіцієнти перетворення робочого і еталонного (на вибраній точці) лічильників;

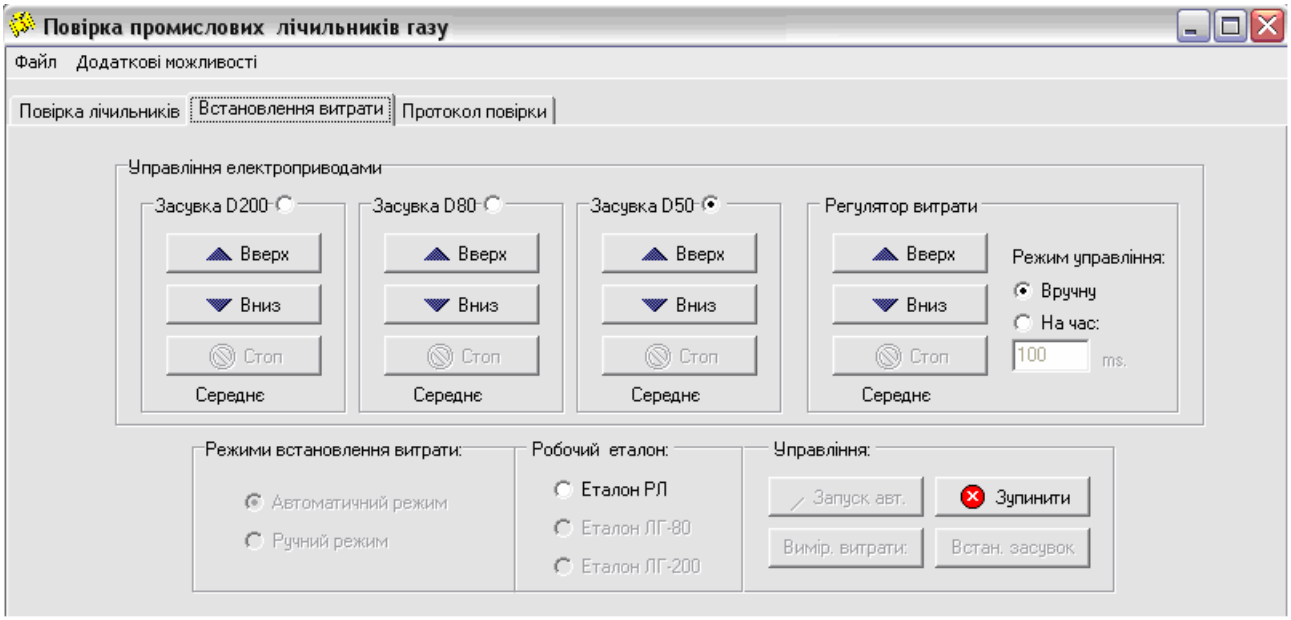


Рисунок 3.5 – Форма встановлення витрати

На сторінці «Встановлення витрати», яка зображена на рисунку 3.5 розміщені:

- панель управління електроприводами в ручному режимі, на якій для кожної засувки і регулятора знаходяться кнопки їх відкривання («Вверх»);
- закривання («Вниз») та зупинки («Стоп»). Панель активна тільки при ручному режимі виставлення витрати;
- панель «Режими встановлення витрати» дозволяє обрати ручний чи автоматичний режими управління електроприводами;

- панель «Робочий еталон» показує який еталон працює на поточній витраті;
- панель «Управління» містить кнопки «Запуск авт.» і «Зупинка» (активні в автоматичному режимі) відповідно для запуску і зупинки процесу встановлення витрати в автоматичному режимі, а також кнопки «Вимір. Витрати» і «Встан. Засувок», які активні в ручному режимі і служать відповідно для вимірювання витрати, а також для комутації засувок у відповідне до типу робочого еталона положення;
- панель «Потрібна витрата» містить поле, в якому можна задати нестандартне значення витрати;
- повідомлення «Значення частоти, Гц» показує частоту імпульсів з активного робочого еталона, яка відповідає значенню витрати, що відображається повідомленням «Значення витрати, $м^3$/год»;
- полоса «Обчислення витрати» показує хід обчислення витрати;
- повідомлення « Чекайте, встановлюється витрата**…** », « Працює регулятор » або «Стабілізується еталон» з'являється після початку встановлення витрати в автоматичному режимі;
- повідомлення «Номер ітерації» показує число відкривань/закривань регулятора витрати при встановленні поточної витрати.

На сторінці «Протокол повірки», яка зображена на рисунку 3.6 знаходиться протокол повірки лічильника газу, який формується автоматично відповідно до даних отриманих в процесі повірки лічильника. В протоколі неможливо поміняти вручну будь-які значення, окрім типу лічильника, який можна підкоректувати в спеціальному полі.

Меню «Файл», яке зображено на рисунку 3.7 містить наступні пункти:

«Новий» – обнуляє всі дані протоколу повірки;
- «Завантажити» і «Зберегти»– служать для роботи з файлами повірки (відкривання існуючих і збереження нових);
- «Перегляд Протоколу» і «Друк Протоколу» – відповідно попередній перегляд і друк протоколу повірки лічильника;
- «Вихід» – завершує роботу з програмою.

