НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Радіотехнічний факультет
(повна назва інституту/факультету)

Кафедра радіоконструювання та виробництва радіоапаратури
(повна назва кафедри)

«На правах рукопису»
УДК 6218

«До захисту допущено»
Завідувач кафедри
(підпис) Є.А. Нелін (ініціали, прізвище)
"13" 12 2019 р.

Магістерська дисертація

за спеціальністю 172 Телекомунікації та радіотехніка
за освітньою програмою (спеціалізацією) Інтелектуальні технології мікросистемної радіоелектронної техніки
(код і назва спеціальності)

на тему: Автоматизація тестування пристроїв електричної енергії

Виконав (-ла): студент (-ка) 2 курсу, групи РІ-81 мп
(шифр групи)

Шевчик Дмитро Миколайович (прізвище, ім'я, по батькові) (підпис)

Науковий керівник к.т.н., доцент Горбатов С.Б. (посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант з охорони праці к.т.н., доцент Каштанов С.Ф (назва розділу) (науковий ступінь, вчене звання, , прізвище, ініціали) (підпис)

Рецензент к.т.н. доцент Піддубний В.О (посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент (підпис)

Київ – 2019 року

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

Факультет (інститут) радіотехнічний факультет
(повна назва)

Кафедра радіоконструювання та виробництва радіоапаратури
(повна назва)

Рівень вищої освіти – другий (магістерський) за освітньо-професійною програмою
За спеціальністю 172 Телекомунікації та радіотехніка
За освітньою програмою (спеціалізацією) Інтелектуальні технології мікросистемної радіоелектронної техніки

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Є. А. Нелін
(підпис) (ініціали, прізвище)
02 вересня 2019р.

ЗАВДАННЯ
на магістерську дисертацію студенту

Шевчику Дмитру Миколайовичу
прізвище, ім'я, по батькові)

1. Тема дисертації "Автоматизація тестування лічильників електричної енергії"

науковий керівник дисертації доцент Тарабаров С.Б.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «11» листопада 2019 р. № 3881-с

2. Строк подання студентом дисертації 16. 12. 2019р.

3. Об'єкт дослідження Програма автоматизованого контролю електричної енергії в сучасних лічильниках

4. Предмет дослідження (вихідні дані для магістерської дисертації за освітньо-професійною програмою) автоматизоване тестування параметрів імпульсів електричної енергії та дослідження їх похибок

5. Перелік завдань, які потрібно розробити ознайомитись з документацією існуючих імпульсів електричної енергії керованих мікроконтролером, розробити алгоритм автоматизованого тестування, розробити програму автоматизованого тестування

6. Орієнтовний перелік ілюстративного (графічного) матеріалу _____

7. Орієнтовний перелік публікацій _____

8. Консультанти розділів дисертації*

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
З охорони праці	Каштанов С.Ф., доцент ,к.т.н.		

9. Дата видачі завдання 02 вересня 2019 року

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Строк виконання етапів магістерської дисертації	Примітка
1.	Підбір літератури за темою МД	02.09.19 – 05.09.19	
2.	Розробка ТЗ	06.09.19 – 20.09.19	
3.	Ознайомлення з докум. імпульсів	21.09.19 – 03.10.19	
4.	Аналіз вимірювальної установки	05.10.19 – 12.10.19	
5.	Вибір програмного пакета	13.10.19 – 01.11.19	
6.	Розробка програми	05.12.19 – 05.12.19	

Студент _____ Шевчик Д.М.
(підпис) (ініціали, прізвище)

Науковий керівник дисертації _____ С.Ф. Каштанов
(підпис) (ініціали, прізвище)

* Консультантом не може бути зазначено наукового керівника

РЕФЕРАТ

Структура й обсяг дипломної роботи

Магістерська дисертація: 96 с., 20 рис., 26 таб., 6 додатків, 15 джерел.

Ключові слова. ТЕСТУВАННЯ, АВТОМАТИЗАЦІЯ, ЛІЧИЛЬНИК, ПРОГРАМА.

Актуальність теми. Розробка програми автоматизованого тестування з метою підвищення якості та швидкості тестування лічильників електричної енергії.

Мета дослідження. Дослідження особливостей автоматизованого тестування сучасних лічильників електроенергії.

Для реалізації мети були сформовані такі **завдання на магістерську дисертацію:** ознайомитись з документацією сучасних лічильників електричної енергії, керованих мікроконтролером, оглянути ринок калібрувальних установок для лічильників, проаналізувати протокол обміну даними між калібрувальною установкою та лічильником електричної енергії, що тестується, розробити драйвер для комунікації зовнішнього програмного забезпечення з установкою калібрування лічильників електричної енергії, розробити алгоритм автоматизованого тестування лічильників, розробити програму автоматизованого тестування сучасних лічильників**,** проаналізувати результати.

Об’єкт дослідження — тестування сучасних лічильників електричної енергії.

Предмет дослідження — автоматизоване тестування параметрів лічильників електричної енергії.

Практичне значення одержаних результатів виражається в потенціалі для подальшого розвитку.

ABSTRACT

Structure and volume of the thesis

Master's Thesis: 96 pp., 20 figs., 26 tabs., 6 supplements, 15 sources.

Keywords. TESTING, AUTOMATION, METER, PROGRAM.

Actuality of theme. Development of an automated testing program to improve the quality and speed of testing electricity meters.

The aim of the study. To accomplish this goal, the following tasks have been created for the master's documentation: to get acquainted with the documentation of modern electricity meters controlled by a microcontroller, to look at the market of calibration installations for meters, to analyze the protocol of installation installation, to create al-horythm of automation of testing of meters, to develop an external driver for installations calibration of electricity meters, create an automated testing program to control electricity in specific counters, analyze the results.

The object of study is testing of modern electricity meters.

The subject of the study is automated testing of the parameters of electricity meters.

The practical significance of the results is expressed in the potential for further development.

ПОЯСНЮВАЛЬНА ЗАПИСКА

до магістерської дисертації

на тему: Автоматизація тестування лічильників електричної енергії

Київ — 2019 рок

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАЧЕННЯ

АЦП — аналогово – цифровий перетворювач

ІТ — інформаційні технології

ПС — програмні системи

COM–порт — communications port — двонаправлений послідовний інтерфейс, призначений для обміну байтовою інформацією;

MDM — measurement deviation minus — похибка вимірювання в мінус

MDP — measurement deviation plus — похибка вимірювання в плюс

ВСТУП

Тестування лічильників є досить важливим для постачальника електричної енергії. Основний компонент лічильника – мікроконтролер - значно розширює можливості сучасного лічильника. При цьому, ручне тестування останнього вимагає значних часових витрат і пов'язане з можливими помилками. Засоби ж автоматизованого тестування дозволяють підвищити якість та зменшити час тестування.

Автоматизоване тестування в лічильниках електричної енергії передбачає використання спеціального програмного забезпечення. Контролювання енергії здійснюється порівнянням очікуваного фактичного результату програми і еталонного. Цей тип тестування допомагає автоматизувати часто повторювані, але й необхідні для максимізації тестового покриття завдання.

В такому виді тестування є багато плюсів в порівнянні з ручним тестуванням. Оскільки ручне тестування потребує значно більше часу і зусиль, а також присутня ймовірність похибки зняття результатів. Деякі задачі тестування (наприклад, низькорівневе регресійне тестування) можуть бути трудомісткими і вимагають багато часу при їх ручному виконанні. Крім того, ручне тестування може недостатньо ефективно знаходити деякі типи помилок. У таких випадках автоматизація може допомогти заощадити час і зусилля проектної команди при тестуванні лічильників електричної енергії.

1 АНАЛІЗ ТЕХНІЧНОГО ЗАВДАННЯ

Розробляється програма, здатна перевіряти лічильник електричної енергії на коректність розрахунку енергії.

Розробляється драйвер, який виступає посередником між програмою автоматизації та установкою. З його допомогою здійснюється подача тестових параметрів на установку, яка в свою чергу під’єднана до лічильника.

На основі аналізу документації з лічильникам електронної енергії, а також ознайомлення з їх протоколом обміну буде створюватись програма, яка здатна спілкуватись з драйвером установки та виконувати тестування програми контролера лічильника електронної енергії.

2 ОСОБЛИВОСТІ СУЧАСНИХ ЛІЧИЛЬНИКІВ ЕЛЕКТРИЧНОЇ ЕНЕРГІЇ

Сучасні лічильники електричної енергії призначений для вимірювання електричної активної енергії, миттєвих значень потужності, напруги, сили струму, а також організації багатотарифного обліку в однофазних колах змінного струму в комунально–побутовій сфері та в інших галузях.

В сучасних лічильниках електричної енергії є мікроконтролер, що керує електронним дисплеєм, електричним і оптичним інтерфейсами, радіоканалом, імпульсними виходами, а також обробляє інформацію, яка надходить від кнопки «Перегляд», давачів розкривання кожуха і клемної кришки лічильників.

Мікроконтролер має вбудований годинник реального часу, стабілізований кварцовим резонатором, який відраховує роки, місяці, дні тижня, години, хвилини і секунди. Дані годинника використовуються для виконання програми тарифів, формування періодів інтеграції середньої потужності та реєстрації подій з часовою міткою. Годинник має функцію переводу часу на зимній та ліній час. Перехід часу може здійснюватися в автоматичному режимі, або по даті, яка встановлюється вручну при параметризації.

Технічні характеристики лічильника наведені в таблиці 2.1.

Таблиця 2.1 — Технічні характеристики сучасного лічильника

№	Характеристика	Значення
1	Клас точності лічильника при вимірюванні активної енергії за ГОСТ 30207 і ДСТУ IEC 61036	1,0
2	Робочий діапазон напруг, В	від 143 до 300
3	Номінальна сила струму, А	5
4	Чутливість, мА	12,5
5	Стала лічильника, імп/(кВт·год)	6400
6	Активна потужність, споживана колом напруги при, Вт	не більше 1
7	Повна потужність, споживана колом напруги при, В. А	не більше 2
8	Повна потужність, споживана колом струму при, В. А	не більше 0,2

Продовження таблиці 2.1

9	Електронний дисплей	Семисегментний з під світкою і додатковими символами
10	Діапазон температури, °С: робочий; зберігання	від мінус 40 до плюс 70 від мінус 0 до плюс 70

Для зменшення залежності похибки годинника від навколишньої температури, в лічильник вмонтований температурний давач. Лічильник при відключенні напруги мережі для забезпечення неперервності роботи вбудованого годинника має літієву батарейку живлення. При відсутності напруги мережі, мікроконтролер лічильника перемикається на економний режим, який підтримується літієвою батарейкою. В цьому режимі працює тільки внутрішній годинник лічильника. При ввімкненні напруги мережі, енергія літієвої батареї не використовується [1].

Зовнішній вигляд лічильника представлений на рисунку.2.1. На рисунку.2.1а відображений лічильник з встановленими клемною кришкою і закритою кришкою оптопорта, а на рис. 2.1б відображений лічильник без клемної кришки і відкритою кришкою оптопорта.

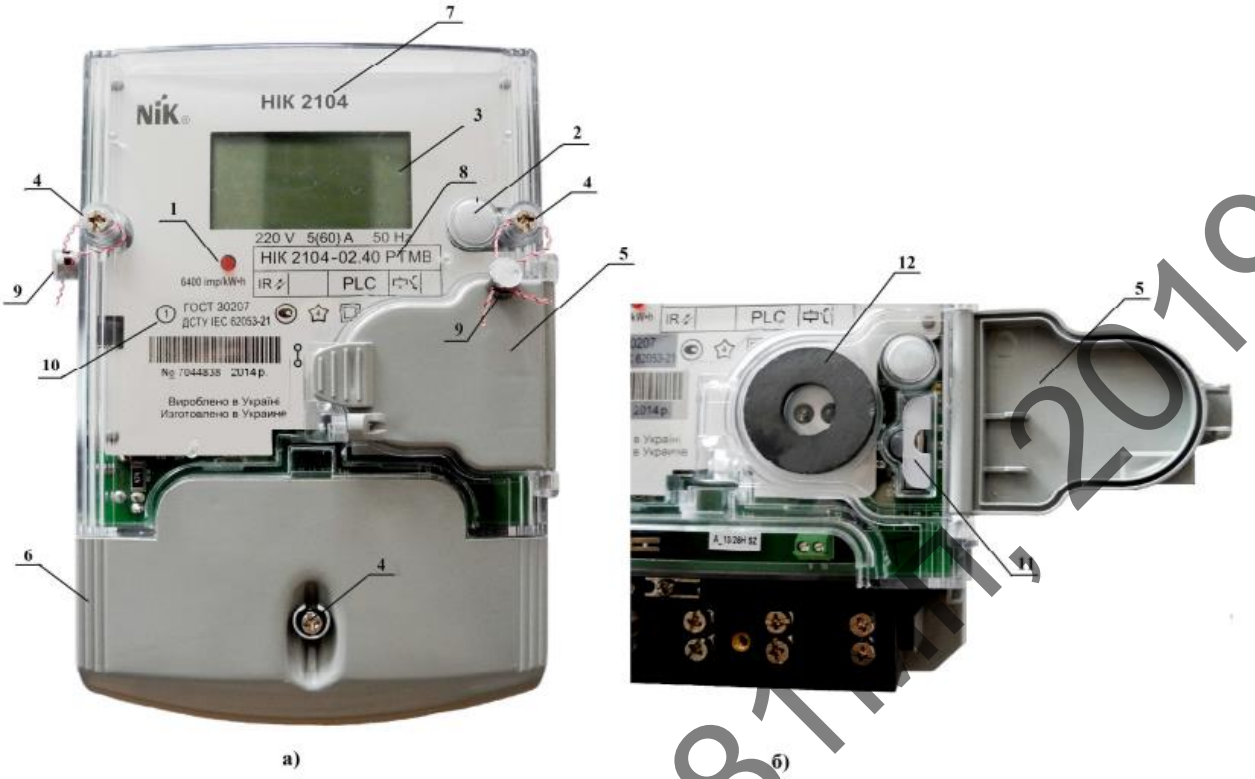


Рисунок 2.1 — Зовнішній вигляд однофазного лічильника електричної енергії

На передній панелі зображено:

1 – індикатор функціонування; 2 – кнопка «Перегляд»; 3 – електронний дисплей; 4 – пломбувальні гвинти; 5 – кришка оптопорта; 6 – клемна кришка; 7 м тип лічильника; 8 – виконання лічильника; 9 – пломба; 10 – клас точності лічильника; 11 – батарейка живлення; 12 – оптопорт.

Лічильники виконані в пластмасовому корпусі, який складається з цоколя та прозорого кожуха. В цоколь встановлюється друкована плата, а також затискна плата з затискачами і давачами струму. Затискна плата лічильників закривається кришкою затискачів. Цоколь і кожух лічильників, з'єднуються пломбувальними гвинтами. Лічильники мають давачі розкриття кожуха і кришки затискачів.

Вимірювання активної та реактивної електричної енергії проводиться шляхом аналого–цифрового перетворення електричних сигналів, що надходять від первинних перетворювачів сили струму і напруги на вхід вбудованого аналого–цифрового перетворювача(АЦП) мікроконтролера, який перетворює сигнали в послідовність цифрових відліків.

Мікроконтролер розраховує ефективні значення сили струму, напруги, потужності, і значення активної і реактивної енергії сумарно і по кожному тарифу. Мікроконтролер керує електронним дисплеєм, електричними і оптичними інтерфейсами, радіоканалом, імпульсними виходами, а також обробляє інформацію, що надходить від оптичних кнопок, давачів відкриття кожуха і клемної кришки лічильників.

Для зберігання даних в лічильниках використовується енергонезалежна пам'ять. У пам'яті зберігаються виміряні значення електроенергії і параметри лічильника. Виміряні значення енергії та параметри лічильників, при відсутності напруги на затискачах напруги лічильників, повинні зберігається не менше 10 років. В лічильниках застосований семисегментний електронний дисплей з додатковими символами який відображений на рисунку 2.2.

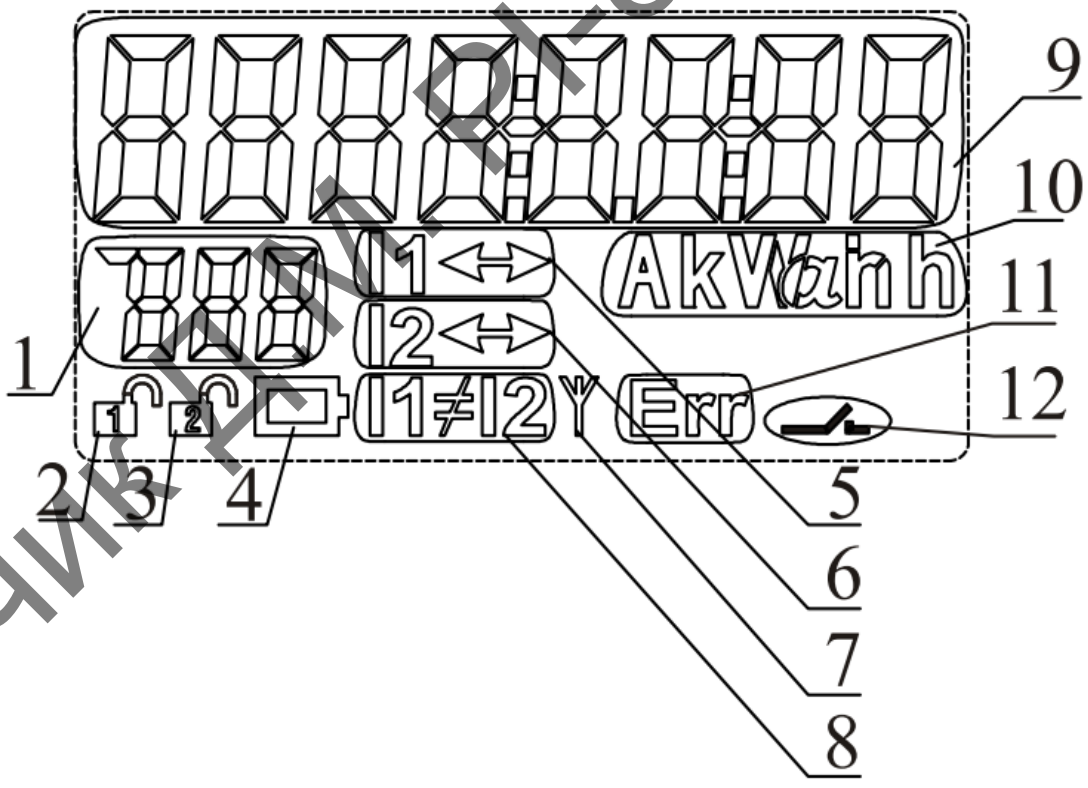


Рисунок 2.2 — Зовнішній вигляд електронного дисплея

Символи, які зображені на електронному дисплеї рисунку мають таке призначення:

1 – додаткові символи;

2 – якщо символ блимає, розкритий кожух лічильника;

3 – якщо символ блимає, розкрита кришка затискачів лічильника;

4 – якщо символ блимає, батарейка живлення годин вимагає заміни;

5 – якщо символ світиться, в першому вимірювальному елементі зворотний напрямок струму;

6 – якщо символ світиться, в другому вимірювальному елементі зворотний напрямок струму;

7 – якщо символ світиться, йде сеанс зв'язку з лічильником і зовнішніми пристроями;

8 – якщо символ світиться, то сила струму в першому і другому вимірювальних елементах не однакова;

9 – параметр, який відображається;

10 – одиниці виміру параметру, який відображається: « » сила струму в Амперах; « » напруга в Вольтах; « » активна потужність в кіловатах; « » активна енергія в кіловат-годинах;

11 – зафіксована внутрішня помилка лічильника;

12 – якщо символ світиться, увімкнене навантаження у споживача (відключено реле керування навантаженням).

Для живлення лічильників використовується імпульсне джерело живлення, що перетворить випрямлену вхідну напругу, в напругу необхідну для живлення всіх вузлів і модулів лічильників [1].

3 ОПИС СТРАТЕГІЙ РОЗРОБКИ ПРОГРАМ ТЕСТУВАННЯ

Розробка програмного забезпечення — це складний, поетапний процес, який потребує плану, організації та ресурсів. Його розробка повинна контролюватись належним чином. Тому, протягом всієї епохи комп'ютерів створювались стратегії–методології з контролю розробки програмного середовища.

Усі продукти всіх процесів програмної інженерії являють собою певні описи — тексти вимог до розробки, погодження домовленостей, документацію, тексти програм, інструкції з експлуатації тощо.

Головними ресурсами програмної інженерії, які визначають ефективність її розробок, є час і вартість цих розробок [2]. І в залежності від вибраної моделі розробки програмних систем буде залежати швидкість та якість розробки лічильників електричної енергії. Кожна модель має свій підхід до тестування програмного забезпечення контролера лічильника, оскільки цей етап відіграє велику роль у стратегіях розробки програм тестування.

