МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ АЕРОНАВІГАЦІЇ, ЕЛЕКТРОНІКИ ТА ТЕЛЕКОМУНІКАЦІЙ
КАФЕДРА АЕРОКОСМІЧНИЇ СИСТЕМ УПРАВЛІННЯ

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач кафедри
_____Тачиніна О.М.
«___» _____2021р.

ДИПЛОМНА РОБОТА

(ПОЯСНЮВАЛЬНА ЗАПИСКА**)**
ВИПУСКНИКА ОСВІТНЬОГО СТУПЕНЯ
«БАКАЛАВР»

Тема: « Клієнт-серверна система автоматизованого контролю та відображення параметрів технічного стану паливної системи гвинтокрила.»

Виконавець: студент групи СУ-401Б Пасічний Петро Сергійович
(студент, група, прізвище, ім'я, по батькові)

Керівник: старший викладач Воронов Сергій Ігорович
(науковий ступінь, вчене звання, прізвище, ім'я, по батькові)

Нормоконтролер: _____ Дивнич М.П.
(підпис) (П.І.Б.)

Київ 2021

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Навчально-науковий інститут аеронавігації, електроніки та телекомунікацій

Кафедра аерокосмічних систем управління

Напрям (спеціальність, спеціалізація): 151 «Автоматизація та комп'ютерно-інтегровані технології»

(шифр, найменування)

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____Тачиніна О.М.

«__»_____2021р

ЗАВДАННЯ

на виконання дипломної роботи (проекту)

Пасічний Петро Сергійович

(П.І.Б. випускника)

1. Тема роботи: « Клієнт-серверна система автоматизованого контролю та відображення параметрів технічного стану паливної системи гвинтокрила.» затверджена наказом ректора від « » _____2021 р. №584/ст.
2. Термін виконання роботи: з 29.03.2021 по 15.06.2021
3. Вихідні дані роботи: складові частини паливної системи гвинтокрила Мі-8 і його модифікацій та завдання які вони вирішують.
4. Зміст пояснювальної записки: Переваги програмних способів реалізації систем управління. Опис проблем які виникають в паливній системі гвинтокрила та варіанти їх вирішення. Розробка системи автоматизованого контролю та відображення параметрів паливної системи гвинтокрила.
5. Перелік обов'язкового ілюстративного матеріалу: схема паливної системи гвинтокрила, схеми клієнт-серверної системи.

6. Календарний план-графік

№ п/п	Завдання	Термін виконання	Відмітка про виконання
1	Знайти літературу зв'язану з темою роботи.	1.04.2021-09.04.2021	
2	Визначити проблеми які виникають в паливній системі гвинтокрила.	10.04.2021-23.04.2021	
3	Знайти рішень проблем, що виникають в паливній системі.	24.04.2021-7.05.2021	
4	Запропонувати альтернативні рішення для вдосконалення паливної системи .	8.05.2021-14.05.2021	
5	Визначити датчики для використання клієнт-серверною системою.	15.05.2021-21.05.2021	
6	Розробити клієнт-серверну систему контролю та відображення інформації.	22.05.2021-4.06.2021	
7	Зробити висновки по пророблені роботі.	05.06.2021-9.06.2021	
8	Закінчити оформлення пояснювальної записки.	10.06.2021-13.06.2021	

7. Дата видачі завдання: «1» квітня 2021 р.

Керівник дипломної роботи (проекту): _____ Воронов С.І.

(підпис керівника) (П.І.Б.)

Завдання прийняв до виконання: _____ Пасічний П. С.

(підпис випускника) (П.І.Б.)

РЕФЕРАТ

Пояснювальна записка до дипломної роботи «Клієнт-серверна система автоматизованого контролю та відображення параметрів технічного стану паливної системи гвинтокрила.» 54 с., 21 рис., 1 табл., 9 літературних джерел.

Об’єкт дослідження: нові підходи до діагностування вузлів паливної системи гвинтокрила.

Предмет дослідження: способи діагностування вузлів паливної системи гвинтокрила.

Мета роботи: розробка клієнт-серверної системи автоматизованого контролю та відображення параметрів технічного стану паливної системи гвинтокрила.

Методи дослідження: аналіз існуючих рішень проблем паливної системи гвинтокрила з метою вияснення головних недоліків, перспективних вирішень окремих задач, пороговий та допусковий контроль параметрів діагностування.

Результатом виконання дипломної роботи є клієнт-серверна програма контролю та відображення параметрів паливної системи гвинтокрила, таких як кількість палива в баках паливної системи, інформація про справність насосів та положення електричних перекривних кранів.

ПАЛИВНА СИСТЕМА ГВИНТОКРИЛА, КЛІЄНТНТ СЕРВЕРНА СИСТЕМА, ІНФОРМАЦІЙНО-ВИМІРЮВАЛЬНА СИСТЕМА З ІМІТАТОРАМИ ДАТЧИКІВ.