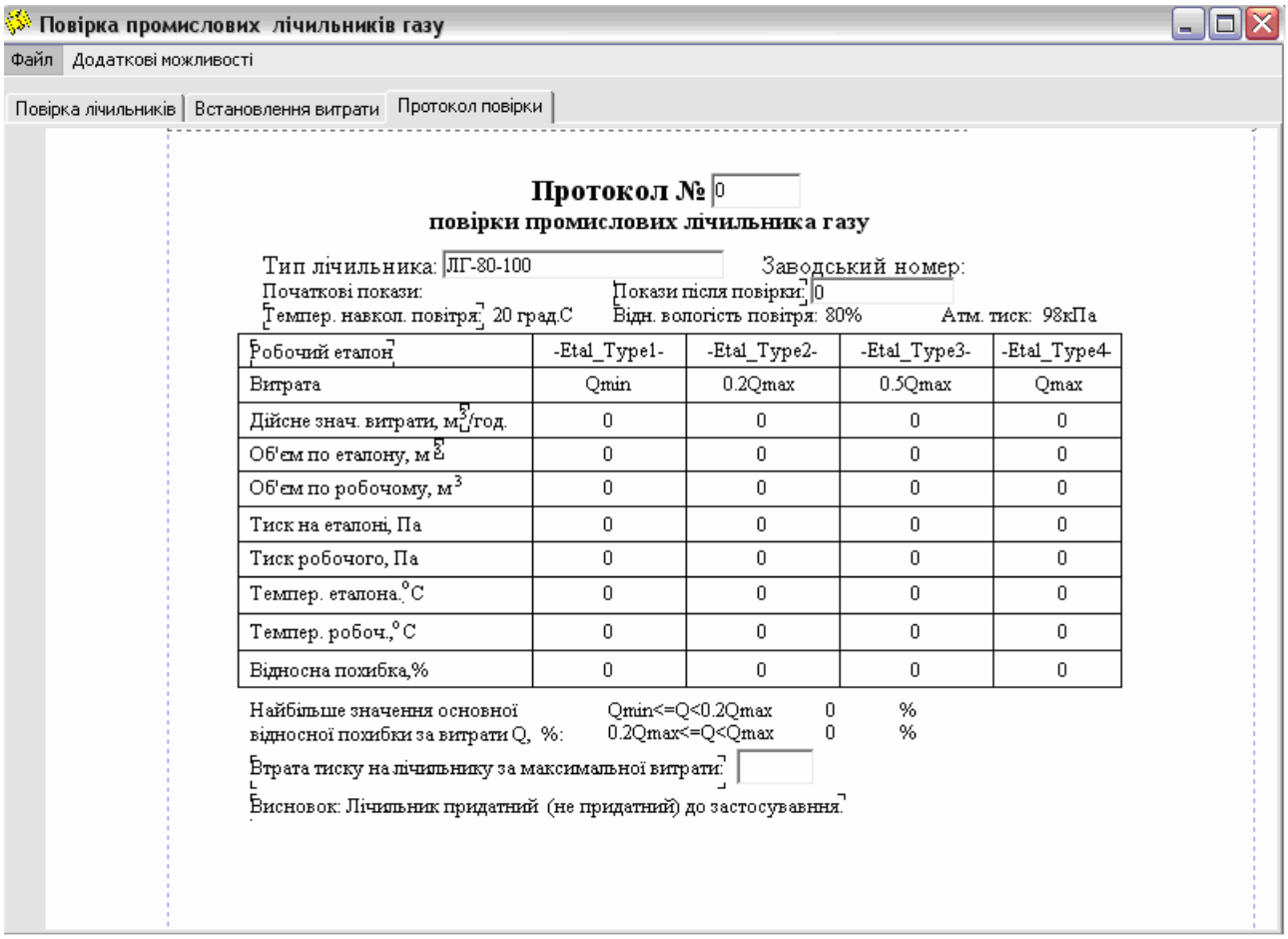


Рисунок 3.6 – Форма протоколу повірки промислових лічильників газу

ВИСНОВКИ

В дипломній роботі бакалавра було виконано розробку програмного рішення обробки метрологічних даних, а саме для процесів повірки лічильників газу.

Для цього в роботі виконано огляд сучасних методів та систем повірки газових лічильників, проведено опис інформаційно-вимірювальних комплексів та програмних комплексів для повірки та градуювання газових лічильників.

Виконано розробку алгоритмів функціонування програмного забезпечення, сформульовано вимоги до алгоритму.

Здійснено опис вибраного середовища розробки програмного продукту повірки промислових лічильників газу. Для реалізації програми було вибрано C++ Builder, оскільки саме це програмне середовище на сьогоднішній день є найбільш довершеним візуальним середовищем швидкої розробки програмних додатків на C++ для операційної системи Windows. Також в роботі детально описано всі функціональні кнопки розробленого програмного рішення та блоки реалізовані за допомогою програми.