Розробка програмного забезпечення передбачає застосування стратегії з розробки продукту тестування. Існує декілька моделей з розробки програмних систем:

- водоспадна;
- v – подібна;
- ітераційна;
- спіральна;
- гнучка.

В залежності від складності проекту, ресурсів компанії, сфери діяльності вибирається зручна модель розробки ПО.

3.1.1 Водоспадна модель

Водоспадна модель — являє собою одну з перших моделей з розробки ПО. ЇЇ суть полягала в ретельному підході роботи до кожного з етапів розро-

бки. Якщо один з етапів не був завершений то робота наступного етапу не починалась. Структура даної моделі відображена на рисунку 3.1.

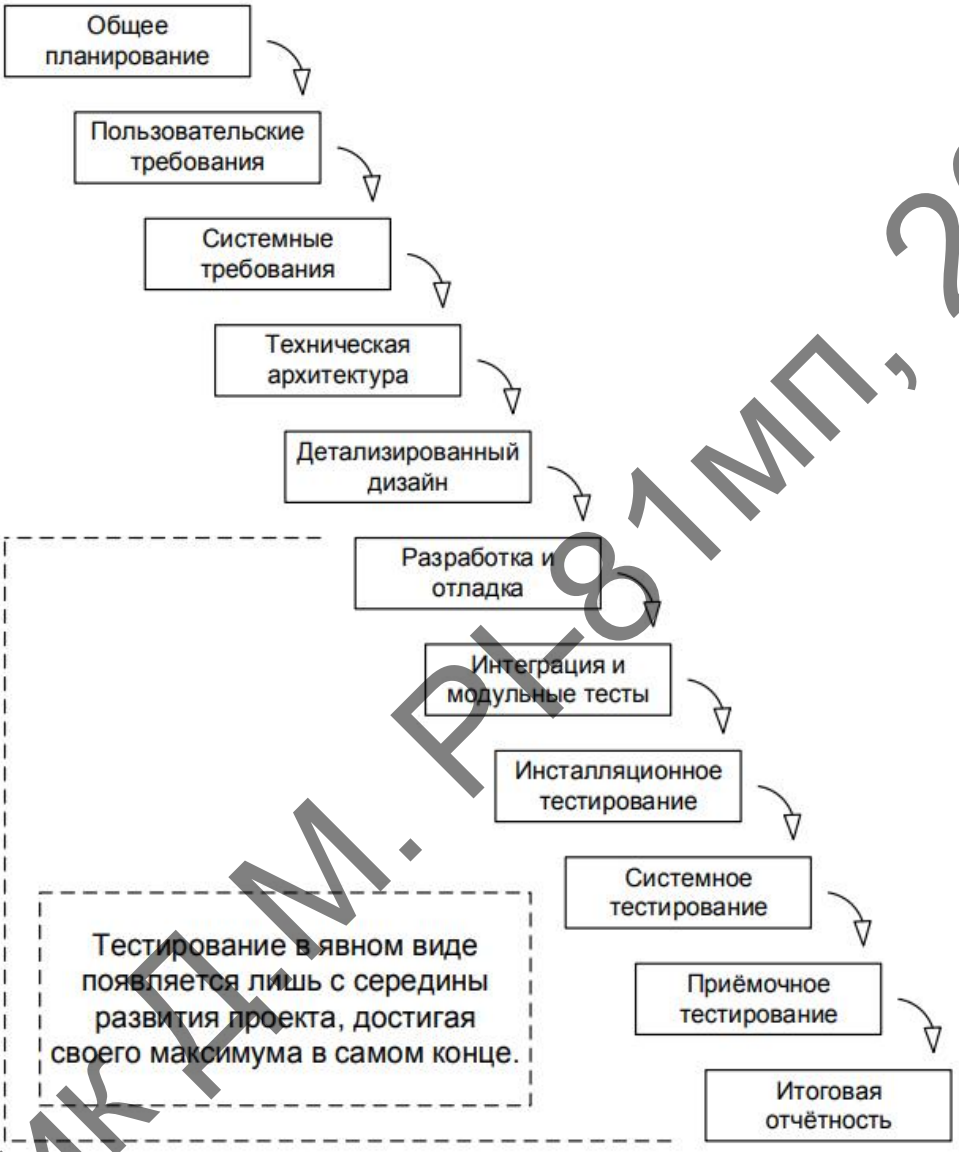


Рисунок 3.1 — Водоспадна модель розробки ПО

Водоспадна модель отримала позитивний пріоритет з боку повноти формулювання завдання, документації, а також критеріїв.

3.1.2 V–подібна модель

V–подібна модель схожа на водоспадну модель, аналізуючи рисунку 3.2 то можна помітити на схожість у структурах. Але відмінність полягає в тому що інформація в процесі розробки вже реалізовується. Тобто кожний етап життєвого циклу програмного забепечення пов’язаний з наступним.

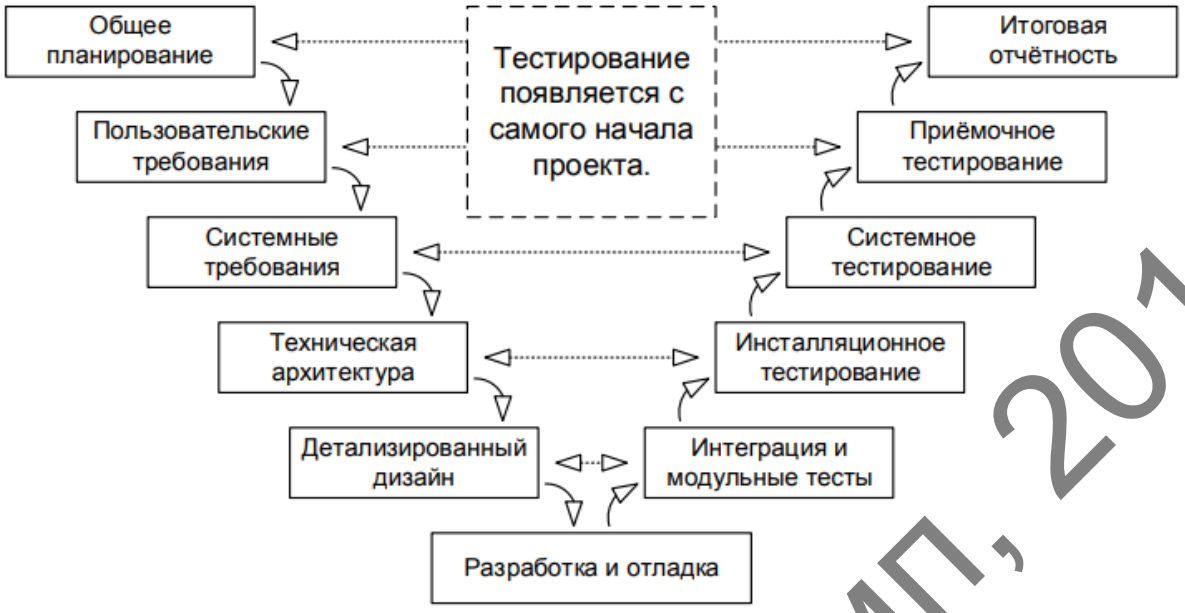


Рисунок 3.2 — V – подібна модель розробки ПО

При використанні V–подібної моделі на кожній стадії «на спуску» потрібно думати про те, що і як буде відбуватися на відповідній стадії "на підйомі". Тестування тут з'являється вже на самих ранніх стадіях розвитку проекту, що дозволяє мінімізувати ризики, а також виявити і усунути безліч потенційних проблем до того, як вони стануть проблемами реальними [3].

3.1.3 Ітераційна модель

Ітераційна модель непослідовна порівняно з вище вказаними. Життєвий цикл розробки програмного забезпечення має інший підхід. Дана стратегія циклічна і кожний етап являє собою цикл. В кожній з ітерацій виконується окремий компонент системи, після чого компонент добавляється до попереднього розробленого компоненту. Структура ітераційної моделі відображена на рисунку 3.3.

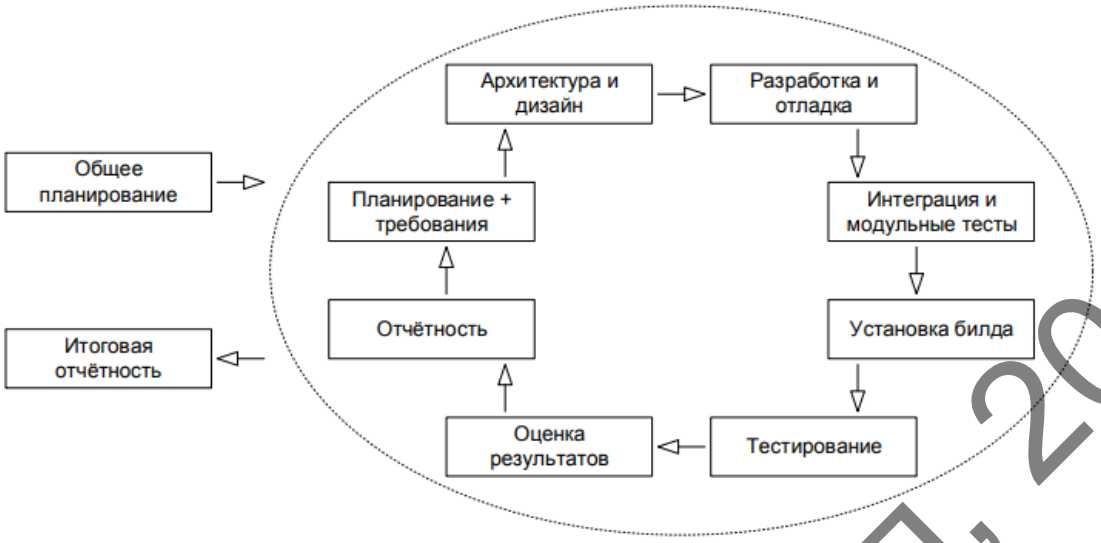


Рисунок 3.3 — Ітераційна модель розробки ПО

Довжина ітерацій може змінюватися в залежності від багатьох факторів, однак сам принцип багаторазового повторення дозволяє гарантувати, що і тестування, і демонстрація продукту кінцевому замовнику (з отриманням зворотного зв'язку) буде активно застосовуватися з самого початку і протягом усього часу розробки проекту.

У багатьох випадках допускається розпаралелювання окремих стадій всередині ітерації і активна доробка з метою усунення недоліків, виявлених на попередніх стадій.

Ітераційна інкрементальна модель дуже добре зарекомендувала себе на об'ємних і складних проектах, які виконуються великими командами протягом тривалих термінів. Однак до основних недоліків цієї моделі часто відносять високі накладні витрати [3].

3.1.4 Спіральна модель

Спіральна модель являє собою як ітеративна, так і поетапна модель. Суть її в тому, що весь процес створення кінцевого продукту представлений у вигляді умовної площини, розбитою на чотири сектори, кожен з яких представляє окремі етапи його розробки: визначення цілей, оцінка ризиків, розробка і тестування, планування нової ітерації. Детальна інформація по спіральній моделі зображена на рисунку 3.4.

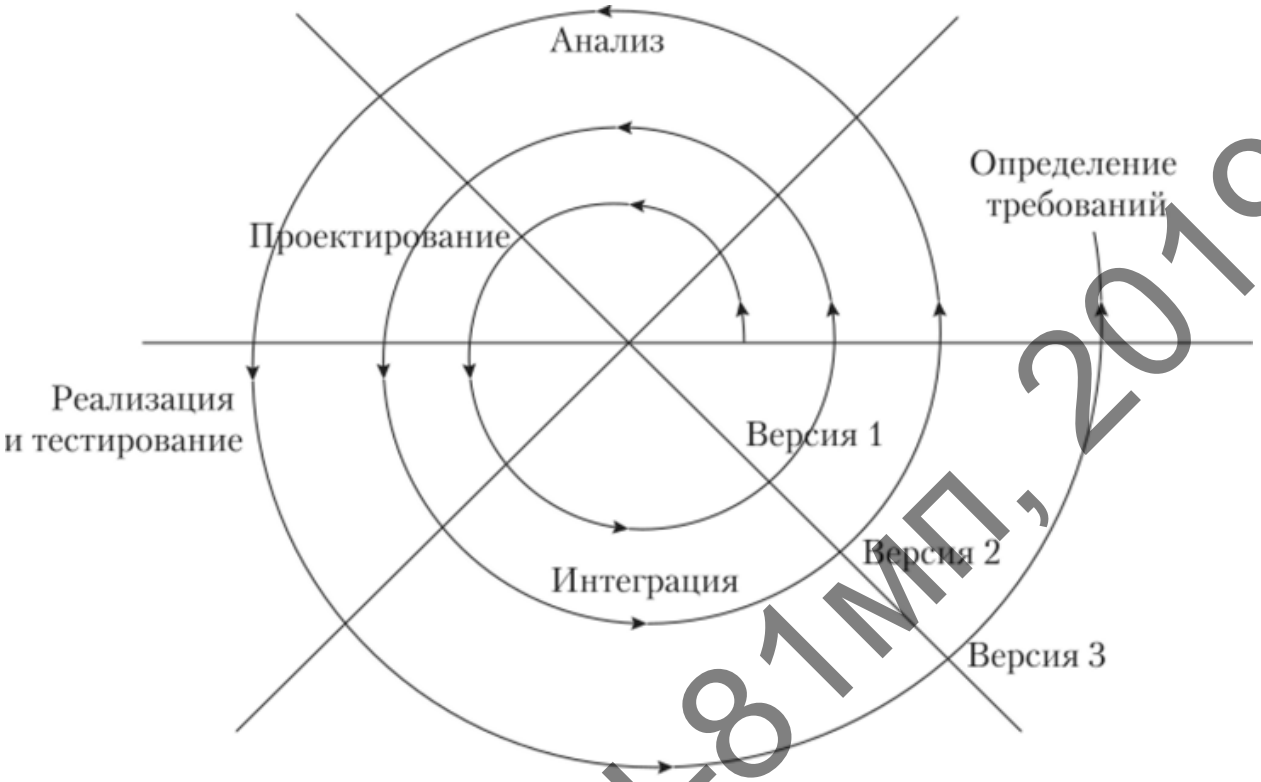


Рисунок 3.4 — Спіральна модель розробки ПО

У спіральної моделі життєвий шлях продукту, що розробляється зображується у вигляді спіралі, яка, розпочавшись на етапі планування, розкручується з проходженням кожного наступного кроку. Таким чином, на виході з чергового витка ми повинні отримати готовий протестований прототип, що задовольняє всім вимогам готовий до релізу [4].

3.1.5 Гнучка модель

Гнучка модель має сукупність підходів до розробки програмного забезпечення і базується на документі що описує основні принципи цієї моделі. Цей документ має назву *«Agile Manifesto»*.

Основні принципи *«Agile Manifesto»*:

- люди та їх взаємодія важливіша ніж процеси та інструменти;
- працюючий продукт важливіший вичерпаної документації;
- співпраця з клієнтом важливіша ніж узгодження умов контракту;
- готовність до змін важливіше проходження попереднім планом.

З точки зору тестування і управління якістю підвищену увагу до ризиків є відчутною перевагою при використанні спіральної моделі для розробки концептуальних проектів, в яких вимоги природним чином є складними і нестабільними (можуть багаторазово змінюватися по ходу виконання проекту).

Структура гнучкої моделі розробки програмного забезпечення зображена на рисунку 3.5.

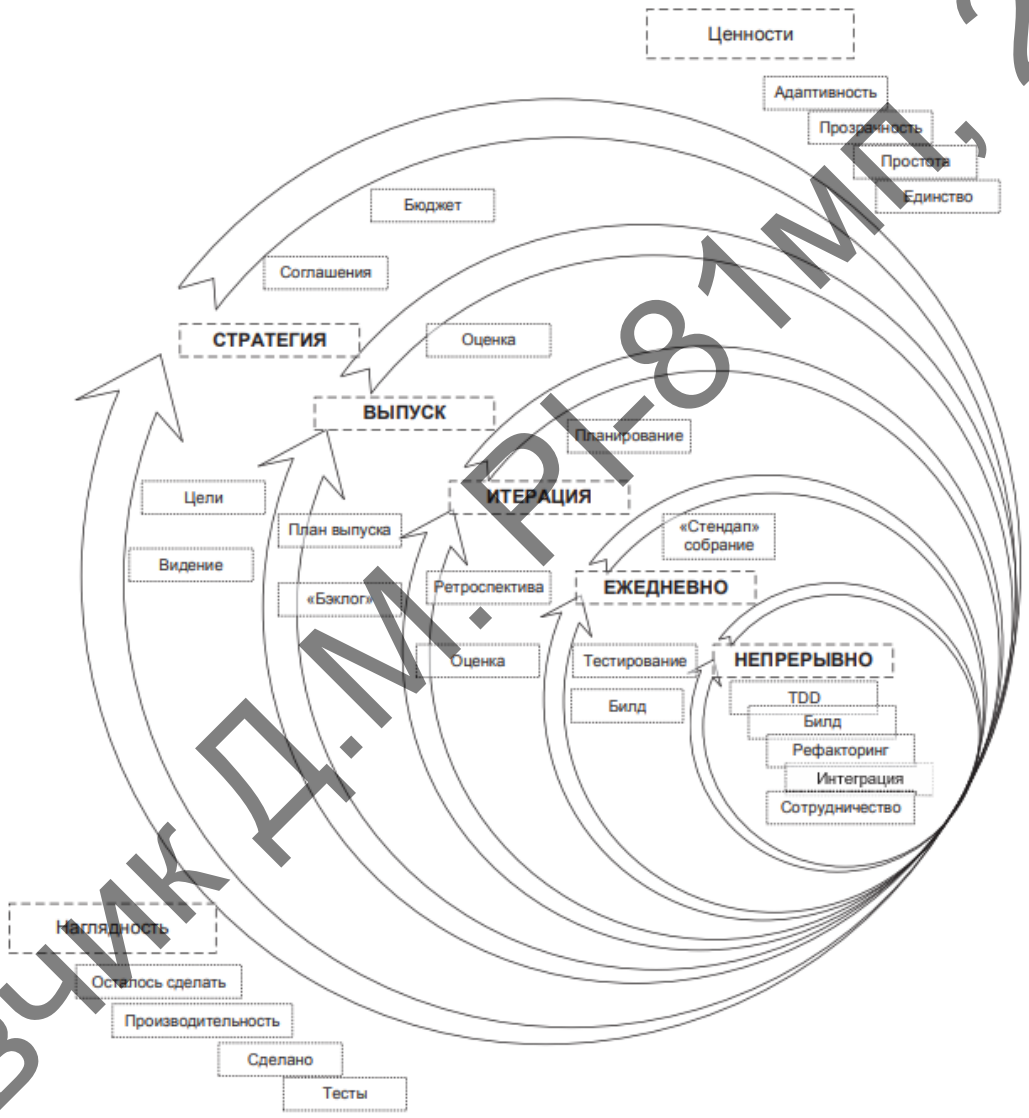


Рисунок 3.5 — Гнучка модель життєвого циклу розробки ПО

Основна особливість такої стратегії ставити в найвищому пріоритеті відносини між клієнтом. Цим може займатись менеджер проекту або бізнес – аналітик.

Дана модель дає змогу швидко показати що розробка продукту йде не в правильному руслі.

Питання тестування слід піднімати від початку розробки продукту. Адже тестування являє собою відслідковування помилок в програмних системах і усунення їх до використання. Незалежно від етапу, починаючи від технічного завдання і закінчуючи повноцінним працюючим програмним забезпеченням, тестуванні слід йти паралельно з кожним етапом.

Водоспадна модель найбільш придатна стратегія для даної роботи. Оскільки проект розробляється поетапно, тому не закінчивши теперішній етап, на розробку наступного не перейдемо.

4 ВИДИ ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Тестування пронизує весь життєвий цикл програмного забезпечення, починаючи від проектування і закінчуючи невизначено довгим етапом експлуатації. Ці роботи безпосередньо пов'язані із завданнями управління вимогами та змінами, адже метою тестування є якраз можливість переконатися у відповідності програм заявленим вимогам.

Слід розділити перевірку працездатності програм в ході безпосереднього написання коду самим програмістом і після завершення основного етапу кодування (швидше за все, спеціальними тестувальниками).

Тестування — процес також ітераційний та поетапний. Після виявлення і виправлення кожної помилки обов'язково слід повторення тестів, щоб переконатися в працездатності програми. Більш того, для ідентифікації причини виявленої проблеми може знадобитися проведення спеціального додаткового тестування [5].

4.1.1 Ручне тестування

Ручне тестування — це частина процесу тестування на етапі контролю якості в процесі розробки програмного забезпечення. Воно проводиться тестувальником без використання програмних засобів, для перевірки програми або сайту шляхом моделювання дій користувача. У ролі тестувальників можуть виступати і звичайні користувачі, повідомляючи розробникам про знайдені помилки [6].