ЗМІСТ

ВСТУП

На теперішній час більшість об’єктів і систем управління відносяться до класу розподілених об’єктів і систем, це означає, що є багато локальних підсистем в складі системи управління такими об’єктами і ці локальні підсистеми знаходяться на певній відстані одна від одної.

При більш детальному розгляді локальних підсистем бачимо, що вони поділяються на апаратні системи, основна модель управління яких реалізована за допомогою механіки, гідравліки, елекроніки, пневматики і їх комбінації та програмні системи, основна модель управління яких реалізована засобами цифрової обробки інформації. В більшості випадків, такими засобами є мікроконтролери або мікрокомп’ютери. Перетворення фізичних величин в числові і навпаки в таких випадках виконують функціональні блоки АЦП (аналого-цифровий перетворювач) та ЦАП (цифро-аналоговий перетворювач).[1]

В обох реалізаціях локальних підсистем якість виконання поставлених задач залежить від точності та швидкодії датчиків та регуляторів, а також від степені відповідності моделі поставленій задачі і швидкодії модулів управління.

Актуальність теми: Порівняємо апаратну та програмну реалізації по найважливішим характеристикам при використанні однакових стандартних датчиків.

Життєвий цикл системи

Оскільки в програмній реалізації вдосконалення системи в процесі експлуатації відбувається без зміни апаратного середовища вона на відміну від апаратної реалізації не потребує часу та грошей на закупку нових апаратних комплектуючих та створення чи виправлення макета.[1]

Програмна реалізація не потребує розробки конструкторської документації і виготовлення зразків системи.

Після налагодження програмну реалізацію можна легко розповсюдити по всім робочим приладам.

Ціна закупки мікрокомп'ютера набагато менше ціни закупки і розробки обладнання для апаратної реалізації системи управління при однакових функціональних можливостях.

Спостережуваність системи

Оскільки всі системи управління для захисту від зовнішніх впливів розміщуються в закритому корпусі, то основні можливості спостерігати реалізуються через інтерфейс зовнішнього спостереження. В наш час для такого інтерфейсу майже не використовуються прямі вимірювання пневматики чи гідравліки. Зазвичай використовують перетворювачі фізичної величини в електричний сигнал який поступає в електричний роз'єм.[1]

В апаратній реалізації кількість контрольних точок спостереження буде обмежено кількістю електричних роз'ємів. Програмна реалізація має значну перевагу за рахунок стандартних інтерфейсів мікрокомп'ютера (Ethernet, WiFi, Bluetooth,USB) через які можна надати доступ на рівні комірок оперативної чи вбудованої пам'яті.

Швидкодія системи

Для програмної реалізації час відгуку системи управління буде залежати від flops (кількості операцій з плаваючою комою в секунду яку виконує мікрокомп'ютер) та кількості операцій в алгоритмі управління, необхідних для розрахунку керуючого впливу. Оскільки час вимірювання і час розрахунку відгуку додаються, то для типової технічної системи, з часом перетворення АЦП/ЦАП 4MHz 0,25 мікросекунд та часом виконання алгоритму середньої складності 10 мікросекунд, повний час відгуку складає приблизно 10 мікросекунд, що відповідає частоті 100KHz.[1]

Елементами для апаратної реалізації типових технічних систем управління є мікросхеми операційних підсилювачів. Час відгуку типових операційних підсилювачів на зміну вхідного сигналу, зазвичай знаходиться в діапазоні від десяти до сотні наносекунд. При цьому для реалізації найпростішого алгоритму управління використовується послідовне каскадне включення, як правило, не менше двох операційних підсилювачів, що близько до часу відгуку найпростіших алгоритмів для програмної реалізації.

Отже програмна реалізація має переваги над апаратною і може використовуватися як платформа для побудови систем управління.

Наявність декількох USB-портів і порта HDMI дозволяє підключити до мікрокомп'ютера клавіатуру, маніпулятор та дисплей, перетворивши локальний регулятор в повнофункціональне середовище для розробки, налаштування та тестування програм в умовах повної взаємодії з реальним об'єктом. Наявність кількох ядер в процесорах мікрокомп'ютерів при певних налаштуваннях ОС дозволяє організувати кілька паралельних потоків виконання програм в режимі, максимально наближеному до реального часу.[1]

Кілька локальних систем управління в складі складного, розподіленого фізичного об'єкта, можна розглядати як самостійний об'єкт управління . Основним завданням управління таким об'єктом є завдання синхронізації роботи локальних систем для забезпечення максимально ефективного функціонування розподіленого фізичного об'єкта в цілому. Розв'язок такої задачі бере на себе система управління другого рівня.

Клієнт-серверна архітектура для систем управління

Для усунення проблем між прикладними програмами і мережевими протоколами організовується діалог і структура даних підсистем. Такі підсистеми називають клієнт-серверними підсистемами, а відповідні вузли мають назву сервери та клієнти. Сервери виконують організацію і забезпечення масового обслуговування, а клієнти мають можливість отримувати це обслуговування. Прикладні програми на серверах та клієнтах отримують можливість замість команд і

форматів представлення даних для протоколів TCP чи UDP, використовувати команди і формати даних спеціально розроблені в підсистемах клієнта і сервера, орієнтовані на прикладні потреби систем управління.