ПЕРЕЛІК ПОСИЛАНЬ НА ДЖЕРЕЛА

1. Прикладная теория цифровых автоматов/К.Г.Самофалов и др. – К.: Вища шк.,1987.
2. Скаржепа В.А., Луценко А.Н. Электроника и микросхемотехника. Ч.1. Электронные устройства информационной автоматики. – К.: Вища шк., 1989.
3. Архангельський А. Я. C++ BUILDER 6. Справочное пособие. Книга 1. Язык C++. – М.: Бином-Пресс, 2002 г. – 544 с.: ил.
4. Архангельський А. Я. C++ BUILDER 6. Справочное пособие. Книга 2. Классы и компоненты. – М.: Бином-Пресс, 2002 г. – 528 с.: ил.
5. Зубков С.В. Assembler для DOS, Windows и UNIX –3-е изд., стер. – М.:ДМК Пресс; СПб.:Питер, 2004.–608с.
6. Фронковяк Д. М. Microsoft SQL: Руководство администратора.– М.:Русская Редакция, 200.–672с.
7. http://www.bcbdev.ru
8. М. Теллес. Borland C++ Builder: бібліотека програміста.- СПБ.: Пітер Ком, 1998.- 512с.
9. Павлов С.П., Губонина З.И. Охрана труда в приборостроении. – М.: Высшая школа, 1986.
10. Денисенко Г.Ф. Охрана труда. – М.: Высшая школа, 1985.
11. Домин П.А. Справочник по технике безопасности. – М.: Энергоиздат, 1984.
12. Мелкумов Я.С. Оценка эффективности инвестиций и финансирование инвестиционных проектов. – М.: ИКЦ “ДИС”, 1997. – 160 с.
13. Івано-Франківське СКБ ЗА. Програмне забезпечення. – Електронний ресурс. – Режим доступу - http://www.skbza.if.ua/prg_3ua.htm
14. Електронний ресурс. – Режим доступу - http://samgas.com.ua/ru/4clients/faq.html

15. Електронний ресурс. – Режим доступу - http://www.ukrcsm.kiev.ua/index.php/ru/2009-02-03-14-12-12/677-chto-nuzhno-znat-o-poverke-schetchika-gaza

16. Електронний ресурс. – Режим доступу - http://forum.liga.net/Messages.asp?did=220193

17. Електронний ресурс. – Режим доступу - http://gaz.lg.ua/content/poverka-schetchika

18. Електронний ресурс. – Режим доступу - http://www.gas.lipetsk.ru/viewpage.php?page_id=109

19. Електронний ресурс. – Режим доступу - http://www.eposignal.ru/products/catalog/91/624/

20. Електронний ресурс. – Режим доступу - http://gp-servis.ru/pages/upsg.html

21. Електронний ресурс. – Режим доступу - http://www.gpservis-nn.ru/publik/poverka-na-visokom-davlenii.html

22. Електронний ресурс. – Режим доступу - http://russia.bestpravo.ru/ussr/data02/tex13658.htm

23. Електронний ресурс. – Режим доступу - http://www.kip.spektr.org/flow.html

24. Електронний ресурс. – Режим доступу - http://ru.wikipedia.org/wiki/C%2B%2B_Builder

25. Електронний ресурс. – Режим доступу - https://www.embarcadero.com/ru/products/cbuilder

26. Електронний ресурс. – Режим доступу - http://forum.codeby.net/topic16508.html

27. Електронний ресурс. – Режим доступу - http://www.firststeps.ru/cbuilder/r.php?1

28. Метрологія. Повірка засобів вимірювальної техніки. Організація та порядок проведення: ДСТУ 2708:2006. — [Чинний від 2006–02–03]. — К. :

Держспоживстандарт України, 2006. —18с. — (Національні стандарти України)

29. Метрологія. Державна повірочна схема для засобів вимірювання об’єму та об’ємної витрати газу: ДСТУ 3383:2006. — [Чинний від 2007-29-01]. — К.: Держспоживстандарт України, 2007. — 10с. — (Національні стандарти України).

30. Стан та тенденції розвитку метрологічного забезпечення обліку газу в Україні [Електронний ресурс]. — Режим доступу: http://tools.mirohost. net/news/index.php?mhnews_id=&mhnews_newsid = 5584

ДОДАТКИ

Додаток А. Фрагменти програмного коду

FirstForm.~cpp

```
//---------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop
#include "FirstForm.h"
#include "SecondForm.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TFirstForm *FirstForm;
//---------------------------------------------------------------------------
__fastcall TFirstForm::TFirstForm(TComponent* Owner)
    : TForm(Owner)
{
}
//---------------------------------------------------------------


void __fastcall TFirstForm::BitBtn1Click(TObject *Sender)
{
    SecondForm = new TSecondForm(Application);
    SecondForm->Show();
    FirstForm->Hide();
}
//---------------------------------------------------------------
void __fastcall TFirstForm::BitBtn2Click(TObject *Sender)
{
   Application->Terminate();
}
//---------------------------------------------------------------
```