Програмне забезпечення, що тестується вручну інженером тестувальником, беруть на себе роль користувачів підпорядковуючись певному сценарію. Після виконання даного сценарію фіксують результат. Далі проходить аналіз результатів на основі якого проводять перевірку фактичних результатів з отриманими. Задача ручного тестування програмного забезпечення — виявити поведінку яка буде відрізнятися від очікуваного користувачем.

Завдяки ручному тестуванню можна знизити ризик помилки, а також можна швидко змінити сценарій у разі необхідності. Крім того проект ручного тестуванню можна швидко запустити.

Основні етапи ручного тестування:

1) підготовчий:

– аналіз технічного завдання програмного продукту;

– розробка сценарію тестування (створення задач);

– оцінка граней проекту.

2) основний:

– виконання задач по сценарію;

– фіксування помилок.

3) заключний:

– створення звіту на основі проведених тестів:

– поради по програмному продукту з метою покращення.

4.1.2 Автоматизоване тестування

Автоматизоване тестування — це процес тестування або його частина яке проводиться з для покращення якості програмного забезпечення за допомогою додаткових програмних засобів. Дане тестування особливо має значення коли виникають задачі тестування, що потребують багаторазового виконання.

Існує три типи тестування, які можна автоматизувати:

– функціональне (модульне тестування);

– регресійне (тестування старого функціоналу і перевірка на наявність помилок в новому функціоналі);

– навантажувальне (тестування функціоналу під час навантаження).

Даний вид тестування дає можливість виконання тестів, які не можуть бути виконані вручну або ж вони потребують великих затрат. Результати тестування автоматично зберігаються та розсилаються на поштову скриньку [7].

Таблиця 4.1 — Порівняльна характеристика автоматизованого та ручного тестування

№	Критерії	Ручне тестування	Автоматизоване тестування
1	Задання вхідних параметрів	Гнучкість в завданні даних. Дозволяє використовувати різні значення на різних циклах прогону тестів, розширюючи покриття	Вхідні дані строго задані
2	Перевірка результатів	Гнучка, дозволяє тестувальника оцінювати нечітко сформульовані критерії	Сувора. Нечітко сформульовані критерії можуть бути перевірені тільки шляхом порівняння з еталоном
3	Повтореність	Низька	Висока
4	Надійність	Низька	Висока
5	Чутливість до змін продукту	Залежить від складності задачі	Висока
6	Швидкість виконання	Низька	Висока
7	Можливість генерації тестів	Відсутня	Підтримується

Порівняння показує тенденцію сучасного тестування лічильників електричної енергії. Автоматизація тестування дає можливість працювати з великими масивами даних і видавати точні результати. На відмінну від ручного, авто–тести виконуються швидко і можливість повторення не габаритна по ресурсам.

5 ОСОБЛИВОСТІ АВТОМАТИЗОВАНОГО ТЕСТУВАННЯ СУЧАСНИХ ЛІЧИЛЬНИКІВ

З допомогою сучасних інструментів розробки та протоколів обміну даними автоматизовують тестування лічильників електричної енергії. Нижче представлена структурна схема, в якій описується робота автоматизованого тестування.

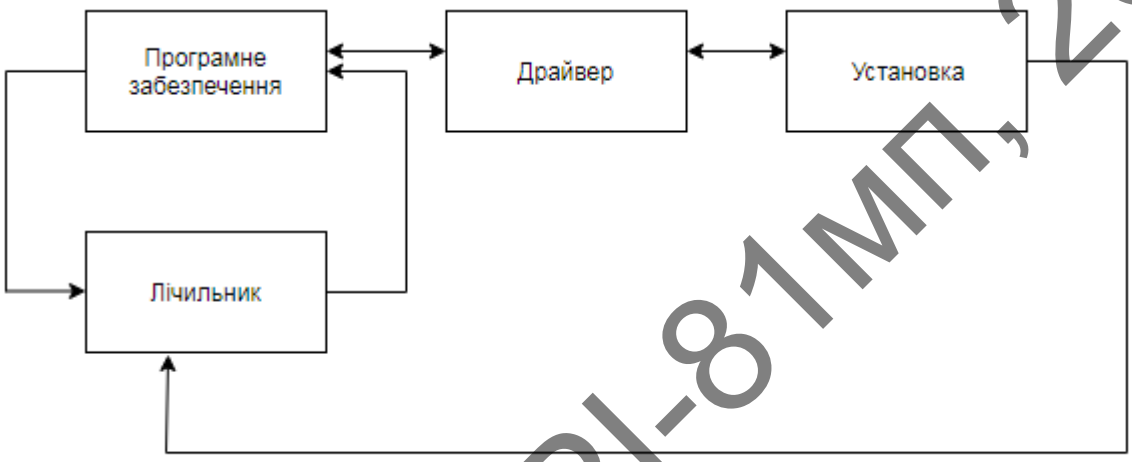


Рисунок 5.1 —Структурна схема роботи автоматизованого тестування

З допомогою драйверу, що ґрунтується на використанні протоколу RS–232, здійснюється керування установкою, яка в свою чергу під'єднана до лічильника.

RS–232 — це послідовний інтерфейс для передачі даних між двома пристроями.

Задаючи вхідні параметри в програмі вищого рівня керуємо установкою, паралельно спілкуючись з лічильником з допомогою оптопорта.

Оптопорт — перетворювач USB–оптопорт призначений для організації безконтактного сеансу зв'язку між електролічильників с інфрачервоним портом і персональним комп'ютером з інтерфейсом USB. У лічильника і перетворювача є магнітне кріплення для стійкого позиціонування і обміну даними [8].

5.1 Вибір калібрувальної установки

На ринку калібрувальних установок є достатня кількість пристроїв, які б задовольняли потреби завдання. Установки такого типу мають багато метрологічних функцій і одні з найважливіших це: повірка, калібровка, тестування.

Найбільш поширеними калібрувальними установками для сучасних лічильників є: Fluke 6003A, PTS 400.3, CalmetC300B.

5.1.1 Установка Fluke 6003A

Калібрувальна установка Fluke 6003A призначена для генерації постійного та змінного струму, а також постійної і змінної напруги. Також є можливість задання кута зсуву по кожній фазі. Зовнішній вигляд даної установки зображений на рисунку 5.1.

Рисунок 5.1 — Зовнішній вигляд Fluke 6003A

Основні функції:

- імітує електричну потужність та енергію постійного або змінного струму в діапазоні напруг до 600 В змінного струму або 280 В постійного струму і діапазоні струму до 30 А по фазі або 90 А комбіновано;– три фази живлення в одному приладі;
- рентабельний;
- простий у використанні;
- зсув фази між каналами напруги і струму може бути встановлений від 0 ° до 359,99 °;
- характеристики ± 375 мільйонних часток для потужності і 0,01° для фази;
- струмові виходи можуть бути ізольовані від заземлення до піку 450В;
- додаткові можливості контролю якості енергії і систем електропостачання;
- вбудований мультиметр постійного струму для вимірювань виходу перетворювача;
- пропонує адаптер сильного струму для задач, де використовується струм від 30 до 90 А.

Установка Fluke 6003A має графічний інтерфейс з допомогою якого можна керувати параметрами [9].

5.1.2 Установка PTS 400.3

Калібрувальна установка PTS 400.3 — це трифазна повністю автоматична калібрувальна система класу точності 0,02. До складу установки входить зразковий еталон, джерело потужності і панель управління. Прилад призначений для всебічного дослідження і тестування всіх параметрів вимірювальних приладів і осцилографів. Зовнішній вигляд даної установки показаний на рисунку 5.2.

Рисунок 5.2 — Зовнішній вигляд PTS 400.3

Основні функції:

- паралельне вимір до 3–х приладів або 3–х регістрівкомбінованих вимірювальних приладів;
- діапазони струму / напруги / частоти: 1 мА ... 120 А / 30 В ... 520 В / 45 Гц ... 70 Гц;
- шість роз'ємів для підключення струму та калібрування вимірювальних приладів електроенергії прямого і трансформаторного включення;
- можливість використання різних струмових кліщів (до 100 А або до 3000 А), а також вимірювальних штанг для первинної сторони по напрузі до 40 кВ;
- незалежна пам'ять для збереження результатів і даних про місце вимірювання;
- інтерфейс RS–232C для передачі даних і управління за допомогою зовнішнього комп'ютера;

- модуль управління PCS 400.3 для дистанційного керування приладом за допомогою протоколу bluetooth;
- трифазний високоточний джерело, який використовує однофазну мережу для живлення;
- струм і напруга можуть задаватися окремо;
- струм, напруга і зсув фази задаються з високою точністю за допомогою клавіатури;
- встановлювані значення стабілізуються за допомогою цифрового і аналогового модулів контролю.

Прилад відрізняється широким діапазоном вимірювання, високою точністю і високим ступенем захисту від впливу зовнішніх полів [10].

5.1.3 Установка Calmet C300B

Калібрувальна установка C300B призначена для регулювання, перевірки та верифікації вимірювальних приладів, що застосовуються в енергетиці: реле електролічильників, частоти, напруги та струму, трансформатори струму та затискачі, лічильники активної та реактивної потужності, лічильники фаз, частотні вимірювачі, амперметри, вольтметри , перетворювачі, системи моніторингу та аналізатори якості електроенергії в однофазних та трифазних з'єднаннях.

C300B має високу точність і одночасно велику вихідну потужність при порівняно невеликих розмірах і вазі. Тестер має можливість автоматично отримувати характеристики помилок випробуваного лічильника.

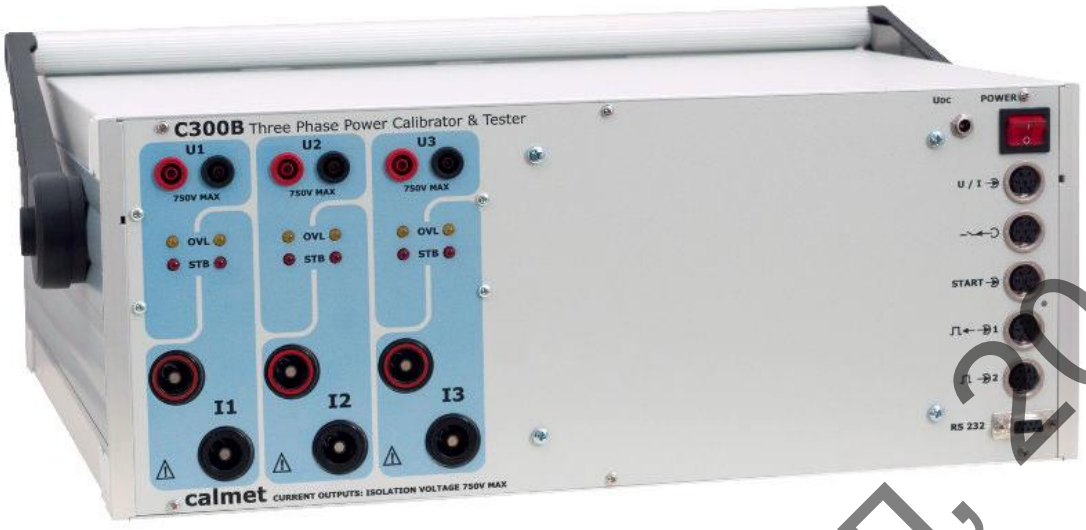
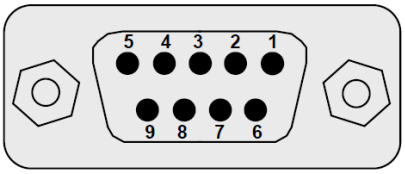


Рисунок 5.3 — Зовнішній вигляд калібрувальної установки Calmet C300B

Установка C300B є трифазним джерелом змінного струму і напруги, що дозволяє встановлювати:

- напругу в діапазоні від 0.5В до 560В, в чотирьох під–діапазонах: 70–140–280–560В,
- струм в діапазоні від 0.001А до 120А в чотирьох під– діапазонах: 0.5–6–20–120А,
- частоту в діапазоні від 40Гц до 500Гц з можливістю синхронізації з частотою живлення мережі
- кут між фазами в діапазоні 0 до 360 градусів, а також є можливість встановлювати косинус і синус, але ця функція можлива тільки в програмному середовищі виробника. Контролер установки не робить математичні перетворення,
- активну, реактивну, повну потужність.

Керування установкою здійснюється з допомогою інтерфейсу RS–232. Детальна інформація підключення по інтерфейсу зображена на рисунку 5.4

RS 232	socket DB-9 of the RS232 interface for PC computer connection	

Pin	Signal - description
1	NC – not connected
2	TX PC
3	RX PC
4, 6	NC – not connected
5	GND - ground
7	CTS PC
8	RTS PC
9	+5V / 100mA – auxiliary supply +5V / max 100mA

Рисунок 5.4. — Підключення по стандарту RS232 до установки

Серед перелічених вище калібрувальних установок було вибрано Calmet C300B. Оскільки його функціонал схожий на інші, але вартість попередніх на порядок вища [11].

6 ІНТЕРФЕЙС ПРОГРАМИ ТЕСТУВАННЯ

Для керування установкою Calmet C300B розроблено драйвер, який ґрунтується на протоколі обміну установки.

6.1 Алгоритм зв’язку контролера установки з комп’ютером

Зв'язок з калібратором здійснюється за схемою *Master/Slave*. ПК *«Master»* може спілкуватися з калібратором *«Slave»* через набір команд *ASCII*. Усі команди повинні бути написані символами верхнього регістру. Кожна команда закінчується послідовністю завершення рядків з двох символів *ASCII* з десятковими кодами *«13 [CR]»* та *«10 [LF]»*. На рисунку 6.1 показана схема обміну даними з установкою.

Рисунок 6.1 — Комунікація з установкою

Структура команди повинна бути наступна. Існує три типи відповіді, які приходять з установки:

- *«OK[CR][LF]»,* це значить що контролер установки проаналізував відіслану користувачем;
- *«ER[CR][LF]»*, команда не вірна або не вірні параметри передачі даних;
- *«<ANSWER_PARAM1><ANSWER_PARAMx>[CR][LF]»,* відповіді даного типу приходять коли користувач хоче дізнатись теперішні налаштування довільного параметру [13].

Структурна схема зв’язку з контролером установки Calmet C300B показа на рисунку 6.2.

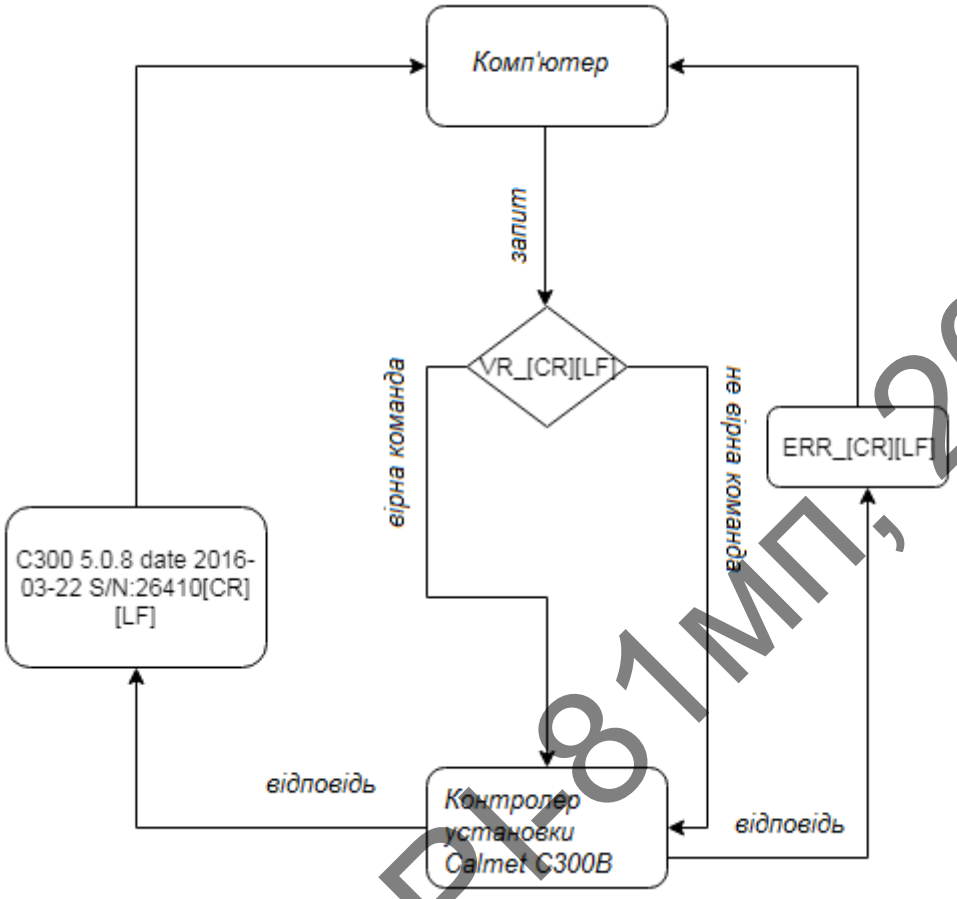


Рисунок 6.2 — Структурна схема зв’язку з контролером установки Calmet C300B

З комп’ютера посилається команда на встановлення контакту з установкою. На команду *«VR_[CR][LF]»* контролер установки відповідає інформацією про самого себе, тобто серійний номер і дату виготовлення.

Для спілкування з лічильником потрібен оптопорт, з допомогою якого буде відбуватись побітовий обмін даними з лічильником.

6.2 Опис драйверу для керування установкою

Драйвер реалізований мовою Python, оскільки вона може працювати байтами, а також має необхідний пакет вбудованих модулів для роботи з декількома COM–портами одночасно. Повний код драйверу описаний в Додатку Б.

Драйвер складається з таких основних підпрограм–функцій:

1) ініціалізації об’єкту драйверу;

2) затримки програми;
3) очікування відповіді від установки;
4) авторизації;
5) отримання теперішніх параметрів струму та напруги;
6) функції встановлення струму та напруги
7) функція встановлення куту зсуву;
8) функції переключення вимірювального трансформатора;
9) функції вмикання та вимикання установки.

Функція ініціалізації об’єкту драйвера:

```
EXPECTED_INIT_RESPONSE = b'C300 5.0.8 date 2016-03-22 S/N: 26410\r\n'
class Calmet:
    def __init__(self, driver):
        self.driver = driver # атрибут об’єкта драйвер якому присвоюється драйвер Serial.serial(), що призначений для ініціалізації порта по якому спілкується комп’ютер з контролером установки
        self.autorization()
```

Ініціалізація об’єкта класу Calmet, якщо при виконання програми об’єкт не створився тоді програма завершується з помилкою.

Функція затримки:

```
def wait(self, seconds): # функція затримки
    return time.sleep(seconds)
```

Функція очікування відповіді від установки:

```
def get_response(self): # функція отримання відповіді від контролера
    self.wait(1)
    res = self.driver.readline()
    return res
```

Функція отримання відповіді *get_response()* чекає поки контролер установки сформує пакети щоб віддати їх комп'ютеру.

Функція авторизації:

```
def autorization(self): # функція авторизації
    self.driver.write('VR_\r\n'.encode('ascii'))
    if self.get_response() != EXPECTED_INIT_RESPONSE:
        self.driver.close()
        raise CalmetExeption("Error during calmet init")
```

Якщо при запиті на команду *«VR_[CR][LF]»* калібратор не відповідає власним серійним номером тоді програма завершується з помилкою.

Функція отримання теперішніх параметрів струму та напруги:

```
def read_voltage_and_current_info(self):
    self.driver.write('ENDAMP_\r\n'.encode('ascii'))
    self.get_response()
```

З допомогою *read_voltage_and_current_info()* можна дізнатись які струми і напруги подає установка на лічильник, а також по яким фазам.

Функція встановлення напруги:

```
def set_voltage(self, default1=0.5000, default2=0.5000, default3=0.5000):
        self.driver.write(('U_{},{},{}\r\n'.format(default1, default2,
        default3)).encode('ascii'))
        self.get_response()
        self.wait(3)
```

set_voltage() — дає можливість встановити напругу по кожній фазі. Якщо по першій фазі то аргумент *default1* має бути заповнений.

Функція встановлення струму:

```
def set_current(self, default1=0.001000, default2=0.001000, default3=0.001000):
        self.set_current_range(default1, default2, default3)
        self.driver.write(('I_{},{},{}\r\n'.format(default1,default2,
        default3)).encode('ascii'))
```

```
        self.get_response()
        self.wait(3))
```

set_ current () — дає можливість встановити струм по кожній фазі. Якщо по першій фазі то аргумент *default1* має бути заповнений.

Функція вмикання та вимикання установки:

```
def set_stendby_or_operate(self, phase1_u=1, phase2_u=1, phase3_u=1, phase1_i=1, phase2_i=1, phase3_i=1):
    """ STB_1,1,1,1,1,1 if set 1 = OFF phase, if set 0  = ON PHASE """
        self.driver.write(
        'STB_{},{},{},{},{},{}\r\n'.format(phase1_u, phase2_u, phase3_u,
        phase1_i, phase2_i, phase3_i).encode(
        'ascii'))
         self.wait(3)'
```

З допомогою функції *set_stendby_or_operate()* відбувається подача параметрів на лічильник. Тобто дана команда інформує контролер про канали які повинні відкритись. Якщо була подана команда *«STB_0,1,1,0,1,1\r\n»* тоді на лічильник піде напруга і струм тільки по першій фазі.

Функція встановлення куту між фазами:

```
  def set_angle(self, u1i1=0, u2i2=0, u3i3=0, u1u2=120, u1u3=-120):
        minimal angle  = -360, maximum value of angle = 360"""
        self.driver.write
        (('FA_{},{},{},{},{}\r\n'.format(u1i1,u2i2,u3i3,u1u2,u1u3)).encode('as
ciі'))
        self.get_response()
        self.wait(3)
```

set_angle() — дозволяє встановити кут зсуву фази по кожній з фаз лічильника. А також між першою і другою, а також між першою і третьою фазою завжди буде 120 градусів.

В установки Calmet C300B є чотири трансформатори. І в залежності від того яку напругу чи струм ми подаємо замикається реле одного з чотирьох трансформаторів.

Діапазони напруг:

1) від 0.5 В до 70 В;

2) від 1 В до 140 В;

3) від 2 В до 280 В;

4) від 5 В до 560 В.

Діапазони струмів:

1) від 0.005 А до 0.5 А;

2) від 0.5 А до 6 А;

3) від 0.5 А до 20 А;

4) від 1 А до 120 А.

Реалізовано функцію здатну аналізувати значення струму чи напруги введене користувачем. Тобто в залежності від величини параметра включався потрібний трансформатор.

Функція вставлення вимірювального трансформатора в залежності від вхідної напруги:

```
def_set_voltage_range(self,voltage_by_one_phase=0.0,voltage_by_second_phase=0.0, voltage_by_third_phase=0.0):
    try:
        range_voltage = []
        currents = [voltage_by_one_phase, voltage_by_second_phase, voltage_by_third_phase]
        for i in currents:
            if 0.5 <= i <= 70:
                range_voltage.append(1)
            elif 1 <= i <= 140:
                range_voltage.append(2)
            elif 2 <= i <= 280:
```

```
                range_voltage.append(3)
            elif 5 <= i <= 560:
            range_voltage.append(4)
            else:
                range_voltage.append(1)
            self.driver.write('RU_{},{},{}\r\n'.format
        (*range_voltage).encode('ascii'))
            self.get_response()
        except Exception as ex:
            print("Not correct set voltage range")
              raise CalmetException(ex)
```

Реалізована вище функція проводить аналіз вхідного параметра напруги і встановлює трансформатор який підходить.

Функція вставлення вимірювального трансформатора взалежності від вхідної струму:

```
def set_current_range(self, current1=0.0, current2=0.0, current3=0.0):
    try:
        range_current = []
        currents = [current1, current2, current3]
        for i in currents:
            if 0.005000 <= i <= 0.5:
                range_current.append(1)
            elif 0.05 <= i <= 6:
                range_current.append(2)
            elif 0.2 <= i <= 20:
                range_current.append(3)
            elif 1 <= i <= 120:
                range_current.append(4)
            else:
                range_current.append(1)
```

```
            self.driver.write('RI_{},{},{}\r\n'.format(*range_current).encode(
        'ascii'))
            self.get_response()
        except Exception as ex:
            print("Not correct set current range")
            raise CalmetException(ex)
```

Функція *set_current_range()* робить аналіз вхідного параметра струму і подає запит на включення трансформатора який підходить по діапазону.

6.3 Особливості автоматизованих тестів

Програма автоматизованого тестування лічильників електричної енергії складається з наступних модулів: configs, Logs, Test_Logs, __calculations, __logger_maker, __send_by_mail, check_registers_of_energy.

Вхідні параметри програми тестування зберігаються в файлі *configs.json* папки *configs.*

Папка *Logs* міститься інформація про помилки під час спілкування з лічильником.

Test_Logs накопичує результати тестування.

__calculations.py — має два класи: *WorkWithJson*, *Computation*.

Клас *WorkWithJson* працює з файлом *configs.json*, а в класі *Computation* знаходяться функції по розрахунку усіх видів енергії.

__logger_maker.py — модуль з допомогою якого відбувається запис та зберігання всіх результатів тестування.

__send_by_mail.py — відправляє результати тестування на електронну пошту.

check_registers_of_enegry.py — сценарій який містить тести.

В процесі виконання програми тестування, перелічені вище модулі починають взаємодіти між собою.

6.3.1 Опис вторинних підпрограм для запуску автоматизованих тестів

Програма автоматизованого тестування лічильників електричної енергії дозволяє проводити тестування як однофазних, так і трьох фазних лічильників. Алгоритм програми автоматизованого тестування зображений на рисунку 6.3.

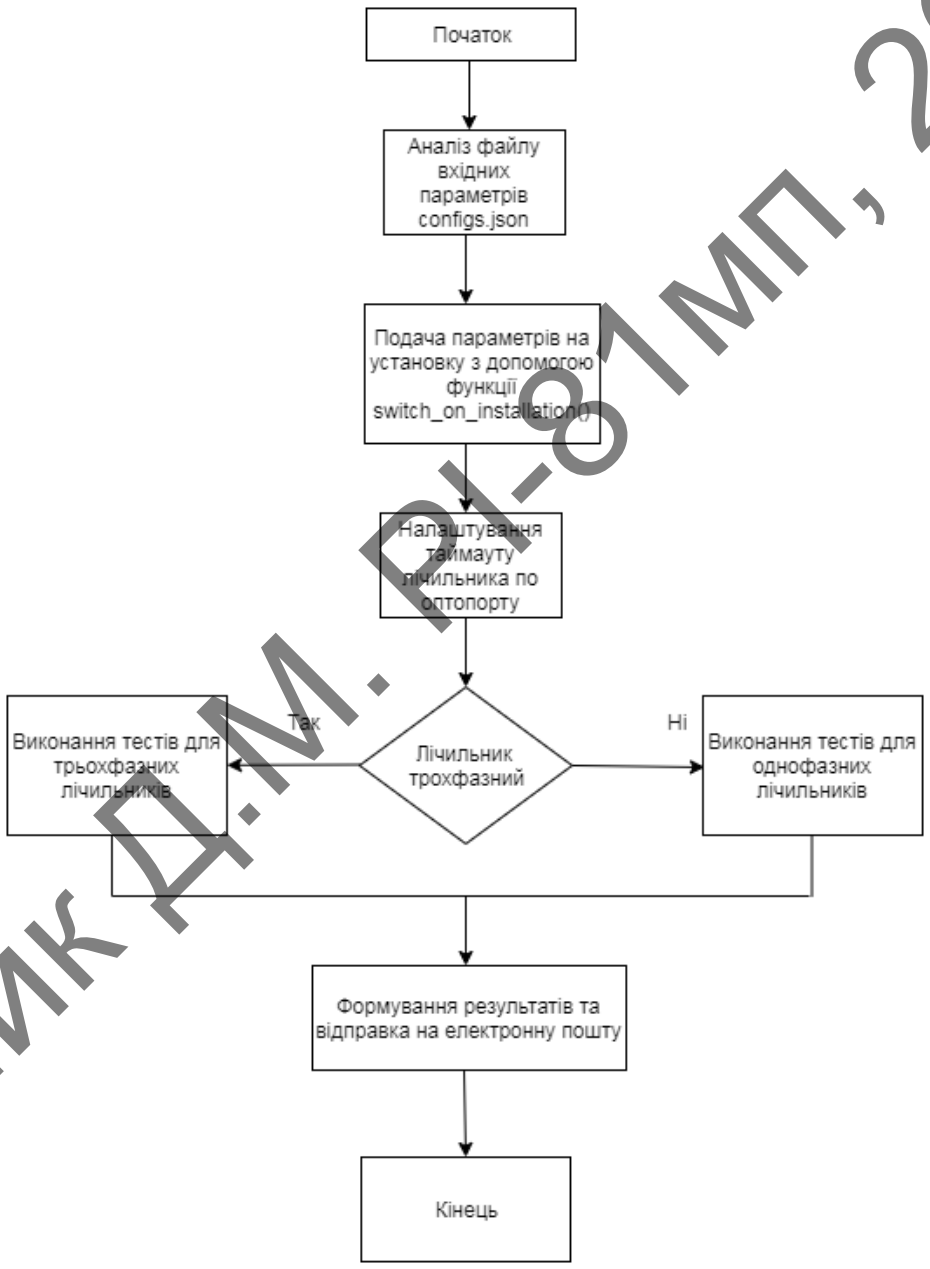


Рисунок 6.3 — Алгоритм роботи програми автоматизованого тестування лічильників

Для даної роботи дослідження проводилося на однофазних лічильниках. Для того, щоб виконати тестування, потрібно задати параметри при який бу-

де виконуватись перевірка. В якості середовища для вводу параметрів застосовується файл *configs.json, в* якому задається Com–порт калібрувальної установки і Com–порт для оптопорта. Приклад вхідних параметрів у файлі *configs.json* зображений на рисунку 6.4.

```
{
  "CALMET_PORT": "Com14",
  "OPTO_PORT": "Com10",
  "METER_TYPE": "NIK 2104 ARP6T.1802.MC.21",
  "USER_MAIL": "User@gmail.com"
}
```

Рисунок 6.4 — Приклад вхідних параметрів в файлі *configs.json*

JSON — це текстовий формат обміну даними між комп'ютерами. JSON базується на тексті, може бути прочитаним людиною. Формат дає змогу описувати об'єкти та інші структури даних. Цей формат використовується переважно для передачі структурованої інформації через мережу [15].

Файл *check_registers_of_enegry.py* запускає всі функції які перевіряють коректність накопичення енергії лічильником. Перед запуском тестів відбудеться аналіз вхідних параметрів у файлі *configs.json*, алгоритм якого зображений на рисунку 6.5.

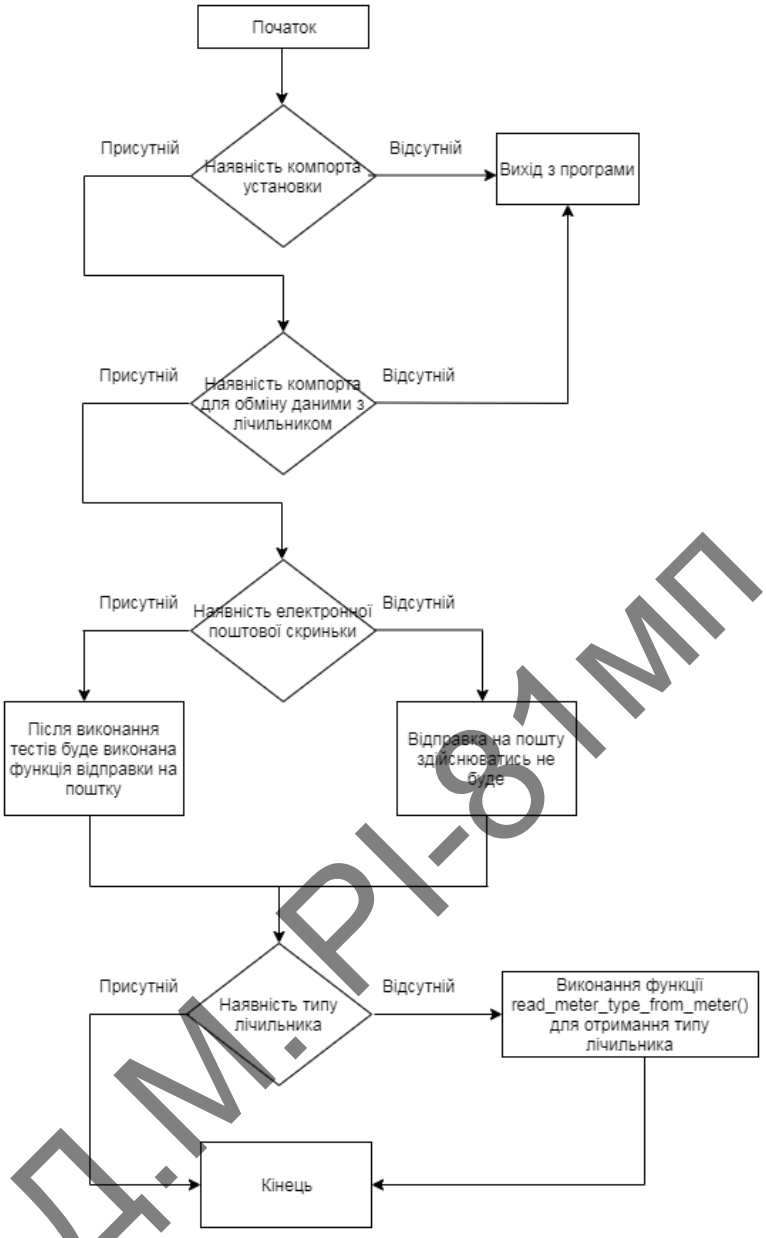


Рисунок 6.5 — Алгоритм аналізу вхідних параметрів

Відсутність Com–портів або не правильне їх вказанання призводить до помилки в програмі. Бажано задати тип лічильника, якщо він не заданий програма сама витягне дані з лічильника. Нижче представлена функція, здатна отримувати тип лічильника.

```
def _read_meter_type_from_meter(self):
        calmet_port = self.from_json("CALMET_PORT")
        opto_port = self.from_json("OPTO_PORT")
        port = sys.argv[1] if len(sys.argv) > 1 else calmet_port
```

```
        driver = serial.Serial(port=port, baudrate=57600, stopbits=1,
    rtscts=1, bytesize=8)
        calmet = Calmet(driver)
        calmet.set_voltage(220)
        calmet.set_current(0.5)
        one_phase_meter = (0, 1, 1, 0)
        calmet.set_stendby_or_operate(*one_phase_meter)
        calmet.get_response()
        port_meter = sys.argv[1] if len(sys.argv) > 1 else opto_port
        driver_meter = serial.Serial(port=port_meter, baudrate=9600,
    timeout=1.5)
        with Tester(driver_meter, "e") as test:
            test.Authoriz()
            meter_type = test.Get(METER_TYPE_OBIS)
             return meter_type
```

Функція *def _read_meter_type_from_meter()* — читає з файлу *configs.json*, та отримує дані про Com–порти установки і оптоголови. Подає запит установці на 220 В з метою запустити лічильник. Коли контролер лічильника буде в роботі, тоді робиться запит до лічильника щоб отримати його тип, командою *test.Get(METER_TYPE_OBIS).*

Результати тестування можуть бути відправлені на пошту. Для цього необхідно у файлі *configs.json* вказати електронну скриньку.

Після виконання аналізу вхідних параметрів почне виконуватись код програми який знаходиться в тілі умови *if __name__ == '__main__':*

Виконання наступного сценарію відбувається нижче.

```
if __name__ == '__main__':
   try:
      send_to_mail = True if MY_MAIL else False
      my_logger = LoggerMaker()
      my_logger.make_logger()
```

```
_switch_on_installation(voltage=250, current=45)
port_meter = sys.argv[1] if len(sys.argv) > 1 else OPTO_PORT
driver_meter = serial.Serial(port=port_meter, baudrate=9600, timeout=1.5)
with NikTest.Tester(driver_meter, "e") as test:
        test.Authoriz()
        before = test.Get(TIMEOUT_CONNECTION_OBIS)
        # 255 second timeout by optoport
         test.Set(TIMEOUT_CONNECTION_OBIS, [18, 255])
        test.Set(DST_TRANSITION_OBIS, [3, False])
        if METER_TYPE[5] == '3':
                tests_for_three_phase_meter()
        else:
                tests_for_one_phase_meter()
        test.Set(TIMEOUT_CONNECTION_OBIS, [18, before])
        SendLogToMail.send_email(MY_MAIL, my_logger.full_log_path,
        file_name, send_log=send_to_mail)
  except Exception as err:
        logging.info(err)
```

Змінна *send_to_mail* грає роль прапора. Якщо змінна *MY_MAIL* немає ніяких даних (*None*) тоді відправка по пошту не відбувається. *MY_MAIL* має функцію *configs.check_user_mail()* яка перевіряє наявність електронної скриньки в файлі конфігурації *configs.json*.

Створюється об’єкт логера тобто *my_logger* щоб записувати всю інформацію про виконання програми.

Функція *_switch_on_installation()* подає параметри на установку.

```
def _switch_on_installation(voltage, current, cos_=None, sin_=None):
    if METER_TYPE[5] == '1':
        calmet.set_voltage(voltage)
        calmet.set_current(current)
    if cos_:
```

```
            calmet.set_angle(Computation.cos_to_degree(cos_))
        if sin_:
            calmet.set_angle(Computation.sin_to_degree(sin_))
        if METER_TYPE[5] == '3':
            calmet.set_voltage(voltage, voltage, voltage)
             calmet.set_current(current, current, current)
            angle = None
        if cos_ is not None:
            angle = Computation.cos_to_degree(cos_)
        if sin_ is not None:
            angle = Computation.sin_to_degree(sin_)
      if angle is not None:
            calmet.set_angle(angle, angle, angle)
    one_phase_meter = (0, 1, 1, 0) if METER_TYPE[5] == '1' else (0, 0, 0, 0, 0, 0)
    calmet.set_stendby_or_operate(*one_phase_meter)
    calmet.get_response()
```

Для однофазного лічильника виконається умова *if METER_TYPE[5] == '1'* . Файл в якому виконується ця функція підтягує клас *Calmet* в якому знаходяться всі функції по управлінню установкою. Для того щоб спрацювали функції подачі струму, а саме *calmet.set_voltage(voltage)* і *calmet.set_current(current)* потрібно налаштувати параметри зв'язку з установкою та їх основі створити об'єкт керування нею. Нижче приведені команди для зв'язку з калібрувальною установкою.

```
    port = sys.argv[1] if len(sys.argv) > 1 else CALMET_PORT
    driver = serial.Serial(port=port, baudrate=57600, stopbits=1, rtscts=1,
bytesize=8)
        calmet = Calmet(driver)
```

Створивши об'єкт даного класу, подаємо напругу та струм командами *calmet.set_voltage(voltage)* і *calmet.set_current(current)*. Головною метою роботи *_switch_on_installation()* це налаштувати лічильник в робочий режим для

того щоб налаштувати лічильник для виконання тестів. Налаштування лічильника електричної енергії виконується наступними командами:

```
before = test.Get(TIMEOUT_CONNECTION_OBIS)
# 255 second timeout connection by optoport
test.Set(TIMEOUT_CONNECTION_OBIS, [18, 255])
test.Set(DST_TRANSITION_OBIS, [3, False])
if METER_TYPE[5] == '3':
    tests_for_three_phase_meter()
if METER_TYPE[5] == '1':
    tests_for_one_phase_meter()
test.Set(TIMEOUT_CONNECTION_OBIS, [18, before])
```

Програма контролера лічильника налаштована таким чином. Якщо напротязі якогось часу лічильник був у стані спокою, тобто до нього не було відправлених жодних запитів, тоді лічильник падає у стан спокою. В цьому стані лічильник не відповідає на будь які запити відіслані йому. Тому функція *test.Get(TIMEOUT_CONNECTION_OBIS)* дізнається таймаут за який лічильник падає у стан спокою, а функція *test.Set(TIMEOUT_CONNECTION_OBIS, [18, 255])* встановлює 255 секунд таумаут, якого вдосталь буде для виконання тестів.

Для однофазного лічильника виконається функція *tests_for_one_phase_meter()* вона слугує обгорткою тестів які будуть проводитись з лічильником. В цій функції, що представлена нижче, проводиться виконання функцій які проводять тестування накопичення енергії лічильником.

```
def tests_for_one_phase_meter():
""" running all test for one phase meter"""
    ActiveEnergy.active_energy_summary()
    ActiveEnergy.total_active_energy_summary()
    ActiveEnergy.full_active_energy_summary()
    if "R" in METER_TYPE:
        ReactiveEnergy.reactive_energy_summary()
```

6.3.2 Опис підпрограм по тестуванню сучасних лічильників

Для тестування основних видів енергії приведені наступні функції:

active_energy_summary() — проводить тестування активної позитивної енергії A+, а також активної негативної енергії A-,

total_active_energy_summary() — виконує тестування повної активної енергії, а також повної активної енергії по зеленому тарифу,

full_active_energy_summary() — виконує тестування повної енергії S+, а також повної негативної енергії S-,

reactive_energy_summary() — проводить тестування реактивної позитивної енергії, а також реактивної негативної енергії.