Фрагмент MainFormMy.~cpp

```
#include <vcl.h>
#include <math.h>
#include <stdio.h>
#include <string.h>
#pragma hdrstop
#include "ntport.h"
#include "MainFormMy.h"
#include "SecondForm.h"
#include "ADC_Thread.h"
#include "Count_Thread.h"
#include "QAutoThread.h"
#include "CounterQ.h"
#include "OprosZasuvok.h"
#include "RobotaZas.h"
#include "ChangeSensorKoefCPP.h"
#include "ChangeEtalon.h"
#pragma package(smart_init)
#pragma link "CGAUGES"
#pragma resource "*.dfm"
TMainFormMy *MainFormMy;

//*************************************************************
// -------------------------Глобальні змінні--------------------
// -------------------------------------------------------------

String FileDataNameMy="default";
```

```
ADC_Thread *ADC;
CounterThread *Counter;
CounterQ *CounterVThread;
QAutoSetThread *QAutoSet;
Zasuvki_Read *Zasuvki_Status_Read;

  double Data[12];    // Масив значень даних з АЦП
  int Nr; // кількість імпульсів робочого лічильника
  int Ne; // кількість імпульсів еталонного лічильника

  float Ket=1;  // коефіцієнт перетворення еталона на поточній витраті
(апроксимований)
  int tEtTip=0;
  int tRobTip;

extern int t_Rob_tip;     // номер давача температури

int TimeDelay=150;  // Час затримки включення електроприводів
int TimeWaiting=0;

extern int TimesOfIteration;

extern float MaxDelta02, MaxDelta, PrecisionClass;

bool CounterSuspend=true;
bool ADCSuspend=true;
bool CounterVSuspend=true;
bool QAutoSuspend=true;
float Kp1, Kp_before;

int pEtTip=0;       // Номер датчика тиску (на ЛГЕ-200 pEtTip=2; на ЛГЕ-80
pEtTip=1; на РЛЕ pEtTip=2;

//  Масив коефіцієнтів перетворення еталону ЛГ-200
float Ket200[2][20];
//  Масив коефіцієнтів перетворення еталону ЛГ-80
float Ket80[2][20];
//  Масив коефіцієнтів перетворення еталону РЛ
float KetRL[2][20];

int k_i=0;  // Коефіцієнт для множення при переповненні (65536) лічильників
імпульсів
int Ne1_previous=0;

float volog=0,tisk=0; //Початкові параметри
//String n_beg;
float N;

char A0T=0;        //  bit 0 -   A0(t)
char A1T=0;        //  bit 1 -   A1(t)
char OET=0;        //  bit 2 -   OE(t)
char Start=0;      //  bit 3 -   Start/Stop.  [0/1]
char Ruch=0;       //  bit 4 -   Ручн/Авт.    [0/1]
char Reset=0;      //  bit 5 -   Reset        [0/1/0]
char A0C=0;        //  bit 6 -   A0(c)
char A1C=0;        //  bit 7 -   A1(c)
//-------------------------------------------------
char RegUp=0;      //  bit 8 -   Регулятор вверх
char RegDown=0;    //  bit 9 -   Регулятор вниз
char Z200_UP=0;      //  bit 10 -  Засувка 200 вверх
char Z80_UP=0;      //  bit 11 -  Засувка 80 вверх
char Z50_UP=0;      //  bit 12 -  Засувка 50 вверх
char Z200_Down=0;    //  bit 13 -  Засувка 200 вниз
char Z80_Down=0;    //  bit 14 -  Засувка 80 вниз
```

```
char Z50_Down=0;    //  bit 15 -  Засувка 50 вниз

extern bool D200_ready_ok;
extern bool D80_ready_ok;
extern bool D50_ready_ok;


// Час відкриття/ закриття затворів і засувок в секундах
int TimeD50=19;
int TimeD80=21;
int TimeD200=31;
int TimeThrough=115;

// Змінні, що показують відкриті чи закриті засувки
bool D50IsOpen=false;
bool D80IsOpen=false;
bool D200IsOpen=false;
bool ThroughIsOpen=false;
bool D50_Prom=false;
bool D80_Prom=false;
bool D200_Prom=false;
bool Through_Prom=false;

// Змінні, що показують які (закриті чи відкриті)
//          повинні бути засувки
bool D50ToBeOpen=false;
bool D80ToBeOpen=false;
bool D200ToBeOpen=false;

float k200=31.49;   // -коефіцієнт пропорційності для ЛГЕ-200 для визначення
float k201=3982.4;  // кількості імпульсів за витратою: n=k200*Q+k201
                    // для часу повірки 3,6хв.-1хв. на витратах 250 і 2500
відповідно.

float k80=1483.7333;// -коефіцієнт пропорційності для ЛГЕ-80 для визначення
float k81=46367;    // кількості імпульсів за витратою: n=k80*Q+k81
                    // для часу повірки 7,2хв.-3,6хв. на витратах 25 і 250
відповідно.

float k10=3818.3636;// -коефіцієнт пропорційності для ЛГЕ-10 для визначення
float k11=30546;    // кількості імпульсів за витратою: n=k10*Q+k11
                    // для часу повірки 20хв.-7,2хв. на витратах 3 і 25
відповідно.

float ADC_Koef[2][10]={{0,0,0,0,0,0,0,0,0,0},
                       {0,0,0,0,0,0,0,0,0,0}};  // масив коефіцієнтів для
обчислення показів давачів


int krok=0;
extern float Vitraty[4];
bool RepModified=false;    // Показує чи змінений протокол після збереження


//---------------------------------------------------------------
__fastcall TMainFormMy::TMainFormMy(TComponent* Owner)
    : TForm(Owner)
{
}

//---------------------------------------------------------------------------
void TMainFormMy::ReadCounterProperties(void)
                                    //Встановлює глобальні параметри
{                                   //для робочого лічильника
```

```
   Edit2->Text=FloatToStrF(Vitraty[krok],ffGeneral,4,4);