Алгоритм підпрограм для тестування лічильників приведений на рисунку 6.6.

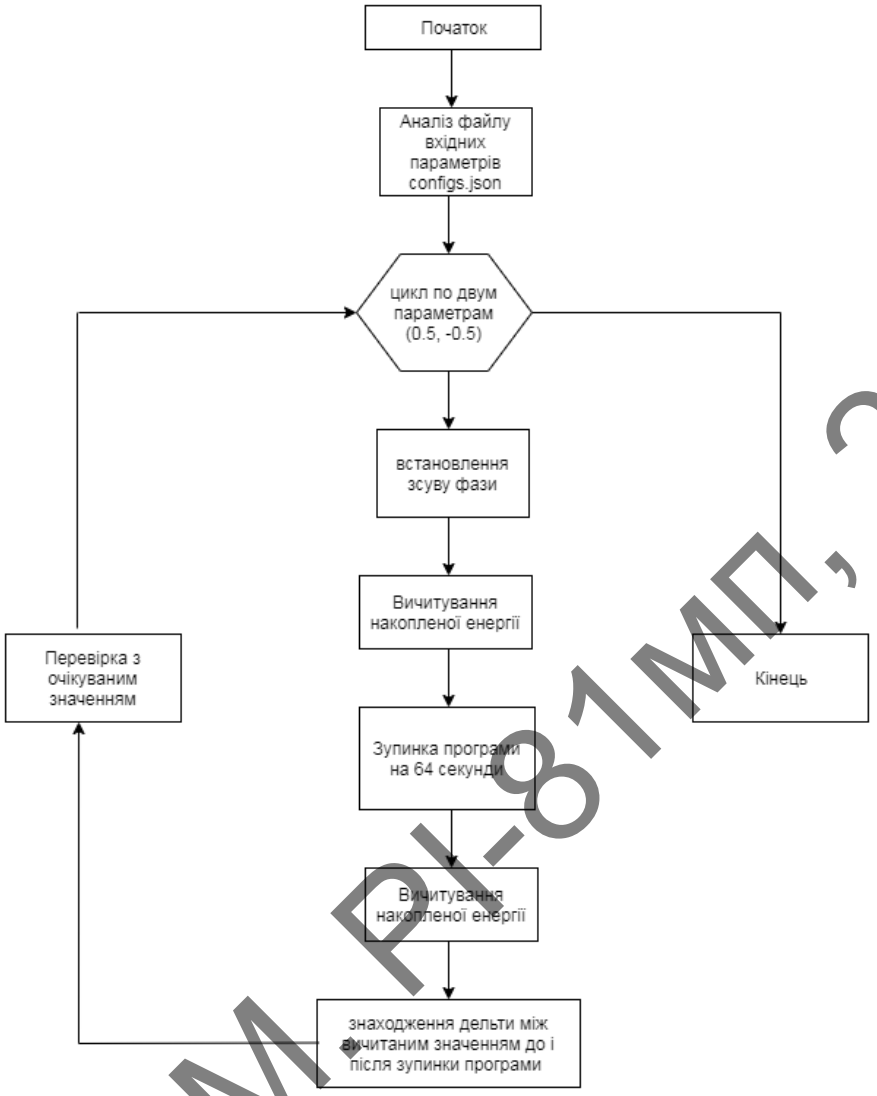


Рисунок 6.6 —Алгоритм підпрограм для тестування лічильників

Код програми з тестування активної позитивної енергії, а також активної енергії в мережі показаний в додатку В.

Робота тесту полягає в наступному. Спочатку на установку подається *cos(0.5)* для того щоб почала накопичуватись активна позитивна енергія. Далі читається з лічильника теперішнє значення енергії. Чекаємо 64 секунди, щоб лічильник нарахував 100 Вт*с. Після затримки, знову читаємо теперішню енергію і рахуємо дельту. Тепер у змінній *delta_energy* знаходиться значення енергії яку нарахував лічильник за 64 секунди.

Наступним кроком йде перевірка з еталоном, з допомогою функції *calculate_active_energy_by_one_phase*

@staticmethod

```
def calculate_active_energy_by_one_phase(voltage, current, cos):
    result = (voltage * current * cos) / 3600
    return result * 64  # 64 because at 64 seconds 100 watts winds up (a round number is convenient to debug)
```

Ця функція розраховує значення енергії, при параметрах 250В, 45А і cos(0.5), за 64 секунди.

Лічильник має допустиму похибку вимірювання 1%. Як тільки функція повернула нам 100 Вт, функція *deviation_measure_plus()* вираховує похибку одного відсотка в плюс, тобто 101 Вт*с, а функція *deviation_measure_minus()* вираховує похибку в мінус, тобто 99 Вт*с. Сформувавши значення, команда *res.append(_check_correctness_data(minus, plus, abs(delta_energy), "A-registers"))* порівнює виміряне значення енергії з розрахованим і записує результат у змінну *result.*

Наступна ітерація циклу *for* повертає значення косинуса -0.5. І після передачі даних установці з даним кутом зсуву, проходить тестування активна негативна енергія тобто *A-*. Результат порівняння записується у змінну *result.*

```
if len(res) == 2 and all(res):
        logging.info("OK")
        return True
else:
        logging.info(res)
        return False
```

Вище приведений код слугує для фіксування результату тесту. Якщо тестування активної позитивної енергії та активної негативної енергії пройшов успішно, тоді у файл записується «*OK»,* в інакшому випадку функція *_check_correctness_data()* повідомить який тип енергії не пройшов перевірку. Якщо помилка буде система або з боку установки, тоді виконається умова *else* і всі помилки будуть записані у файлі.

Алгоритм роботи наступних функцій тестування відрізнятись буде лише зверненням до пам’яті лічильника яка накопичує тестуєму енергію, а також

функцією по розрахунку енергії з якою порівнюється виміряне значення енергії.

Код по тестуванню повної активної енергії, а також повної активної енергії по зеленому тарифу відображений у додатку Г.

В даній функції використовуються посилання на адресу пам'яті лічильника. В цій пам'яті постійно накопичується енергія. Логіка тесту та сама, відрізняється лише посиланням на пам'ять в якій накопичується повна активна енергія, а також окрема пам'ять виділена на накопичення повної енергії по зеленому тарифу.

Далі відображений код функції яка тестує повну позитивну енергію і повну енергію в мережі у додатку Ґ.

Дана функція не аналізує тип лічильника тому, що даний тип енергії не залежить від фазності лічильника. Також у змінних *active_energy* та *reactive_energy* знаходиться результат розрахунку фактичного значення енергії, які беруть участь у розрахунку повної енергії *calculated_full_energy*.

Тестування реактивної позитивної енергії, а також реактивної енергії в мережі виконує функція *reactive_energy_summary()* код програми якої знаходиться у додатку Д.

Функція фактичного розрахунку реактивної енергії *calc_react_ener_one_ph(voltage, current, angle),* якій передаються вхідні параметри напруги, струму, та синус кута між напругою та струмом. Тільки в цьому випадку посилаємо запит установці на встановлення синусу 0.5 та синусу -0.5, якщо тестуємо реактивну негативну енергію. Калібрувальна установка Calmet C300B приймаю лише градуси, тому для переведення радіан в градуси була написана функція перетворення *Computation.sin_to_degree()*. Аналогічно для косинуса *Computation.cos_to_degree()*.

Після виконання тестів виконується функція по зміні таймауту. Щоб встановити налаштування таймауту на попереднє значення, а також виконується функція відправки на електронну пошту результатів проведення тестів.

7 РЕЗУЛЬТАТИ ЕКСПЕРЕМЕНТАЛЬНИХ ДОСЛІДЖЕНЬ ПРОГРАМИ ТЕСТУВАННЯ

Нехай потрібно протестувати однофазний лічильник електричної енергії запропонованим програмним забезпеченням автоматизованого тестування та ручним тестуванням.

Підключили комп’ютер до оптопорту лічильника та встановили останній на установку тестування Calmet C300B. Під'єднали USB-порт комп’ютера до установки тестування кабелем інтерфейсу RS–232.

Для запуску програми потрібно виконати такий перелік дій:

– встановити інтерпретатор мови програмування *python;*

– вказати Com–порти лічильника та калібрувальної установки у файлі *config.json*;

– запустити програму Windows PowerShell (командний рядок Windows) в корені проекту;

– в PowerShell написати команду *python check_registers_of_energy.py;*

– дочекатись автоматичного закриття програми Windows PowerShell.

Програма тестує наступні види енергії:

– активна енергія та активна енергія в мережі;

– повна активна енергія та повна активна енергія в мережі;

– повна енергія та повна енергія в мережі;

– реактивна енергія та реактивна енергія мережі.

7.1.1 Приклад автоматизованого тестування

Програма автоматизованого тестування виконується в середньому п’ять хвилин. Основний час займає затримка програми, яка виконується два рази у кожній функції тестування. Затримка програми триває 64 секунди на один параметр. Затримка програми була обрана емпіричним способом з метою зручності аналізу результатів. При вхідних даних U=250В, I=45А, cos=0,5 за 64 секунди, лічильник має нарахувати 100Вт*с. Нижче представлена формула розрахунку даної величини.

$$E = (\frac{U \cdot I \cdot \cos}{3600}) \cdot 64 = \frac{250 \cdot 45 \cdot 0.5}{3600} \cdot 64 = 100 Bm \cdot c \qquad (7.1)$$

Виконання автоматизованих тестів проводили на лічильнику, який пройшов калібрування (в іншому випадку він не рахував би енергію). При запуску програми калібрування лічильник налаштовується на вимірювання та готується до фіксації накопиченої енергії.

Після запуску програми результати тестування лічильника електричної енергії записуються у файл check_registers_of_energy.txt у вигляді (рис. 7.1).

2019-11-28 16:54:02,910 - check_active_energy_summary_total - INFO - 101, MDM 99.0, MDP 101.0, for |A+| - |A-| registers

2019-11-28 16:55:08,936 - check_active_energy_summary_total - INFO - 100, MDM 99.0, MDP 101.0, for |A+| + |A-| registers

2019-11-28 16:55:08,937 - check_active_energy_summary_total - INFO - OK

2019-11-28 16:56:14,960 - check_active_energy_summary - INFO - 100, MDM 99.0, MDP 101.0, for A- registers

2019-11-28 16:57:20,988 - check_active_energy_summary - INFO - 100, MDM 99.0, MDP 101.0, for A+ registers

2019-11-28 16:57:20,989 - check_active_energy_summary - INFO - OK

2019-11-28 16:57:23,050 - check_full_active_energy_summary - INFO - Get error:4, obis:[1, 0, 10, 8, 0, 255, 3, 2]

2019-11-28 16:58:29,077 - check_reactive_energy_summary - INFO - 100, MDM 99.0, MDP 101.0, for R- registers

2019-11-28 16:59:35,087 - check_reactive_energy_summary - INFO - 101, MDM 99.0, MDP 101.0, for R+ registers

2019-11-28 16:59:35,088 - check_reactive_energy_summary - INFO - OK

Рисунок 7.1 — Склад файлу результатів тестування лічильника

На вище показаних результатах показано дату *(2019-11-28)* і час *(16:54:02,910)*, назву тесту (*check_active_energy_summary_total*), статус запису даних (*INFO*), результати тестування (*101, MDM 99.0, MDP 101.0*) та вид тестуємої енергії (*for |A+| + |A-| registers*).

Результат тестування ділиться на три частини:

– фактичний результат *101 Вт*с*;

– очікуваний (розрахований) результат з похибкою в мінус *MDM 99.0 Вт*с*;

– очікуваний результат з похибкою в плюс *MDP 101.0 Вт*с*.

Рядок (*for |A+| + |A-| registers)* означає, що тестування відбулося для повної активної енергії.

Тестування повної енергії завершилось помилкою *Get error:4, obis:[1, 0, 10, 8, 0, 255, 3, 2]*. Рядок *Get error: 4* означає, що вичитування даних виконано з помилкою 4, за адресою *obis:[1, 0, 10, 8, 0, 255, 3, 2]*.

Контролер лічильника таким чином пояснює нам що даної адреси немає.

Перелік можливих помилок наведено на рисунку 7.2.

```
enum
{
                ACCESS_RESULT_SUCCESS = 0,
                ACCESS_RESULT_HARDWARE_FAULT = 1,
                ACCESS_RESULT_TEMPORARY_FAILURE = 2,
                ACCESS_RESULT_READ_WRITE_DENIED = 3,
                ACCESS_RESULT_OBJECT_UNDEFINED = 4,
                ACCESS_RESULT_OBJECT_CLASS_INCONSISTENT = 9,
                ACCESS_RESULT_OBJECT_UNAVAILABLE = 11,
                ACCESS_RESULT_TYPE_UNMATCHED = 12,
                ACCESS_RESULT_SCOPE_OF_ACCESS_VIOLATED = 13,
                ACCESS_RESULT_DATA_BLOCK_UNAVAILABLE = 14,
                ACCESS_RESULT_LONG_GET_ABORTED = 15,
                ACCESS_RESULT_NO_LONG_GET_IN_PROGRESS = 16,
                ACCESS_RESULT_LONG_SET_ABORTED = 17,
                ACCESS_RESULT_NO_LONG_SET_IN_PROGRESS = 18,
                ACCESS_RESULT_OTHER_REASON = 250
};
```

Рисунок 7.2 — Перелік можливих помилок

З даного списку на рисунку помітили опис четвертої помилки. *ACCESS_RESULT_OBJECT_UNDEFINED* — даний опис означає що в лічильнику не передбачено встановлення даної опції. Тобто лічильник не буде вимірювати даний тип енергії хоча схематично це передбачено.

Обґрунтування даної помилки має значення для замовника. Якщо в технічному завданні не передбачено вимірювання такого параметру тоді не виділяють пам’ять під цей тип енергії.

Час за який виконується тестування одного виду енергії становить 64 секунди.

7.1.2 Приклад ручного тестування

Ручне тестування здійснюється з допомогою програми, яка була розроблена з метою ручного задання параметрів на установку.

Інтерфейс програми Calpro 300 зображений на рисунку 7.3.

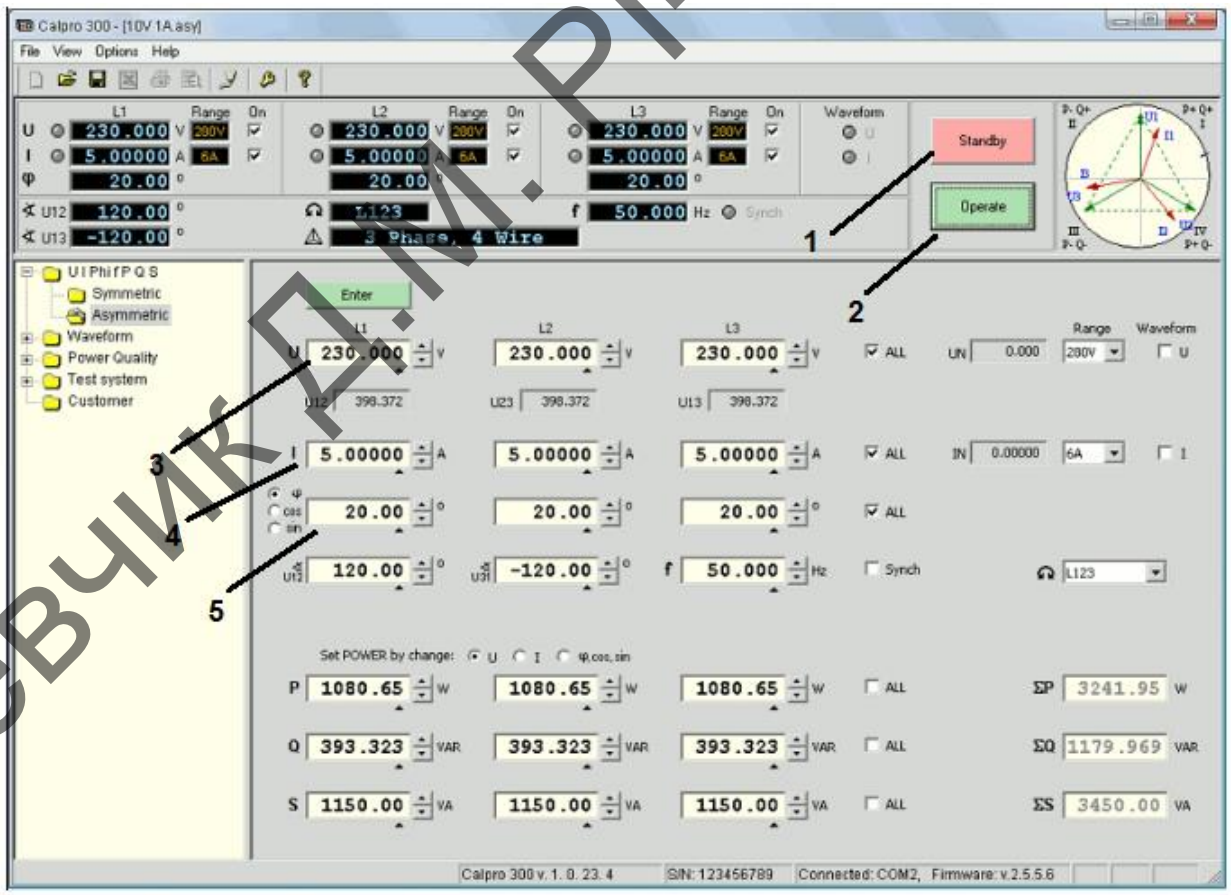


Рисунок 7.3 — Інтерфейс програмного середовища Calpro 300

На передній панелі програми Calpro 300 зображено: 1 – кнопка *Standby* для вимикання установки, 2 – кнопка *Operat*e для включення установки, 3 – поле для вводу вхідної напруги по першій фазі, 4 – поле для вводу вхідного струму по першій фазі, 5 – поле для вводу куту зсуву.

Ручне тестування виконується в такій послідовності:

1) вмикаємо установку;
2) подаємо напругу на установку;
3) фіксуємо поточне значення енергії, яке відображається на дисплеї лічильника;
4) подаємо параметри для тестування: 250В, 45А, 0.5cos;
5) заміряємо час і чекаємо 64 секунди;
6) після затримки миттєво вимикаємо установку;
7) подаємо напругу на лічильник без навантаження;
8) вираховуємо дельту.

При ручному тестуванні вирахувана дельта може бути не точною. Оскільки потрібно ретельно слідкувати за часом. Крім того важливою частиною тесту є врахування часу на відключення установки. Даний сценарій є складним і потребує ретельної уваги, що призводить до помилкового тестування. У разі помилкового тестування, яке може дуже часто повторюватись, слід провести тестування мінімум п’ять разів.

Результати тестування одного виду енергії:

- 97 Вт·с;
- 99 Вт·с;
- 104 Вт·с;
- 98 Вт·с;
- 101 Вт·с.

Середнє величина фактичного результату 99.8 Вт·с. Даний результат входить в діапазон від 99 Вт·с до 101 Вт·с, а значить тест пройшов успішно. Середній час на тестування одного виду енергії 6400 секунд.

Запропонована програма виконує тестування одного виду енергії за 64 секунди.

Дана програма виграє в часі тестування. Часу на тестування одного параметру потрібно в 10 разів менше.

8 ОХОРОНА ПРАЦІ

У даному розділі дипломної роботи розглянуто питання охорони праці при виконанні досліджень, розробці та експлуатації вимірювальної системи. Проводиться визначення та оцінка шкідливих впливів. Оскільки робота є, значною мірою, теоретичною, основна увага приділена питанням забезпечення належних умов праці з використанням персональних електронно-обчислювальних машин (ПЕОМ), електробезпеки та безпеки в надзвичайних ситуаціях з урахуванням вимог ДСанПіН 3.3.2.007–98 та «Вимог щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями» Зареєстровано в Міністерстві юстиції України 25 квітня 2018 р. №508.31960.

8.1 Основні потенційно шкідливі та небезпечні , шкідливі фактори

До основних шкідливих і небезпечних факторів, що впливають на людей, зайнятих у розробці та виробництві Радіоелектронних апаратів, відносяться:

1. Недостатня освітленість робочої зони (умови освітленості виробничих приміщень повинні відповідати нормам, зазначеним у ДБН В.2.5–28–2006);
2. Небезпека ураження електричним струмом;
3. Незадовільні параметри мікроклімату робочої зони (величини показників мікроклімату у виробничих приміщеннях повинні відповідати нормам, зазначеним у ДСН 3.3.6.042–99);
4. Вміст у повітрі робочої зони шкідливих речовин різного характеру впливу в концентраціях, що перевищують гранично допустимі (ГДК шкідливих речовин у повітрі робочої зони повинні відповідати нормам, зазначеним у ГОСТ 12.1.005-88 і ГОСТ 12.1.007–80);

5. Підвищений рівень шуму на робочому місці (допустимі рівні звукового тиску в октавних смугах частот, рівні звуку й еквівалентні рівні звуку на робочих місцях варто приймати відповідно до ДСН 3.3.6.037–99);
6. Вплив шкідливих факторів моніторів ПЕОМ (ДСанПіН 3.3.2.007–98) Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин);
7. Психофізіологічні фактори.

8.2 Технічні рішення та організаційні заходи з безпеки гігієни праці та виробничої санітарії

8.2.1 Електробезпека

Згідно ДСТУ IEC 61140:2015 все електрообладнання, що знаходиться в робочому приміщенні можна віднести I класу по електрозахисту .

> I клас — обладнання, що приєднується для ланцюга живлення трьохконтактними вилками, один з цих контактів при цьому з'єднується з заземленим контактом розетки.

Робоче приміщення за ступенем небезпеки ураження людей електричним струмом можна віднести, згідно Правил Улаштування Електроустановок, до приміщень без підвищеної небезпеки, оскільки:

– відносна вологість повітря не перевищує 75%;

– матеріал підлоги (паркет) є діелектриком;

– температура повітря не досягає значень, більших ніж 35ºС;

– відсутня можливість одночасного торкання людини до елементів, що мають з'єднання з землею, металоконструкцій будівель, технологічних апаратів, механізмів та ін. з одного боку і до металевих корпусів електрообладнання – з іншого.

Основними технічними засобами, що забезпечують безпеку робіт (згідно ПУЕ–2017) є: надійна ізоляція, захисне заземлення, занулення, захисне відключення, засоби індивідуального захисту. У системі трифазних мереж із глухо заземленою нейтраллю, яка використовується у науково–дослідницькій

лабораторії, найкращими засобами захисту є: надійна ізоляція струмоведучих частин електроустаткування і занулення відповідно до ПУЕ (з'єднання елементів, що перебувають під напругою, із глухо заземленою нейтраллю). Крім того, для заземлення переносних частин обладнання застосовують спеціальне з'єднання.

Мережа у приміщенні окрема трипровідна, виконана шляхом прокладання фазового, нульового робочого та нульового захисного провідників та з використанням автоматів струмового захисту як в мережі, так і в електроприладах окремо. Нульовий захисний провідник використовується для заземлення електроприладів і прокладається від розподільчого щита, до розеток живлення. Використання нульового робочого провідника як нульового захисного провідника заборонено.

Електромережі штепсельних з’єднань та електророзеток слід виконувати лише за магістральною схемою, по 3–6 з’єднань або електророзеток в одному колі. Групові з’єднання слід виконувати із спеціальних матеріалів, що є термотривкими з урахуванням правил пожежної безпеки України.

Усі пристрої що використовуються мають корпусні елементи, виконані із ізолюючої пластмаси, що унеможливлює доторкання до елементів, що знаходяться під напругою.

Що стосується лабораторного макета – усі його частини, що є активними під час роботи не знаходяться під напругою, що перевищувала би допустиму напругу дотику.

8.2.2 Вимоги до приміщень в яких розміщені ВДТ ПЕОМ

Облаштування роюочих місць, обладнаних ЕОМ, ВДТ повинно забезпечувати:

– належні умови освітлення приміщення і робочого місця, відсутність відблисків;

– оптимальні параметри мікроклімату (температура, відносна вологість, швидкість руху, рівень іонізпції повітря)

– належніергономічні характеристики основних елементів робочого місця.

Заборонено розміщувати робочі місця з ВДТ, ЕОМ у підвальних приміщенняї, на цокольних поверїах, поряж з приміщеннями, в яких рівні шуму та вібрації перевищують допустимі значення (поряд з механічними цехами, майстернями тощо), з мокрими виробництвами, з вибухопожежонебезпечними приміщеннями категорій А і Б, а також над такими приміщеннями або під ними.

Приміщення мають бути обладнані системами водяного опалення, кондиціонування або припливно–витяжною вентиляцією відповідно СНиП 2.04.05–91.

Згідно ДНАОП 0.00–1.31–99 площу приміщень визначають з розрахунку, що на одне робоче місце вона має становити не менше ніж 6 м^2, а об’єм не менше ніж 20м^3 з урахуванням максимальної кількості осіб, які одночасно працюють у зміні. Приміщення являє собою кімнату розміром 7 х 5м, висотою 4м. Розмір дверного прорізу 1,5м.

Площа й об’єм приміщення знаходимо по формулах:

$$S = a \cdot b, \quad (8.1)$$

$$V = S \cdot h, \quad (8.2)$$

Де а — довжина, b — ширина, h — висота приміщення.

Маємо $S = 7 \cdot 5 м^2$, $V = 35 \cdot 4 = 140 м^2$.

Зведемо нормативні та фактичні дані приміщення в таблицю 8.1.

Таблиця 8.1 — Нормативні та фактичні дані приміщення

Назва характеристи	Нормативна	Фактична
Площа приміщення з розрахунку	> 6м^2	35м^2
Об’єм приміщення з розрахунку на 1	> 20м^3	> 140м^3
Висота приміщення	3,5 – 4м	4м

Продовження таблиці 8.1

Розміри дверей	≥1,1х1,8м	1,5х2м
Відстань від стіни зі світловими прорізами до ВДТ	≥1м	1,5м

На підставі отриманих результатів можна зробити висновок, що геометричні розміри приміщення цілком відповідають нормативним вимогам.

8.2.3 Розрахунок електромережі на здатність відключення в аварійному режимі

Аварійним режимом роботи для електричного автомата є коротке замикання.

Струм короткого замикання визначається за формулою:

$$I_{к.з.} = \frac{U_ф}{R_ф + R_н + Z_T}, \quad (8.3)$$

де $U_ф$ – фазова напруга, становить 220 В;

$R_ф$ – опір фазового проводу, становить 0,7 Ом;

$R_н$ – опір нульового проводу, становить 0,2 Ом;

Z_T – розрахунковий опір трансформатора, становить 0,1 Ом.

Підставивши у формулу, отримаємо:

$$I_{к.з.} = \frac{220}{0,7 + 0,2 + 0,1} = 220A. \quad (8.4)$$

Необхідно дотриматись виконання наступної вимоги :

$$I_{к.з.} \geq 1,4 \cdot I_{авт}.$$

Відповідно струм відключення автомату $I_{авт}$ повинен бути не більше:

$$I_{авт} \leq \frac{I_{к.з.}}{1,4} \cong 158A$$

У приміщенні встановлено автомат із струмом відключення 15А, що задовольняє цій вимозі.

Напругу дотику до зануленого обладнання визначають за формулою:

$$U_{дот} = I_{к.з.} \cdot R_н \quad (8.5)$$

Підставивши, отримаємо:

$$U_{дот} = 220 \cdot 0{,}2 = 44B.$$

Напруга дотику менша за допустиму за час спрацювання автоматів струмового захисту і відповідає вимогам ПУЕ–2017.

8.2.4 Відповідність параметрів робочого приміщення санітарним нормам

Першочергово слід розглянути відповідність приміщення, в якому проводяться всі дослідницькі дії, санітарним нормам.

Довжина та ширина приміщення, в якому проводяться всі дії складають $a = 6м$ та $b = 4м$ відповідно.

Тоді площа приміщення становить:

$$S = a \cdot b,$$

$$S = 6 \cdot 4 = 24м^2.$$

Висота приміщення $h = 2{,}75м$, тому об’єм приміщення становить:

$$V = S \cdot h,$$

$$V = 24 \cdot 2{,}75 = 66м^3.$$

Кількість людей n, що одночасно працюють в приміщенні становить 4 чоловіки.

Площа та об’єм приміщення, виділені на одну людину тоді приймають значення

$$S_1 = S/n,$$

$$= 24/4 = 6м^2, \text{ та}$$

$$V_1 = V/n,$$

$$= 66/4 = 16{,}5м^3 \text{ відповідно.}$$

Це відповідає нормам, зазначеним у СНиП 2.09.02-85 (площа на одного працюючого не повинна бути меншою за $4{,}5м^2$, а об’єм – не менше $15м^3$).

8.2.5 Мікроклімат робочої зони

Метеорологічні умови на робочому місці можна охарактеризувати трьома наступними параметрами: температура повітря, відносна вологість повітря, швидкість руху повітря.

Метеорологічні умови повітря робочої зони регламентує ДСН 3.3.6.042–99.

Норми є різними для різних категорій тяжкості праці, періодів року.

Роботи ведуться в звичайному побутовому приміщенні, здебільшого в сидячому положенні, що не потребує значного фізичного напруження, тому їх можна віднести до категорії Іа (споживання енергії до 138 Дж/год., або 120 ккал/год.).

Таблиця 8.2 – Метеорологічні умови

Період року	Категорія робіт	Температура, °С			Відносна вологість повітря		Швидкість руху повітря, м/с	
		Оптимальна	Допустима на робочих місцях		Оптимальна	Допустима на постійному та непостійному робочих місцях	Оптимальна	Допустима на постійному та непостійному робочих місцях
			Постійних	Непостійних				
Холодний	Іа	22-24	21-25	18-26	40-60	75	0,1	Не більше 0,1
Теплий	Іа	23-25	22-28	20-30	40-60	55 при 28 °С	0,1	0,1-0,2

Проаналізувавши параметри приміщення на відповідність параметрам із таблиці було виявлено, що гранична допустима температура в теплий період року може бути перевищена. Для запобігання цьому використовується при-

родна вентиляція через віконні прорізи. У холодний період року застосовується система центрального опалення.

8.2.6 Виробничий шум

Характеристики шуму, який є допустимим на робочому місці регламентується ДСН 3.3.6.037–99.

Вплив зовнішніх джерел шуму відсутній. Внутрішніми джерелами шуму є ПЕОМ, телефони, та інші побутові прилади.

Допустимі рівні звукового тиску, рівні звуку та еквівалентні рівні звуку на робочих місцях нормуються відповідно до НПАОП 0.00–1.28–10 та ДСН 3.3.6.037–99.

Відповідно до ДСН 3.3.6.037–99, захист від шуму, створеного внутрішніми джерелами на робочих місцях, повинен здійснюватись в основному такими двома методами: зменшенням шуму від джерел та акустичною обробкою робочого приміщення звукоізоляційними матеріалами.

За даними вимірювань, рівень шуму у будь–якій точці приміщення не перевищує 50 дБ, що входить в межі нормального рівня шуму у робочих приміщеннях.

8.2.7 Охорона праці при використанні ПЕОМ

Відповідно до ДСанПіН 3.3.2.007–98 основними шкідливими та небезпечними виробничими факторами, які пов’язані з роботою на ПЕОМ, є:

— механічні шуми, пов’язані з роботою комп’ютера;
— значне напруження органів зору та пов’язана із цим загальна перевтома організму;
— можливість ураження електричним струмом;
— значне навантаження на пальці та кисті рук під час роботи за ПЕОМ, що при відсутності профілактики і медичного контролю може перетікати у хронічні захворювання;

— довготривале перебування в положенні сидячи під час роботи на ПЕОМ викликає застійні явища в організмі людини, які є надзвичайно шкідливими.

Професійна діяльність людини на ПЕОМ може призводити до порушення функціонування органів зору, кістково–м'язової системи.

Комп'ютерна техніка, що використовувалась під час дослідів є сучасною та виконана з урахуванням усіх вимог щодо охорони праці. Екран має гарні кути обзору, функцію автоматичного контролю за яскравістю зображення та застосовує технологію послідовної розгортки, що зменшує стомлення зорових органів при роботі із ним.

Окрім цього, на ПЕОМ було встановлено спеціальне програмне забезпечення, що виконує функцію сповіщення користувача про необхідність розминки органів зору(один раз на 10 хвилин) та фізичного розвантаження для зняття напруження із м'язів (один раз на годину).

Це відповідає вимогам тривалості регламентованих перерв під час роботи з ПЕОМ та заходам щодо зниження нервово-емоційного напруження, втомленості зорового аналізатора, для поліпшення мозкового кровообігу і запобігання втомі, які передбачені ДСанПіН 3.3.2.007–98.

8.3 Безпека в надзвичайних ситуаціях

Безпека у надзвичайних ситуаціях регламентується ПЛАС («Положенням щодо розробки планів локалізації та ліквідації аварійних ситуацій і аварій»). Одними з основних складових ПЛАС є розробка технічних рішень та організаційних заходів щодо оповіщення, евакуації та дії виробничого персоналу у разі виникнення надзвичайної ситуації.

8.3.1 Вимоги щодо організації ефективної роботи системи оповіщення персоналу при НС

Для підвищення безпеки в надзвичайних ситуаціях (НС) пропонується встановлення системи оповіщення (СО) виробничого персоналу.

Оповіщення виробничого персоналу у разі виникнення НС, наприклад при пожежі, здійснюється відповідно до вимог НАПБ А.01.003-2009.

Необхідність обладнання виробничих приміщень певним типом СО визначається згідно з додатком Е до ДБН В.1.1–7–2016 "Захист від пожежі. Пожежна безпека об'єктів будівництва".

При обладнані виробничих будівель системою оповіщення, їх необхідно поділяти на зони оповіщування з урахуванням об'ємно-планувальних рішень будинків, шляхів евакуації, поділення на протипожежні відсіки тощо, а також з урахуванням вимог, що наведені в примітці 1 таблиці Е.1 додатка Е до ДБН В.1.1–7–2016.

Оповіщування про НС та управління евакуацією людей здійснюється одним з наступних способів або їх комбінацією:

- поданням звукових і (або) світлових сигналів в усі виробничі приміщення будівлі з постійним або тимчасовим перебуванням людей;
- трансляцією текстів про необхідність евакуації, шляхи евакуації, напрямок руху й інші дії, спрямовані на забезпечення безпеки людей;
- трансляцією спеціально розроблених текстів, спрямованих на запобігання паніці й іншим явищам, що ускладнюють евакуацію;
- розміщенням знаків безпеки на шляхах евакуації згідно з ДСТУ ISO 6309;
- ввімкненням евакуаційних знаків "Вихід";
- ввімкненням евакуаційного освітлення та світлових покажчиків напрямку евакуації;
- дистанційним відкриванням дверей евакуаційних виходів;
- зв'язком оперативного (чергового) персоналу СО (диспетчера пожежного поста) із зонами оповіщування.

Повинен бути забезпечений розподіл пріоритетів щодо повідомлень для виробничого персоналу у такій послідовності:

I — (найвищий) повідомлення оперативного (чергового) персоналу СО (диспетчера пожежного поста) під час пожежі, або у разі виникнення будь-якої іншої НС;

II — повідомлення, які записані на будь-якому носії та вмикаються автоматично від спрацювання систем пожежної автоматики, або за сигналом оперативного (чергового) персоналу СО (диспетчера пожежного поста);

III — службові повідомлення, що не стосуються організації та управління евакуацією людей.

У багатоповерхових виробничих будівлях, які поділені на протипожежні відсіки по вертикалі, СО повинна вмикатися одразу для всього протипожежного відсіку, де виникла пожежа. Затримку часу оповіщування про НС для інших вертикальних протипожежних відсіків будинку слід передбачати з урахуванням злиття потоків людей на шляхах евакуації.

Для трансляції мовних повідомлень необхідно використовувати повідомлення, які записані заздалегідь на відповідному носії інформації.

Текст мовного повідомлення повинен бути записаний українською мовою. Необхідність запису тексту мовного повідомлення та його транслювання іншими мовами визначається призначенням виробничих приміщень. Тривалість трансляції одного мовного повідомлення не повинна перевищувати 1 хвилини.

Пульти управління СО необхідно розміщувати у приміщенні пожежного поста, диспетчерської або іншого спеціального приміщення (в разі його наявності). Ці приміщення повинні відповідати вимогам ДБН В.2.5–56–2014.

У місцях, де є небезпека механічного ушкодження оповіщувачів, повинен бути забезпечений їх захист, що не порушує працездатності оповіщувачів.

Вимоги до світлових покажчиків "Вихід" приймаються відповідно до ДБН В.2.5–28–2006 "Інженерне обладнання будинків і споруд. Природне і штучне освітлення".

Вимоги до евакуаційного освітлення приймаються відповідно до ДБН В.2.5–28–2006 "Природне та штучне освітлення".

СО в режимі "Тривога" повинна функціонувати протягом часу, необхідного для евакуації людей з будинку, але не менше 15 хвилин.

Електропостачання СО здійснюється за І категорією надійності згідно з "Правила устройства електроустановок" (ПУЕ) від двох незалежних джерел енергії: основного від мережі змінного струму, резервного від акумуляторних батарей тощо.

Звукові оповіщувачі повинні відповідати вимогам ДСТУ EN 54–3:2003 "Системи пожежної сигналізації. Частина 3. Оповіщувачі пожежні звукові".

Мовні оповіщувачі повинні відтворювати нормально чутні частоти у діапазоні від 200 до 5000 Гц. Світлові оповіщувачі, які працюють у режимі спалахування, повинні бути червоного кольору, мати частоту мигтіння в межах від 0,5 Гц до 5 Гц та розташовуватись у межах прямої видимості з постійних робочих місць.

8.3.2 Обов'язки та дії персоналу у разі виникнення надзвичайної ситуації

У разі виявлення ознак НС працівник, який їх помітив, повинен:

негайно повідомити про це службою зв'язку органів ДСНС, вказати при цьому адресу кількість поверхів, місце виникнення пожежі, наявність людей, а також своє прізвище;

повідомити про пожежу керівника, адміністрацію, пожежну охорону підприємства;

організувати оповіщення людей про НС;

вжити заходів щодо евакуації людей та матеріальних цінностей;

вжити заходів щодо ліквідації наслідків НС з використанням наявних засобів.

Керівник та пожежна охорона установи, яким повідомлено про виникнення пожежі, повинні :

- перевірити, чи викликані підрозділи ДСНС;
- вимкнути у разі необхідності струмоприймачі та вентиляцію;
- у разі загрози життю людей негайно організувати їх евакуацію та їх рятування, вивести за межі небезпечної зони всіх працівників, які не беруть участь у ліквідації НС;
- перевірити здійснення оповіщення людей про НС;
- забезпечити дотримання техніки безпеки працівниками, які беруть участь у ліквідації НС;
- організувати зустріч підрозділів Державної служби України з надзвичайних ситуацій, надати їм допомогу у локалізації та ліквідації НС.

Після прибуття на НС підрозділів ДСНС повинен бути забезпечений безперешкодний доступ їх до місця, де виникла НС.

8.3.3 Пожежна безпека

Відповідно до ДСТУ Б.В.1.1.–36:2016 робоче приміщення лабораторії відноситься до категорії В по вибухопожежній небезпеці. Відповідно до та ДНАОП 0.00–1.32–01 клас робочих зон приміщення лабораторії по пожежонебезпеці – П–IIа. Можливими причинами пожежі в приміщенні є несправність електроустаткування, коротке замикання проводки, і порушення протипожежного режиму (використання побутових нагрівальних приладів, паління).

У зв’язку з цим, відповідно до вимог ПБЕ та ПУЕ, необхідно передбачити наступні заходи:

- постійний контроль стану засобів пожежогасіння;
- застосування автоматичних установок пожежної сигналізації;
- організація за допомогою технічних засобів, включаючи автоматичні, своєчасного оповіщення та евакуації людей.
- контроль за станом ізоляції струмоведучих дротів;
- заборонено паління в приміщенні;

– неприпустимість знаходження у приміщенні горючих і вибухонебезпечних речовин;
– допуск до роботи осіб, які в установленому порядку пройшли навчання, інструктаж і перевірку знань з пожежної безпеки.

Для гасіння пожежі в робочому приміщені лабораторії (клас пожежі „Е” – наявність електрообладнання під напругою) використовуються вогнегасники ОП–1 — “Момент” (2 шт.). Додатково в коридорі розташовані вогнегасники ОХП–10. Також на сходовій клітці розташований пожежний кран. Така кількість первинних засобів пожежогасіння відповідає вимогам ДСТУ 3675–98 та ISO3941–2007, якими передбачене обов’язкова наявність двох вогнегасників до 100 м^2 площі підлоги для приміщення типу конструкторське бюро.

Будинок має два евакуаційних виходів: через головний хід і додатковий евакуаційний вихід. Шляхи евакуації відповідають установленим нормам. Двері відкриваються назовні. Коридор веде до двох сходових кліток, одна з яких виходить безпосередньо на вулицю, а друга має вихід на вулицю через вестибюль і головний вхід. Сходова клітка виконана з не пальних матеріалів. Сходи мають природне бічне освітлення і штучне евакуаційне освітлення. Сходові площадки ширше коридорів. Усі співробітники ознайомлені з планом евакуації.

Дотримано усі вимоги СНиП 2.09.02–85 по вогнестійкості будинку і ширині евакуаційних проходів і виходів із приміщень назовні. Значення основних параметрів шляхів евакуації приведені в табл.8.3.