   Aproc(StrToFloat(Edit2->Text));

}

//---------------------------------------------------------------------------
float TMainFormMy::NtoQ(int n)   //перетворює імпульси в об"єм (для робочого)
{                                // використовуючи глобальну
  return (n/Kp1);                // змінну  Кр(коеф. перетворення)
}
//---------------------------------------------------------------------------
float TMainFormMy::NtoQEt(int ne)   //перетворює імпульси в об"єм (для еталона)
{                                   // використовуючи глобальну
  return (ne/Ket);                  // змінну  Кet(коеф. перетворення)
}

//---------------------------------------------------------------------------
float TMainFormMy::QtoN(float q)    //перетворює об"єм в імпульси (для робочого)
{                                   // використовуючи глобальну
  return (q*Kp1);                   // змінну  Кр1(коеф. перетворення)
}
//---------------------------------------------------------------------------
float TMainFormMy::QtoF(float vitr)  //перетворює витрату  в частоту (для
еталону)
{                                     // використовуючи параметр Кet
  return ((vitr/3600)*Ket);
}
//---------------------------------------------------------------------------
float TMainFormMy::FtoQ(float fr)   //перетворює    частоту в витрату(для
еталону)
{                                   // використовуючи параметр Кet
  return ((3600*fr)/Ket);
}

//---------------------------------------------------------------------------
void __fastcall TMainFormMy::Aproc(float vitrata) // апроксимація між точками
еталонних лічильників
{     int index;                                  // вхідний параметр - витрата
еталона
      float K1,K2,V1,V2;                          // вихідний параметр -
коефіцієнт перетворення
      if (vitrata<25)  // Також встановлює необхідний лічильник
                { goto RL10; }         // активним (необхідний еталон)
      if ((vitrata>=25)&&(vitrata<250)) // Встановлює кількість імпульсів N на
даній
                { goto LG80;  }        // витраті, що відповідає мінімальному
обєму
      if (vitrata>=250)
                { goto LG200; }

//---------------------------
RL10: MainFormMy->RadioButton1->Enabled=true;
      MainFormMy->RadioButton2->Enabled=false;
      MainFormMy->RadioButton3->Enabled=false;
      MainFormMy->RadioButton16->Enabled=true;
      MainFormMy->RadioButton18->Enabled=false;
      MainFormMy->RadioButton17->Enabled=false;
      tEtTip=5;   pEtTip=2;
      A0C=0;      A1C=1;      Update();  //   Включення необхідного каналу
мультиплексора
      N=k10*vitrata+k11;
      index=0;
      a: if (KetRL[0][index]<vitrata)
```

```
         {
            index++;
            goto a;
         }
         index--;

      V1=KetRL[0][index];    K1=KetRL[1][index];
      V2=KetRL[0][index+1];  K2=KetRL[1][index+1];
      Ket=K2+(K1-K2)*(V2-vitrata)/(V2-V1);
      goto end;
//---------------------------
LG80: MainFormMy->RadioButton1->Enabled=false;
      MainFormMy->RadioButton2->Enabled=true;
      MainFormMy->RadioButton3->Enabled=false;
      MainFormMy->RadioButton16->Enabled=false;
      MainFormMy->RadioButton18->Enabled=true;
      MainFormMy->RadioButton17->Enabled=false;
      tEtTip=6;  pEtTip=1;
      A0C=1;    A1C=0;      Update();  //   Включення необхідного каналу
мультиплексора
      index=0;
      b: if (Ket80[0][index]<vitrata)
         {  index++; goto b;   }

         index--;
      V1=Ket80[0][index];    K1=Ket80[1][index];
      V2=Ket80[0][index+1];  K2=Ket80[1][index+1];
      Ket=K2+(K1-K2)*(V2-vitrata)/(V2-V1);
      goto end;
//---------------------------
LG200: MainFormMy->RadioButton1->Enabled=false;
      MainFormMy->RadioButton2->Enabled=false;
      MainFormMy->RadioButton3->Enabled=true;
      MainFormMy->RadioButton16->Enabled=false;
      MainFormMy->RadioButton18->Enabled=false;
      MainFormMy->RadioButton17->Enabled=true;
      tEtTip=7;   pEtTip=2;
      A0C=0;    A1C=0;      Update();  //   Включення необхідного каналу
мультиплексора
      N=k200*vitrata+k201;
      index=0;
      c: if (Ket200[0][index]<vitrata)
         {  index++; goto c;   }

      index--;
      V1=Ket200[0][index];    K1=Ket200[1][index];
      V2=Ket200[0][index+1];  K2=Ket200[1][index+1];
      Ket=K2+(K1-K2)*(V2-vitrata)/(V2-V1);
end:
      MainFormMy->StaticText31->Caption=FloatToStrF(Ket*1.0,ffGeneral,7,7);   //
      return;
}
//---------------------------------------------------------
void __fastcall TMainFormMy::BitBtn1Click(TObject *Sender)
{                                    //  Кнопка назад
     MainFormMy->Hide();
     SecondForm->Show();
     if(SecondForm->BitBtn1->Enabled)
        SecondForm->BitBtn1->Enabled=false;
}
//---------------------------------------------------------------------------