Таблиця 8.3 — Характеристики і норми еваковиходів

Параметр	Фактичне значення	Норма
Висота дверних прорізів	2,0 м	Не менше 2 м
Ширина дверних прорізів	0,8 м	Не менше 0,8 м

Продовження таблиці 8.3

Ширина проходу для евакуації	Більше 1,5 м	Не менше 1 м
Ширина коридору	2 м	Не менше 2 м
Число виходів з коридору	2	Не менше 2
Ширина сходового маршу	1,2 м	Не менше 1 м
Висота поруччя сходів	1 м	Не менше 0,9 м

У приміщенні є план евакуації. Мінімальний час евакуації в разі виникнення пожежі відповідає вимогам СНиП 2.01.02–85, а максимальне видалення робочих місць від евакуаційних виходів вимогам СНиП 2.09.02–85.

У приміщенні виконуються усі вимоги по пожежній безпеці відповідно до вимог НАПБ А.01.001–2004 “Правила пожежної безпеки в Україні”.

8.3.4 Основні вимоги щодо плану евакуації при пожежі

В даному підрозділі розглянемо план евакуації при пожежі, адже це одне з найнебезпечніших НС, яке може трапитись під час виробничого процесу.

Захист людей у разі пожежі є найважливішим завданням усієї системи протипожежного захисту. Вирішення цього завдання, в першу чергу, потребує впровадження ефективних евакуаційних заходів на випадок виникнення пожежі. Вирішення цього завдання досить складне, оскільки має власну специфіку та здійснюється іншими шляхами, ніж захист будівельних конструкцій чи матеріальних цінностей.

Виходи вважаються евакуаційними, якщо вони ведуть із приміщень:

- першого поверху безпосередньо назовні або через вестибюль, коридор, сходову клітку;
- будь якого поверху, крім першого, у коридор, що веде на внутрішню сходову клітку або безпосередньо на зовнішні відкриті сходи;
- у сусіднє приміщення на тому ж поверсі, яке забезпечене виходами, зазначеними у попередніх пунктах;
- цокольного, підвального, підземного поверху назовні безпосередньо через сходову клітку або коридор, що веде на сходову клітку, яка має вихід назовні.

Текстова частина плану евакуації, яка представляє собою таблицю з переліком та послідовністю дій у разі пожежі для конкретних посадових осіб і працівників, затверджується керівником об'єкту. План евакуації необхідно розташовувати на видному місці, а його положення повинні систематично відпрацьовуватись на практиці.

Також дуже важливо забезпечити протидимний захист робочих приміщень і особливо шляхів евакуації людей. Протидимний захист забезпечується обмеженням розповсюдження продуктів горіння по будівлях та приміщеннях, примусовим видаленням диму. Ці задачі вирішуються за допомогою об'ємно–планувальних та конструктивних рішень при проектуванні об'єктів, деякими технологічними прийомами в процесі будівництва, завдяки використанню спеціальних пристроїв і вентиляційних систем, які призначені для видалення диму зниження температури і конденсації продуктів горіння.

9 РОЗРОБЛЕННЯ СТАРТАП–ПРОЕКТУ

Даний розділ має на меті проведення маркетингового аналізу стартап проекту задля визначення принципової можливості його ринкового впровадження та можливих напрямів реалізації цього впровадження.

9.1 Опис ідеї проекту

В межах цього підрозділу аналізується зміст ідеї, можливі напрямки застосування, основні вигоди які може отримати користувач товару та відмінності від існуючих аналогів та замінників.

Таблиці 9.1 — Опис ідеї стартап–проекту

Зміст ідеї	Напрямки застосування	Вигоди для користувача
Автоматизоване тестування лічильників електричної енергії	Виробництво лічильників електричної енергії, наука	Збільшення кількості лічильників за одиницю часу

Стартап–проект являє собою програму з допогою якої виконується тестування лічильника на коректність зберігання та накопичення електричної енергії. Програма тестує всі види енергії, незалежно від фазності лічильника, а також проводить тестування тарифних лічильників.

9.2 Технологічний аудит ідеї проекту

В межах даного підрозділу проводиться аудит технологій, за допомогою якої можна реалізувати ідею проекту.

Для реалізації цього проекту потрібно вибрати мову програмування чи середовище програмування. Оглянуто такі варіанти:

1. Мова прогрування C# — об'єктно–орієнтована мова програмування з безпечною системою типізації. Даний інструмент являється одним з тих інструментів з допомогою яких виконують автоматизоване тестування. Синтаксис C# близький до C++ і Java. Мова має строгу статичну типізацію, підтримує поліморфізм, перевантаження операторів, вказівники на функції-члени класів, атрибути, події, властивості, винятки, коментарі у форматі XML

2. Мова програмування Java — об'єктно–орієнтована мова програмування, випущена компанією Sun Microsystems у 1995 році як основний компонент платформи Java. Синтаксис мови багато в чому походить від C та C++. Java програми виконуються у середовищі віртуальної машини Java. Java програми компілюються у байткод, який при виконанні інтерпретується віртуальною машиною для конкретної платформи.
3. Мова програмування Python — інтерпретована об'єктно–орієнтована мова програмування високого рівня зі строгою динамічною типізацією. Структури даних високого рівня разом із динамічною семантикою та динамічним зв'язуванням роблять її привабливою для швидкої розробки програм, а також як засіб поєднування наявних компонентів. Python підтримує модулі та пакети модулів, що сприяє модульності та повторному використанню коду. Інтерпретатор Python та стандартні бібліотеки доступні як у скомпільованій, так і у вихідній формі на всіх основних платформах. В мові програмування Python підтримується кілька парадигм програмування, зокрема: об'єктно–орієнтована, процедурна, функціональна та аспектно-орієнтована.

Таблиця 9.2 — Технологічна здійсненість проекту

№	Ідея проекту	Технології її реалізації	Наявність технології	Доступність технології
1	Автоматизоване тестування лічильників електричної енергії	C#	Так	Так
2		Java	Так	Так
3		Python	Ні	Так
Обрана технологія реалізації ідеї проекту: Python				

Даний проект можливо реалізувати і в якості технологічного шляху обрано Python через наявність у автора знань у мові програмування Python та можливості написання автоматизованих тестів та модулів для лічильників електричної енергії.

9.3 Аналіз ринкових можливостей запуску стартап–проекту

В межах даного підрозділу проводиться визначення ринкових можливостей, які можна використати під час ринкового впровадження проекту, та ринкових загроз, які можуть перешкодити реалізації проекту. Визначення ринкових можливостей дозволяє спланувати напрями розвитку із урахуванням стану ринкового середовища, потреб потенційних клієнтів та пропозицій проектів–конкурентів.

Таблиця 9.3 — Попередня характеристика потенційного ринку стартап–проекту

№	Показники стану ринку	Характеристика
1	Кількість головних гравців, од	1
2	Загальний обсяг продаж, ум. од.	Невідомий
3	Динаміка ринку	Зростає
4	Наявність обмежень для входу	Невідома
5	Специфічні вимоги до стандартизації та сертифікації	Існують
6	Середня норма рентабельності в галузі, %	Невідома

За результатами аналізу важко зробити висновок привабливості для входження за попереднім оцінюванням.

Визначимо потенційні групи клієнтів.

Таблиця 9.4 — Характеристика потенційних клієнтів стартап–проекту

№	Потреба що формує ринок	Цільова аудиторія	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару

Продовження таблиці 9.4

1	Програма Автоматизованого тестування лічильників електричної енергії	Науковці, інженери тестувальники, розробники радіоелектронної апаратури	Невідомі	Швидкість тестування, якість

Проведемо аналіз ринкового середовища: складемо таблиці факторів, що сприяють ринковому впровадженню проекту, та факторів, що перешкоджають.

Таблиця 9.5 — Фактори загроз

№	Фактор	Зміст загрози	Можлива реакція компанії
1	Новий функціонал ПЗ конкурентів	Оптимізована програма тестування. Введені функції по розрахунку часу взалежності від вхідних параметрів	Вихід з ринку

Таблиця 9.6 — Фактори можливостей

№	Фактор	Зміст можливості	Можлива реакція компанії
1	Новий функціонал що розробляється	Оптимізація коду програми для швидшого виконання тестів	Розроблення цього функціоналу

Проведемо аналіз пропозиції: визначимо загальні риси конкуренції на ринку.

Таблиця 9.7 — Ступеневий аналіз конкуренції на ринку

№	Особливості конкурентного середовища	В чому проявляється дана характеристика	В чому проявляється дана характеристика	Вплив на діяльність підприємства
1	Тип конкуренції — монополістична	Одне підприємство майже зайняло усю нішу	Одне підприємство майже зайняло усю нішу	Значний

Продовження таблиці 9.7

2	За рівнем конкурентної боротьби — національне	Дане підприємство відомо в рамках України	Дане підприємство відомо в рамках України	Значний
3	За галузевою ознакою — внутрішньогалузева	Конкуренція виконується в рамках однієї галузі	Конкуренція виконується в рамках однієї галузі	Значний
4	Конкуренція за видами товарів — невідомо			
5	За характером конкурентних переваг — цінова	Товар даного підприємства має дуже високу вартість	Товар даного підприємства має дуже високу вартість	Значний
6	За інтенсивністю — невідомо			

Проведемо більш детальний аналіз умов конкуренції у галузі.

Таблиця 9.8 — аналіз конкуренції в галузі за М. Портером

Складові аналізу	Прямі конкуренти в галузі	Потенці конкуренти	Постачальники	Клієнти	Товари – замінники
	«Энергия – 9»	Автоматизація тестування	Невідомо	Невідомо	Невідомо
Висновки	Маючи майже монопольне положення на ринку розробник цього ПЗ не буде приділяти уваги розробці	Є можливість виходу на ринок	Невідомо	Невідомо	Невідомо

За результатами аналізу можна зробити висновок, що працювати на даному ринку можна незважаючи на конкурентну ситуацію. Для поширення продукту він повинен володіти рядом факторів, які відрізняють від існуючого конкурентна.

Перелічимо фактори конкурентноспроможності.

Таблиця 9.9 — Обґрунтування факторів конкурентноспроможності

№	Фактор конкуренто-спроможності	Обґрунтування
1	Простота	Дана розробка вимагає лише базові знання з електроніки
2	Дешевизна	Поширюється безкоштовно і кожний має можливість користуватися нею
3	Швидкодія	Розраховуються найкращі показники для конкретного проекту

Проведемо аналіз сильних та слабких сторін стартап–проекту.

Таблиця 9.10 — Порівняльний аналіз сильних та слабких сторін проекту

№	Фактор конкуренто-спроможності	Бали 1–20	Рейтинг товарів — конкурентів у порівнянні з проектом, що розробляються						
			-3	-2	-1	0	+1	+2	+3
1	Простота								
2	Дешивизна								
3	Швидкодія								

Проведемо SWOT–аналіз

Таблиця 9.11 — SWOT–аналіз стартап–проекту

Сильні сторони: Простота Дешевизна Швидкодія	Слабкі сторони: Невідома компанія Відсутність стартового капіталу
Можливості: Розширення функціоналу Нові технології	Загрози: Продукти–замінники

З огляду на SWOT–аналіз можна прийти до висновку що нема потреби розробляти альтернативи впровадження цього проекту.

9.4 Розроблення ринкової стратегії та маркетингової програми проекту

Розроблення ринкової стратегії першим кроком передбачає визначення стратегії охоплення ринку, а саме опис цільових груп потенційних споживачів.

Таблиця 9.12 — Вибір цільових груп потенціальних споживачів

№	Опис профілю цільової групи потеційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтований попит в межах цільової групи	Інтенсивність конкуренції в сегменті	Простота входу у сегмент
1	Науковці	Готові	Високий	У сегменті значна конкуренція	Важко
2	Інженери автоматизованого тестування	Готові	Високий	У сегменті не значна конкуренція	Важко
Які цільові групи обрано: науковці, інженери автоматизованого тестування					

Для роботи в обраних сегментах ринку сформуємо базову стратегію розвитку.

Таблиця 9.13 — Визначення базової стратегії розвитку

№	Стратегія охоплення ринку	Ключові конкурентоспроможності позиції	Базова стратегія ринку
1	Диференційний маркетинг	Простота, дешевизна, швидкодія	Стратегія спеціалізація

Виберемо конкурентну поведінку.

Таблиця 9.14 — Визначення базової стратегії конкурентної поведінки

№	Чи є проект «першопроходцем» на ринку?	Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?	Чи буде компанія копіювати основні характеристи товару конкуренту?	Стратегія конкурентної поведінки
1	Так	Ні	Ні	Заняття конкурентної ніші

Розробимо стратегію позиціювання, що полягає у формуванні ринкової позиції, за яким споживачі мають ідентифікувати проект.

Таблиця 9.15 — Визначення стратегії позиціювання

№	Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключі конкурентоспроможності власного стартап–проекту	Вибір асоціацій, які можуть сформувати комплексну позицію власного проекту
1	Точність			

Сформуємо маркетингову концепцію товару, який отримає споживач.

Таблиця 9.16 — Визначення ключових переваг концепції потенційного товару

№	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами
1	Автоматизація тестування лічильників електричної енергії	Зменшення часу на тестування	Швидкодія, точність

Таблиця 9.17 — Опис трьох рівнів моделі товару

Рівні товару	Сутність та складові
1) Товар за задумом	Перевірка сучасного лічильника на коректність накопичення енергії
2) Товар у реальному виконанні	Властивості: 1) Простота 2) Дешевизна 3) Швидкодія
	Якість: тестування готових моделей
	Пакування: відсутнє
	Марка: відсутня
3) Товар із підкріпленням	До продажу: невідомо
	Після продажу: невідомо

Товар не буде захищатись від копіювання та буде поширюватись.

Визначимо цінові межі, якими необхідно керуватись при встановленні ціни на товар.

Таблиця 9.18 — Визначення меж встановлення ціни

№	Рівень цін на товар–замінники	Рівень цін на товари–аналоги	Рівень доходів цільової групи споживачів	Верхня та нижня межі встановлення ціни на товар
1	70–100 тис. ум. од.	До 10 тис. ум. од.	Високий	Безкоштовно

Визначимо оптимальну систему збуту

Таблиця 9.19 — Формування систем збуту

№	Специфіка закупівельної поведінки цільових клієнтів	Функції збуту, які має виконувати постачальник товару	Глибина каналу збуту	Оптимальна система збуту
1	Невідома	Вільний доступ дотовару	Невідома	Вільний доступ до товару

Розробимо концепцію маркетингових комунікацій

Таблиця 9.20 — Концепція маркетингових комунікацій

№	Специфіка поведінки цільових клієнтів	Канали комунікацій, якими користуються клієнти	Ключові позиції, обрані для позиціювання	Завдання рекламного повідомлення	Концепція рекламного звернення
1	Невідома	Інтернет, наукові публікації	Можливості проекту	Донести про можливості проекту	Донесення про можливості та сильні сторони проекту

За результатом проведеного аналізу можна зробити висновок, що є можливість ринкової комерсалізації проекту оскільки на ринку є попит на таку продукцію.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Лічильник електричної енергії НІК 2104 багатотарифний [Електронний ресурс]. — Режим доступу: https://axiomplus.com.ua/pub/pdf/5424425-pasport.pdf — Назва з екрану.
2. Базові поняття програмної інженерії [Електронний ресурс]. — Режим доступу: https://web.posibnyky.vntu.edu.ua/fksa/4bevz_proektuvannya_programnyh_zasobiv_system_upravlinnya/2.htm — Назва з екрану.
3. Спиральная модель [Електронний ресурс]. — Режим доступу: https://qalight.com.ua/baza-znaniy/spiralnaya-model-spiral-model/ — Назва з екрану.
4. Тестирование программного обеспечения [Електронний ресурс]. — Режим доступу: https://careers.epam.by/content/dam/epam/by/book_epam_by/Software_Testing_Basics_2_izdanie.pdf — Назва з екрану.
5. Тестирование в модели жизненного цикла разработки ПО [Електронний ресурс]. — Режим доступу: https://webhamster.ru/mytetrashare/index/mtb0/1410498177o1eh16nhi4 — Назва з екрану.
6. Ручное тестирование [Електронний ресурс]. — Режим доступу: https://ru.wikipedia.org/wiki/Ручное_тестирование — Назва з екрану.
7. Основы тестирования программного обеспечения [Електронний ресурс]. — Режим доступу: https://www.intuit.ru/studies/courses/48/48/lecture/1442?page=1 — Назва з екрану.
8. Преобразователь интерфейсов USB – оптопорт [Електронний ресурс]. — Режим доступу: https://robo.market/offer/z-4817283 — Назва з екрану.

9. Трехфазная калибровочная установка PTS 400.3 [Електронний ресурс]. — Режим доступу: https://www.evm.ua/obraztsovoe-oborudovanie/poverochnye-sistemy/trexfaznaja-kalibrovochnaja-ustanovka-pts-400-3 — Назва з екрану.
10. 6003A Трехфазный калибратор электрической мощности Електронний ресурс]. — Режим доступу: https://docplayer.ru/72860847-6003a-trehfaznyy-kalibrator-elektricheskoy-moshchnosti.html — Назва з екрану.
11. THREE PHASE POWER CALIBRATOR AND POWER ENGINEERING APPARATUS TESTER type C300B Електронний ресурс]. — Режим доступу: https://drive.google.com/file/d/14cJspVh8d5RiDFpL2UOnS0LrBA64dqfA/view?usp=sharing — Назва з екрану.
12. ЛІЧИЛЬНИКИ ЕЛЕКТРИЧНОЇ ЕНЕРГІЇ НІК 2104 [Електронний ресурс]. — Режим доступу: http://www.nik.net.ua/uploads/керівництво_з_експлуатації_НІК2104_тарифний_(8U3).pdf — Назва з екрану.
13. C300B Transmission protocol [Електронний ресурс]. — Режим доступу: https://drive.google.com/file/d/1UjpBL3JV_lbMPAZDi9ECwW4wG0-4wEUf/view — Назва з екрану.
14. ПРОТОКОЛ HDLC [Електронний ресурс]. — Режим доступу: https://lektsii.org/1-77158.html — Назва з екрану.
15. JSON [Електронний ресурс]. — Режим доступу: https://uk.wikipedia.org/wiki/JSON — Назва з екрану.

ВИСНОВКИ

Розроблена автором програма автоматизованого тестування лічильників електричної енергії значно скоротила час на розробку та тестування лічильників. Ручне тестування потребує 6400 секунд для тестування одного параметру. Запропоноване програмне забезпечення виконує те саме тестування за 64 секунди, що в свою чергу скорочує час на тестування у 10 разів.

Запропонована програма підвищує якість тестування, так як з першого запуску дає точні результати. Аналогічний вид тестування виконує дану задачу не точно, оскільки причиною виникнення суб'єктивної похибки є людський фактор. Тому задача виконується мінімум п'ять разів.

Програма зручна у використані. Для її запуску обов'язковими є лише два параметра: Com–порт тестувальної калібровки та Com–порт лучильника

Дану програму використовує компанія «НІК–ЕЛЕКТРОНІКА» для тестування сучасних лічильників електричної енергії.

ДОДАТОК А

ПОГОДЖЕНО

Керівник дипломного проекту

доц. С.Б. Тарабаров

_____ _____

(дата) (підпис)

ЗАТВЕРДЖЕНО

Завідувач кафедри радіоконструювання та виробництва радіоапаратури НТУУ «КПІ ім. І. Сікорського»

_____ _____ Нелін Є. А.

(дата) (підпис)

д.т.н., професор

ТЕХНІЧНЕ ЗАВДАННЯ

НА МАГІСТЕРСЬКУ ДИСЕРТАЦІЮ

на тему: «Автоматизація тестування лічильників електричної енергії»

1 ПІДСТАВА ДЛЯ ВИКОНАННЯ РОБОТИ

Підставою для виконання роботи є завдання на магістерську дисертацію

2 МЕТА І ПРИЗНАЧЕННЯ

Мета роботи: дослідження особливостей автоматизованого тестування сучасних лічильників електроенергії..

Об’єкт дослідження: програма автоматизованого контролю електричної енергії в сучасних лічильниках

Предмет дослідження: автоматизоване тестування параметрів лічильників електричної енергії та дослідження їх похибок

3 ВИХІДНІ ДАНІ ДЛЯ ПРОВЕДЕННЯ РОБОТИ

Основні джерела:

1. C300B Transmission protocol [Електронний ресурс]. — Режим доступу: https://drive.google.com/file/d/1UjpBL3JV_lbMPAZDi9ECwW4wG0-4wEUf/view — Назва з екрану.
2. Three phase power calibrator and power engineering apparatus tester type C300B Електронний ресурс]. — Режим доступу: https://drive.google.com/file/d/14cJspVh8d5RiDFpL2UOnS0LrBA64dqfA/view?usp=sharing — Назва з екрану.
3. ЛІЧИЛЬНИКИ ЕЛЕКТРИЧНОЇ ЕНЕРГІЇ НІК 2104 [Електронний ресурс]. — Режим доступу: http://www.nik.net.ua/uploads/керівництво_з_експлуатації_НІК2104_тарифний_(8U3).pdf — Назва з екрану.
4. Основы тестирования программного обеспечения [Електронний ресурс]. — Режим доступу: https://www.intuit.ru/studies/courses/48/48/lecture/1442?page=1 — Назва з екрану.