void __fastcall TMainFormMy::BitBtn2Click(TObject *Sender)
{                                    //  Кнопка вийти
```

```
   /*  if (!CounterSuspend)
          Counter->Suspend();
     if (!ADCSuspend)
          ADC->Suspend();
     if (!CounterVSuspend)
          CounterVThread->Suspend();
     if (!QAutoSuspend)
          QAutoSet->Suspend();
    StopAll();
    Application->Terminate();*/
    MyProgramExit();
}
//---------------------------------------------------------------------------
//---------  Функції активізації сторінок головної форми ------------
}

void __fastcall TMainFormMy::OK_Activate(void)
{                               //Ставить "птички" напроти пройдених витрат

   switch(krok)                 // використовуючи номер кроку повірки
        {
        case 0:
            Image1->Visible=false;
            Image2->Visible=false;
            Image3->Visible=false;
            Image4->Visible=false;
            break;
        case 1:
            Image1->Visible=true;
            Image2->Visible=false;
            Image3->Visible=false;
            Image4->Visible=false;
            break;
        case 2:
            Image1->Visible=true;
            Image2->Visible=true;
            Image3->Visible=false;
            Image4->Visible=false;
            break;
        case 3:
            Image1->Visible=true;
            Image2->Visible=true;
            Image3->Visible=true;
            Image4->Visible=false;
            break;
        case 4:
            Image1->Visible=true;
            Image2->Visible=true;
            Image3->Visible=true;
            Image4->Visible=true;
            break;
        }
    return;
}

void __fastcall TMainFormMy::N4Click(TObject *Sender)
{
       if (Edit8->Text=="")
         {
             Application->MessageBoxA("Перед друком введіть втрату тиску на
лічильнику на максимальній витраті","Попередження!",MB_OK);
         }
       else
       MainFormMy->QuickRep1->Print();
```

```
}
//---------------------------------------------------------------------------

void __fastcall TMainFormMy::N5Click(TObject *Sender)
{
       MainFormMy->QuickRep1->Preview();
}
//---------------------------------------------------------------------------


void __fastcall TMainFormMy::BitBtn4Click(TObject *Sender)
{   //   До витрати
    String A;
    if (BitBtn1->Enabled)   BitBtn1->Enabled=false;

    Edit2->Text=FloatToStrF(Vitraty[krok],ffGeneral,4,4);
    Aproc(StrToFloat(Edit2->Text));
    A="Переходимо до виставлення витрати " + FloatToStr(Vitraty[krok]) + "
м.куб.";
    Application->MessageBoxA(A.c_str(),"Повірка лічильника",MB_OK);
    MainFormMy->SetTab2();
}
//---------------------------------------------------------------------------

void __fastcall TMainFormMy::FormCreate(TObject *Sender)
{
   FILE *koef;
   float a,b;
   krok=3;
   FingerActivate();
   QAutoSet= new QAutoSetThread(QAutoSuspend);

   ADC = new ADC_Thread(ADCSuspend);   // Спочатку ADCSuspend=true
                                       // Створення призупиненого потоку
                                       // читання масиву даних Data[] з АЦП
   CounterVThread = new CounterQ(CounterVSuspend);   // Спочатку ADCSuspend=true
                                       // Створення призупиненого потоку
                                       // читання масиву даних Data[] з АЦП
   Counter = new CounterThread(CounterSuspend);  // Спочатку CounterSuspend=true
                                        // Створення призупиненого потоку
                                        // читання показів лічильників Nr i Ne


   if ((koef=fopen("K_Sensor.dat", "r"))== NULL)
        {
          Application->MessageBox("Неможливо відкрити файл з
коефіцієнтами","Помилка!!!",MB_OK);
        }
   for(int iii=0;iii<10;iii++)
        {
           fscanf(koef,"%f\t",&a);
           fscanf(koef,"%f\n",&b);
           ADC_Koef[0][iii]=a;
           ADC_Koef[1][iii]=b;
        }
   fclose(koef);
   ReadEtalCoifficient();
}
//---------------------------------------------------------------------------

void __fastcall TMainFormMy::BitBtn3Click(TObject *Sender)
{  // кнопка Запустити

    RepModified=true;
```

```
    Kp1=StrToFloat(StaticText35->Caption);
   /*  Щоб не збивати мультиплексор, якщо задавати витрату вручну
    Edit2->Text=FloatToStrF(Vitraty[krok],ffGeneral,4,4);
   */
    //Aproc(StrToFloat(Edit2->Text));
    Obchisl();
    k_i=0;
    Ne1_previous=0;

    if(!QAutoSet->Suspended)
         {
          QAutoSet->Suspend();
          QAutoSuspend=true;
          }
    if(!CounterVThread->Suspended)
         {
         CounterVThread->Suspend();
         CounterVSuspend=true;
         }
    if(ADC->Suspended)
         {
          ADC->Resume();
          ADCSuspend=false;
          }
    if(Counter->Suspended)
         {
          Counter->Resume();
          CounterSuspend=false;
          }

    Button2->Enabled=true;
    BitBtn3->Enabled=false;
    BitBtn1->Enabled=false;
    BitBtn2->Enabled=false;
    BitBtn4->Enabled=false;
    Button4->Enabled=false;
}
//---------------------------------------------------------------------------


void __fastcall TMainFormMy::FormClose(TObject *Sender,
      TCloseAction &Action)
{
    MyProgramExit();
}
//---------------------------------------------------------------
//------------------Функції управління електроприводами----------
//---------------------------------------------------------------------------
//---------------------------------------------------------------------------
void __fastcall TMainFormMy::ThroughUp(bool OnTime,int MyTimeMS)
{
    RegDown=0;
    Update();
    Sleep(TimeDelay);
    RegUp=1;      // вверх
    Update();
    if(OnTime)
        {
            Sleep(MyTimeMS);
            RegUp=0;      // стоп
            RegDown=0;
            Update();
        }
}
```

```
//---------------------------------------------------------------------------
void __fastcall TMainFormMy::ThroughDown(bool OnTime,int MyTimeMS)
{

    RegUp=0;      // вниз
    Update();
    Sleep(TimeDelay);
    RegDown=1;
    Update();
    if(OnTime)
        {
            Sleep(MyTimeMS);
            RegUp=0;      // стоп
            RegDown=0;
            Update();
        }
}

//---------------------------------------------------------------------------
void __fastcall TMainFormMy::StopAll(void)
{
     _200Stop();
     _80Stop();
     _50Stop();
     _ThroughStop();
}

//---------------------------------------------------------------------

void __fastcall TMainFormMy::Button1Click(TObject *Sender)
{
// Кнопка Зупинити/запустити виставлення витрати РУЧН
    if(!QAutoSet->Suspended)
         {
          QAutoSet->Suspend();
          QAutoSuspend=true;
          }
    if(!ADC->Suspended)
         {
          ADC->Suspend();
          ADCSuspend=true;
          }
    if(!Counter->Suspended)
         {
          Counter->Suspend();
          CounterSuspend=true;
          }
   if (!CounterVSuspend)
      {
        CounterVSuspend=true;
        CounterVThread->Suspend();
        Button1->Caption="Вимір. витрати:";
        RadioButton4->Enabled=true;
        RadioButton5->Enabled=true;
        AprocOnStop();
      }
   else
     {
        CounterVSuspend=false;
        CounterVThread->Resume();
        Button1->Caption="Зупинити";
        RadioButton4->Enabled=false;
        RadioButton5->Enabled=false;
```

```
     }
}
//---------------------------------------------------------------------------
//---------------------------------------------------------------------------

void __fastcall TMainFormMy::Obchisl(void)
{
   A0T=1;  A1T=0;                    //  (Встановлення часу визначення витрати в
цій функції не використовуються)
   OET=1; Start=0; Ruch=0;
   Reset=1; k_i=0;                   // Імпульс онулення
   Ne1_previous=0;
   Update();                         //  Обновлення регістру управління
   Reset=0;Start=1;   Update();      // підрахунок імпульсів
                                     // поки не натиснута кнопка стоп
 }
void __fastcall TMainFormMy::Edit2Change(TObject *Sender)
{

Aproc(StrToFloat(Edit2->Text));

if(krok==0)
       {
        Label56->Caption="+5%  =";
        Label57->Caption="-0%  =";
        StaticText29->Caption=FloatToStrF(StrToFloat(MainFormMy->Edit2-
>Text)*1.05,ffGeneral,4,4);
        StaticText30->Caption=FloatToStrF(StrToFloat(MainFormMy->Edit2-
>Text)*1.0,ffGeneral,4,4);
       }

if((krok==1)||(krok==2))
       {
        Label56->Caption="+5%  =";
        Label57->Caption="-5%  =";
        StaticText29->Caption=FloatToStrF(StrToFloat(MainFormMy->Edit2-
>Text)*1.05,ffGeneral,4,4);
        StaticText30->Caption=FloatToStrF(StrToFloat(MainFormMy->Edit2-
>Text)*0.95,ffGeneral,4,4);
       }
if(krok==3)
       {
        Label56->Caption="+0%  =";
        Label57->Caption="-5%  =";
        StaticText29->Caption=FloatToStrF(StrToFloat(MainFormMy->Edit2-
>Text)*1.0,ffGeneral,4,4);
        StaticText30->Caption=FloatToStrF(StrToFloat(MainFormMy->Edit2-
>Text)*0.95,ffGeneral,4,4);
       }


}
//---------------------------------------------------------------------------


void __fastcall TMainFormMy::Button2Click(TObject *Sender)
{
     //Кнопка зупинити
     KnopkaStop();
}
//---------------------------------------------------------------------------


 void __fastcall TMainFormMy::Obchisl_Vitrata(void)
```

```
{
   A0T=1;   A1T=0;            // Встановлення часу визначення витрати
   OET=0;   Start=0; Ruch=0;
   Reset=1; k_i=0;            // Імпульс онулення
   Update();                  // Обновлення регістру управління
   Reset=0;
   Start=1;      Update();    // імпульс для початку підрахунку імпульсів