4 ВИМОГИ ДО ВИКОНАННЯ

1. Ознайомитись з документацією сучасних лічильників електричної енергії керованих мікроконтролером
2. Оглянути ринок калібрувальних установок для лічильників
3. Проаналізувати протокол обміну даними між калібрувальною установкою та лічильником електричної енергії, що тестується
4. Розробити алгоритм автоматизованого тестування лічильників
5. Розробити драйвер для комунікації зовнішнього програмного забезпечення з установкою калібрування лічильників електричної енергії
6. Розробити програму автоматизованого тестування для контролю електричної енергії в сучасних лічильниках
7. Проаналізувати результати

5 ЕТАПИ ТА ТЕРМІН ВИКОНАННЯ

Таблиця 1 — Етапи дипломної роботи та терміни їх виконання

№	Назва етапів виконання магістерської дисертації	Термін виконання
1	Підбір джерел за темою магістерської дисертації	12.2018
2	Розробка ТЗ	04.2018
3	Ознайомлення з технічною документацією лічильників електричної енергії	05.2019
4	Аналіз калібрувальної установки	09.2019
5	Вибір програмного пакета та інтегрованого середовища для моделювання	10.2019
6	Моделювання методу в програмному середовищі	11.2019
7	Оформлення роботи	12.2019
8	Підготовка до захисту та отримання допуску до захисту	12.2019

6 ОЧІКУВАНІ РЕЗУЛЬТАТИ ТА ПОРЯДОК РЕАЛІЗАЦІЇ

6.1 Розробка програми автоматизації тестування сучасних лічильників електронної енергії

6.2 Аналіз результатів автоматизації тестування лічильника

7 МАТЕРІАЛИ ЯКІ ПОДАЮТЬСЯ ПІСЛЯ ЗАНІЧЕННЯ ДИСЕРТАЦІЇ

7.1 Завдання на магістерську дисертацію.

7.2 Технічне завдання.

7.3 Пояснювальна записка.

7.4 Презентація.

8 ПОРЯДОК ПРИЙМАННЯ ДИСЕРТАЦІЇ ТА ЇЇ ЕТАПІВ

8.1 Поетапне узгодження з керівником.

8.2 Представлення кафедрі.

8.3 Попередній захист магістерської дисертації.

8.4 Захист дисертації перед екзаменаційною комісією.

9 ВИМОГИ ДО РОЗРОБЛЕННЯ ДОКУМЕНТАЦІЇ

9.1 ДСТУ 3008-2015. Інформація та документація. Звіти у сфері науки і техніки. Структура та правила оформлення.

9.2 ДСТУ 3973-2000. Система розроблення та поставлення продукції на виробництво. Правила виконання науково-дослідних робіт. Загальні положення.

10 ОРІЄНТОВНИЙ ЗМІСТ ДИСЕРТАЦІЇ

10.1 Огляд документації.

10.2 Розробка алгоритму автоматизації тестування.

10.3 Результати апробації програми автоматизації тестування лічильників.

10.4 Аналіз результатів апробації програми автоматизації тестування.

Виконавець Шевчик Д. М.

ДОДАТОК Б. Лістинг драйверу

```
import time
EXPECTED_INIT_RESPONSE = b'C300 5.0.8 date 2016-03-22 S/N: 26410\r\n'
class CalmetException(Exception):
    pass

class Calmet:
    def __init__(self, driver):
        self.driver = driver
        self.autorization()

    @staticmethod
    def wait(seconds):
        return time.sleep(seconds)

    def get_response(self):
        self.wait(2)
        res = self.driver.readline()
        print(res)
        return res

    def autorization(self):
        """checks information from installation"""
        self.driver.write('VR_\r\n'.encode('ascii'))
        if self.get_response() != EXPECTED_INIT_RESPONSE:
            self.driver.close()
            raise CalmetException("Error during calmet init")

    def get_information(self):
        """checks information from installation"""
        self.driver.write('VR_\r\n'.encode('ascii'))
        print("write bytes...")
        self.get_response()

    def get_max_values_of_voltage(self):
        """getting max values of voltage form Calmet"""
        self.driver.write('GETMAXURNG_\r\n'.encode('ascii'))
        self.get_response()

    def read_voltage_and_current_info(self):
        """getting infromation about current voltages and currents form Calmet"""
        self.driver.write('ENDAMP_\r\n'.encode('ascii'))
```

```
            self.get_response()

        def read_angles_info(self):
            """read current degree in Calmet"""
            self.driver.write('ENDPHA_\r\n'.encode('ascii'))
            self.get_response()

        def read_impusles(self, input_value, register):
            self.driver.write(('RDMETS0_{},{}\r\n'.format(input_value,
register)).encode('ascii'))
            # self.get_response()
            self.wait(2)
            result = self.driver.readline()
            print(result)
            return result

        def read_amount_impulses(self):  # read current amount of impulses in real
time
            """read info about impluses by two channels"""
            self.driver.write(('RDMETS0ERR_\r\n'.encode('ascii')))
            self.get_response()
            self.wait(2)

        def read_angles_btwn_phases(self):
            """read angles between phases"""
            self.driver.write(('RPHAMEAS_\r\n'.encode('ascii')))
            self.get_response()
            self.wait(2)

        def save_previous_configuration(self):
            pass

        def set_stendby_or_operate(self,  phase1_u=1,  phase2_u=1,  phase3_u=1,
phase1_i=1, phase2_i=1, phase3_i=1):
            """ STB_1,1,1,1,1,1 if set 1 = OFF phase, if set 0  = ON PHASE """
            self.driver.write(
                'STB_{},{},{},{},{},{}\r\n'.format(phase1_u,   phase2_u,   phase3_u,
phase1_i, phase2_i, phase3_i).encode(
                    'ascii'))
            self.wait(3)

        def set_voltage_range(self, voltage_by_one_phase=0.0,
voltage_by_second_phase=0.0, voltage_by_third_phase=0.0):
            try:
```

```
                range_voltage = []
                currents = [voltage_by_one_phase, voltage_by_second_phase,
voltage_by_third_phase]
                for i in currents:
                    if 0.5 <= i <= 70:
                        range_voltage.append(1)
                    elif 1 <= i <= 140:
                        range_voltage.append(2)
                    elif 2 <= i <= 280:
                        range_voltage.append(3)
                    elif 5 <= i <= 560:
                        range_voltage.append(4)
                    else:
                        range_voltage.append(1)
                # print("Voltage ranges by every phase: {}".format(range_voltage))
                self.driver.write('RU_{},{},{}\r\n'.format
                                  (*range_voltage).encode('ascii'))
                self.get_response()
            except Exception as ex:
                print("Not correct set voltage range")
                raise CalmetException(ex)

        def set_current_range(self, current1=0.0, current2=0.0, current3=0.0):
            try:
                range_current = []
                currents = [current1, current2, current3]
                for i in currents:
                    if 0.005000 <= i <= 0.5:
                        range_current.append(1)
                    elif 0.05 <= i <= 6:
                        range_current.append(2)
                    elif 0.2 <= i <= 20:
                        range_current.append(3)
                    elif 1 <= i <= 120:
                        range_current.append(4)
                    else:
                        range_current.append(1)

self.driver.write('RI_{},{},{}\r\n'.format(*range_current).encode('ascii'))
                self.get_response()
            except Exception as ex:
                print("Not correct set current range")
                raise CalmetException(ex)
```

```
        def set_voltage(self, default1=0.5000, default2=0.5000, default3=0.5000):
            self.set_voltage_range(default1, default2, default3)
            self.driver.write(('U_{},{},{}\r\n'.format(default1, default2,
default3)).encode('ascii'))
            self.get_response()
            self.wait(3)

        def set_current(self, default1=0.001000, default2=0.001000,
default3=0.001000):
            self.set_current_range(default1, default2, default3)
            self.driver.write(('I_{},{},{}\r\n'.format(default1, default2,
default3)).encode('ascii'))
            self.get_response()
            self.wait(3)

        def set_frequency(self, f=50.000):
            """setting frequency
            :param: amount of frequency"""
            self.driver.write(('FR_{}\r\n'.format(f)).encode('ascii'))
            self.get_response()

        def set_angle(self, u1i1=0, u2i2=0, u3i3=0, u1u2=120, u1u3=-120):
            self.driver.write(('FA_{},{},{},{},{}\r\n'.format(u1i1, u2i2, u3i3, u1u2,
u1u3)).encode('ascii'))
            self.get_response()
            self.wait(3)

        def set_input_setting_impulses(self, input, register, amount_impulses):
            self.driver.write(('WRMETS0_{},{},{}\r\n'.format(input, register,
amount_impulses)).encode('ascii'))
            self.get_response()
            self.wait(2)
```

ДОДАТОК В. Лістинг функції тестування активної енергії та активної енергії мережі

```
@staticmethod
    def active_energy_summary(voltage=250, current=45):
        res = []
        cos_ = [-0.5, 0.5]
        try:
            for angle in cos_:
                val = Computation.cos_to_degree(angle)
            #встановлення зсуву фази
                calmet.set_angle(val, val, val)
            #перевірка активної енергії
                if angle >= 0:
            #вичитування даних до затримки
                    first_value_of_energy=
test.Get(SUMMARY_POSITIVE_ACTIVE_ENERGY_OBIS)
                    time.sleep(64 - DELTA_TIME)
            #вичитування даних після затримки
                    stored_energy=
test.Get(SUMMARY_POSITIVE_ACTIVE_ENERGY_OBIS)
                    delta_energy = stored_energy - first_value_of_energy
                      #перевірка з очікуваним значенням однофазних лічильників
                    if METER_TYPE[5] == '1':
                        act=
Computation.calculate_active_energy_by_one_phase(voltage, current, angle)
                        minus = Computation.deviation_measure_minus(act)
                        plus = Computation.deviation_measure_plus(act)
                        res.append(_check_correctness_data(minus,                plus,
delta_energy, "A+ registers"))
    #перевірка з очікуваним значенням отрьохофазних лічильників
                    if METER_TYPE[5] == '3':
                        act=
Computation.calculate_active_energy_by_three_phase(voltage, current, angle)
                        minus = Computation.deviation_measure_minus(act)
                        plus = Computation.deviation_measure_plus(act)
                        res.append(_check_correctness_data(minus,                plus,
delta_energy, "A+ registers"))
                    #перевірка активної енергії в мережі
                    if angle < 0:
    #встановлення зсуву фази
                     val = Computation.cos_to_degree(angle)
                    calmet.set_angle(val, val, val)
    #вичитування даних до затримки
```

```
                first_value_of_energy=
test.Get(SUMMARY_NEGATIVE_ACTIVE_ENERGY_OBIS)
                time.sleep(64 - DELTA_TIME)
    #вичитування даних після затримки
                stored_energy=
test.Get(SUMMARY_NEGATIVE_ACTIVE_ENERGY_OBIS)
                delta_energy = stored_energy - first_value_of_energy
    #перевірка з очікуваним значенням для однофазного лічильника
                if METER_TYPE[5] == '1':
                    act=
Computation.calculate_active_energy_by_one_phase(voltage, current, angle)
                    minus = abs(Computation.deviation_measure_minus(act))
                    plus = abs(Computation.deviation_measure_plus(act))
                    res.append(_check_correctness_data(minus,          plus,
abs(delta_energy), "A- registers"))
    #перевірка з очікуваним значенням для трьохфазного лічильника
                if METER_TYPE[5] == '3':
                    act=
Computation.calculate_active_energy_by_three_phase(voltage, current, angle)
                    minus = abs(Computation.deviation_measure_minus(act))
                    plus = abs(Computation.deviation_measure_plus(act))
                    res.append(_check_correctness_data(minus,          plus,
abs(delta_energy)))
            if len(res) == 2 and all(res):
                logging.info("OK")
                return True
            else:
                return False
        except Exception as ex:
            logging.info(RegisterEnergyError(ex))
```

ДОДАТОК Г. Лістинг функції тестування повної активної енергії та повної активної енергії мережі

```
@staticmethod
    def total_active_energy_summary(voltage=250, current=45):
        res = []
        cos_ = [-0.5, 0.5]
        try:
            for angle in cos_:
                val = Computation.cos_to_degree(angle)
#встановлення зсуву фази
                calmet.set_angle(val, val, val)
            # перевірка сумарної активної енергії
                if angle >= 0:
        #вичитування даних до затримки
                    first_value_of_energy=
test.Get(TOTAL_SUMMARY_ENERGY_OBIS)
                    time.sleep(64 - DELTA_TIME)
         #вичитування даних після затримки
                    stored_energy=
test.Get(TOTAL_SUMMARY_ENERGY_OBIS)
                    delta_energy = stored_energy - first_value_of_energy
        #перевірка з очікуваним значенням для трьохфазного лічильника
                    if METER_TYPE[5] == '3':
                        act=
Computation.calculate_active_energy_by_three_phase(voltage, current, angle)
                        minus = Computation.deviation_measure_minus(act)
                        plus = Computation.deviation_measure_plus(act)
                        res.append(_check_correctness_data(minus,
plus, delta_energy, "|A+| + |A-| registers"))
                    #перевірка з очікуваним значенням для однофазного лічиль-
ника
                    if METER_TYPE[5] == '1':
                        act=
Computation.calculate_active_energy_by_one_phase(voltage, current, angle)
                        minus = Computation.deviation_measure_minus(act)
                        plus = Computation.deviation_measure_plus(act)
                        res.append(_check_correctness_data(minus,
plus, delta_energy))
            # перевірка сумарної активної енергії в мережі
                if angle < 0:
                    first_value_of_energy=
test.Get(TOTAL_NEGATIVE_SUMMARY_ENERGY_OBIS)
                    time.sleep(64 - DELTA_TIME)
```

```
                    stored_energy=
test.Get(TOTAL_NEGATIVE_SUMMARY_ENERGY_OBIS)
                    delta_energy = stored_energy - first_value_of_energy
                #перевірка з очікуваним значенням для трьохфазного лічильника
                    if METER_TYPE[5] == '3':
                        act=
Computation.calculate_active_energy_by_three_phase(voltage, current, angle)
                        minus = abs(Computation.deviation_measure_minus(act))
                        plus = abs(Computation.deviation_measure_plus(act))
                        res.append(_check_correctness_data(minus, plus,
abs(delta_energy)))
                #перевірка з очікуваним значенням для однофазного лічильника
                    if METER_TYPE[5] == '1':
                        act=
Computation.calculate_active_energy_by_one_phase(voltage, current, angle)
                        minus = abs(Computation.deviation_measure_minus(act))
                        plus = abs(Computation.deviation_measure_plus(act))
                        res.append(_check_correctness_data(minus, plus,
abs(delta_energy), "|A+| - |A-| registers"))

            if len(res) == 2 and all(res):
                logging.info("OK")
                return True
            else:
                logging.info(res)
                return False

        except Exception as ex:
            logging.info(RegisterEnergyError(ex))
```

ДОДАТОК Д. Лістинг функції тестування повної енергії та повної енергії мережі

```
@staticmethod
    def full_active_energy_summary(voltage=250, current=45):
        res = []
        cos_ = [0.5, -0.5]
        try:
            for angle in cos_:
                val = Computation.cos_to_degree(angle)
  #встановлення зсуву фази
                calmet.set_angle(val, val, val)
               # перевірка повної енергії
                if angle >= 0:
                    # вичитка енергії до затримки
                    first_value_of_energy = test.Get(FULL_POSITIVE_ENERGY)
                    time.sleep(64 - DELTA_TIME)
                    # вичитка енергії після затримки
                    stored_energy = test.Get(FULL_POSITIVE_ENERGY)
                    delta_energy = stored_energy - first_value_of_energy
                    active_energy, reactive_energy = (voltage * current * angle) * 3,
(voltage * current * angle) * 3
                    calculated_full_energy                                    =
Computation.calculate_full_energy(active_energy, reactive_energy)
                    minus                                                     =
Computation.deviation_measure_minus(calculated_full_energy)
                    plus                                                      =
Computation.deviation_measure_plus(calculated_full_energy)
                    res.append(_check_correctness_data(minus,   plus,   delta_energy,
"S+ registers"))
                # перевірка повної енергії в мережі
                if angle < 0:
                    first_value_of_energy = test.Get(FULL_NEGATIVE_ENERGY)
                    time.sleep(64 - DELTA_TIME)
                    stored_energy = test.Get(FULL_NEGATIVE_ENERGY)
                    delta_energy = stored_energy - first_value_of_energy
                    active_and_reactive_energy = (voltage * current * angle) * 3
                    calculated_full_energy                                    =
Computation.calculate_full_energy(active_and_reactive_energy,
                                                                  active_and_reactive_energy)
                    minus                                                     =
Computation.deviation_measure_minus(calculated_full_energy)
                    plus                                                      =
Computation.deviation_measure_plus(calculated_full_energy)
```

```
                res.append(_check_correctness_data(minus, plus, delta_energy,
"S- registers"))

            if len(res) == 2 and all(res):
                logging.info('OK')
                return True
            else:
                return False

        except Exception as ex:
            logging.info(RegisterEnergyError(ex))
```

ДОДАТОК Е. Лістинг функції тестування реактивної енергії та реактивної енергії мережі

```
@staticmethod
    def reactive_energy_summary(voltage=250, current=45):
        res = []
        sin_ = [-0.5, 0.5]
        try:
            for angle in sin_:
        # встановлення куту
                val = Computation.sin_to_degree(angle)
                calmet.set_angle(val, val, val)
            #перевірка сумарної реактивної енергії
                if angle >= 0:
                    first_value_of_energy =
test.Get(SUMMARY_POSITIVE_REACTIVE_ENERGY_OBIS)
                    time.sleep(64 - DELTA_TIME)
                    stored_energy =
test.Get(SUMMARY_POSITIVE_REACTIVE_ENERGY_OBIS)
                    delta_energy = stored_energy - first_value_of_energy
    #перевірка з очікуваним значення трьохфазного лічильника
                    if METER_TYPE[5] == '3':
                        react = Computation.calc_react_ener_three_ph(voltage,
current, angle)
                        minus = Computation.deviation_measure_minus(react)
                        plus = Computation.deviation_measure_plus(react)
                        res.append(_check_correctness_data(minus, plus,
abs(delta_energy), " R+ registers"))
            #перевірка з очікуваним значення однофазного лічильника
                    if METER_TYPE[5] == '1':
                        react = Computation.calc_react_ener_one_ph(voltage, current,
angle)
                        minus = Computation.deviation_measure_minus(react)
                        plus = Computation.deviation_measure_plus(react)
                        res.append(_check_correctness_data(minus, plus,
abs(delta_energy), " R+ registers"))
            #перевірка сумарної реактивної енергії в мережі
                if angle < 0:
                    #  calmet.set_angle(Computation.sin_to_degree(angle),
Computation.sin_to_degree(angle),
                    #  Computation.sin_to_degree(angle))
            #вичитування накопленої енергії до затримки
                    first_value_of_energy =
test.Get(SUMMARY_NEGATIVE_REACTIVE_ENERGY_OBIS)
                    time.sleep(64 - DELTA_TIME)
```

```
        #вичитування накопленої енергії після затримки
                    stored_energy                                                    =
test.Get(SUMMARY_NEGATIVE_REACTIVE_ENERGY_OBIS)
                    delta_energy = stored_energy - first_value_of_energy
                #перевірка з очікуваним значення трьохфазного лічильника
                    if METER_TYPE[5] == '3':
                        react      =      Computation.calc_react_ener_three_ph(voltage,
current, angle)
                        minus = abs(Computation.deviation_measure_minus(react))
                        plus = abs(Computation.deviation_measure_plus(react))
                        res.append(_check_correctness_data(minus,                plus,
abs(delta_energy), " R- registers"))

                    #перевірка з очікуваним значення однофазного лічильника
        if METER_TYPE[5] == '1':
                        react = Computation.calc_react_ener_one_ph(voltage, current,
angle)
                        minus = abs(Computation.deviation_measure_minus(react))
                        plus = abs(Computation.deviation_measure_plus(react))
                        res.append(_check_correctness_data(minus,                plus,
abs(delta_energy), " R- registers"))

                if len(res) == 2 and all(res):
                    logging.info("OK")
                    return True
                else:
                    return False

            except Exception as ex:
                logging.info(RegisterEnergyError(ex))
```