протягом часу T,
   Start=0;      Update();    // визначеного AOT і A1T
 }

//******************-------------------------------------------------------
void __fastcall TMainFormMy::KnopkaStop(void)
{

     Start=0;    Update();
     Delta();
     Button2->Enabled=false;
     BitBtn3->Enabled=true;
     BitBtn1->Enabled=true;
     BitBtn2->Enabled=true;
     BitBtn4->Enabled=true;
     Button4->Enabled=true;

     if(!QAutoSet->Suspended)
         {
          QAutoSet->Suspend();
          QAutoSuspend=true;
          }
     if(!CounterVThread->Suspended)
         {
         CounterVThread->Suspend();
         CounterVSuspend=true;
         }
     if(!ADC->Suspended)
         {
          ADC->Suspend();
          ADCSuspend=true;
//---------------------------------------------------------

void __fastcall TMainFormMy::BitBtn7Click(TObject *Sender)
{
    _200Down();
}
//---------------------------------------------------------------------------
void __fastcall TMainFormMy::_200Down(void)
{
    //              // 200 Вниз
    Z200_UP=0;
    Update();
    Sleep(TimeDelay);
    Z200_Down=1;
    Update();
    BitBtn7->Enabled=false;
    BitBtn8->Enabled=false;
    BitBtn9->Enabled=true;
    TimerPerevirZas->Enabled = true;
    Timer200->Enabled=true;
}
//---------------------------------------------------------------------------
void __fastcall TMainFormMy::BitBtn8Click(TObject *Sender)
{
     _200Up();
}
```

```
//---------------------------------------------------------------------------
void __fastcall TMainFormMy::_200Up(void)
{
    //               200 Вверх
    Z200_Down=0;
    Update();
    Sleep(TimeDelay);
    Z200_UP=1;
    Update();
    BitBtn9->Enabled=true;
    BitBtn7->Enabled=false;
    BitBtn8->Enabled=false;
    TimerPerevirZas->Enabled = true;
    Timer200->Enabled=true;
}
//---------------------------------------------------------------------------

void __fastcall TMainFormMy::BitBtn9Click(TObject *Sender)
{
         _200Stop();
}
void __fastcall TMainFormMy::_200Stop(void)
{
         Z200_UP=0;
         Z200_Down=0;
         Update();
         BitBtn9->Enabled=false;
         BitBtn7->Enabled=true;
         BitBtn8->Enabled=true;
         Timer200->Enabled=false;
}
//---------------------------------------------------------------------------

void __fastcall TMainFormMy::BitBtn12Click(TObject *Sender)
{
        _80Up();
}

void __fastcall TMainFormMy::BitBtn11Click(TObject *Sender)
{
        _80Down();

}
//---------------------------------------------------------------------------
void __fastcall TMainFormMy::_80Down(void)
{
    Z80_UP=0;
    Update();
    Sleep(TimeDelay);
    Z80_Down=1;
    Update();
    BitBtn11->Enabled=false;
    BitBtn12->Enabled=false;
    BitBtn10->Enabled=true;
    TimerPerevirZas->Enabled = true;
    Timer80->Enabled=true;
}
//---------------------------------------------------------------------------
void __fastcall TMainFormMy::BitBtn15Click(TObject *Sender)
{
        _50Up();
}
//---------------------------------------------------------------------------
void __fastcall TMainFormMy::_50Up(void)
```

```
{
    //            50 Вверх
    Z50_Down=0;
    Update();
    Sleep(TimeDelay);
    Z50_UP=1;
    Update();
    BitBtn13->Enabled=true;
    BitBtn14->Enabled=false;
    BitBtn15->Enabled=false;
    TimerPerevirZas->Enabled = true;
    Timer50->Enabled=true;
}
……
……

void __fastcall TMainFormMy::Timer200Timer(TObject *Sender)
{
      if (!D200IsOpen && Z200_Down && !D200_Prom)  // якщо вже закрито
         {
           Z200_Down=0;
           Update();
           BitBtn7->Enabled=false;
           BitBtn8->Enabled=true;
           BitBtn9->Enabled=false;
           Timer200->Enabled=false;
         }
      if (D200IsOpen && Z200_UP && !D200_Prom)     // якщо вже відкрито
         {
           Z200_UP=0;
           Update();
           BitBtn7->Enabled=true;
           BitBtn8->Enabled=false;
           BitBtn9->Enabled=false;
           Timer200->Enabled=false;
         }
      if (RadioButton8->Checked)  RadioButton8->Checked=false;
             else RadioButton8->Checked=true;
}
//---------------------------------------------------------------------------

void __fastcall TMainFormMy::Button3Click(TObject *Sender)
{
PerevirkaZasuvok();
}
//---------------------------------------------------------------------------


void __fastcall TMainFormMy::TimerPerevirZasTimer(TObject *Sender)
{
    PerevirkaZasuvok();
    if( !Z50_Down && !Z80_Down && !Z200_Down && !Z50_UP && !Z80_UP && !Z200_UP)
       TimerPerevirZas->Enabled = false;

}
//---------------------------------------------------------------------------
```

БІБЛІОГРАФІЧНА ДОВІДКА

Тема роботи: “Розробка програмного сервісу обробки метрологічних даних”

Кількість сторінок: 63

Дата закінчення роботи: 17 червня 2022 р.

Підпис _____