Локальні системи управління повинні виступати в ролі сервера, який надає системі управління другого рівня сервіси для спостереження і оптимізації режима свого функціонування.

Локальні системи в випадку втрати зв'язку з другим рівнем управління, можуть приймати на себе роль клієнта для отримання від інших локальних систем інформації про їх стан.

Система управління другого рівня взаємодіє не тільки з локальними системами управління але й з технічним персоналом. В цьому випадку вона повинна виступати в ролі сервера, здатного надавати клієнтам (консолі технічного персоналу) індивідуальні набори прав по спостереженню і управлінню.[1]

Мета і завдання виконання дипломної роботи: розробити клієнт-серверну систему автоматизованого контролю та відображення параметрів технічного стану паливної системи гвинтокрила.

Об'єкт дослідження: нові підходи до діагностування вузлів паливної системи гвинтокрила.

Предмет дослідження: способи діагностування вузлів паливної системи гвинтокрила.

РОЗДІЛ 1
ОПИС ПРОБЛЕМ ПАЛИВНОЇ СИСТЕМИ ГВИНТОКРИЛА ТА ЇХ ВИРІШЕННЯ

Для теоретичних досліджень був використаний метод аналізу існуючих рішень проблем паливної системи гвинтокрила з метою вияснення головних недоліків, перспективних вирішень окремих задач, а також оцінки необхідних значень вимірюваних і контрольованих величин.

1.1. Проблема консервації паливної системи при тривалій стоянці та при стоянці в умовах низьких температур.

Ця проблема існує оскільки можливе пошкодження в трубопроводах паливної системи та частинах двигуна. При тривалій стоянці в незаконсервованій системі можуть виникати корозії.

1.1.1 Традиційне рішення проблеми

Проблема вирішується шляхом зливання палива палива з усіх баків паливної системи через зливні крани наприклад 601100М. Зливний кран з витратного бака знаходиться справа між дванадцятим та тринадцятим шпангоутами. Зливні крани підвісних баків знаходяться в нижній частині підвісних баків. Зливний кран додаткових баків знаходиться зліва між третім і четвертим шпангоутами. Крани складаються з корпусу, клапану зі штоком, пружини, заглушки і гайки. Ручка крану встановлена на хвостовик штока. Кран відкривається при натисканні на ручку і виходи штифта з пазу гайки. [3]

Злив палива виконується через спеціальну трубку один кінець якої з’єднується зі зливним краном, а інший опускається в ємність для зливання палива. Для більш швидкого зливання палива з підвісних баків виконують перекачку палива з підвісних баків в витратний. Щоб це зробити включають бортові насоси підвісних та витратного баків. В такому випадку зливання палива через витратний бак виконується під тиском. Консервація агрегатів паливної системи вертольотів і двигунів виконується не пізніше як через 24 години після викачки палива.[4]

При зберіганні двигунів на гвинтокрилі від двадцяти до тридцяти днів виконується часткова консервація, а саме консервація паливної системи гвинтокрила.[2]

При консервації після зливу палива паливна система заповнюється спеціальним маслом.

1.1.2 Головний недолік традиційного рішення проблеми

Відсутність автоматичного контролю стану зливних кранів на випадок їх відкриття внаслідок дії вібрації та інших чинників.

1.1.3 Перспективне рішення проблеми

Встановлення на зливних кранах датчиків положення.

1.2. Проблема обмеженої максимальної відстані та часу польоту гвинтокрила.

Ця проблема виникає тому, що для виконання деяких видів завдань поставлених перед гвинтокрилом йому необхідно мати можливість знаходитись в польоті без дозаправки якомога довше.

1.2.1 Традиційне рішення проблеми

Для збільшення дальності і тривалості польоту на вертоліт можуть встановлюватися додаткові баки в вантажному відсіку або зовнішні додаткові паливні баки ВДБ.6130.000, що збільшують тривалість польоту на час від пять до шести годин та дальність польоту на відстань від тисячі до тисячі сімдесяти кілометрів.[5]

1.2.2 Головний недолік традиційного рішення проблеми

Встановлення додаткових баків збільшує масу гвинтокрила що призводить до зменшення максимального корисного завантаження гвинтокрила та у випадку заправлення надлишкової кількості палива до більших втрат палива за рахунок перевезення надлишкової маси пального, а отже й до додаткових економічних витрат.

1.2.3 Перспективне рішення проблеми

Встановлення датчиків кількості палива в додаткових баках та заправка гвинтокрила кількістю палива не більшою ніж максимальна кількість потрібна для виконання завдання .

1.3. Проблема підтримання оптимальної температури повітря в кабінах гвинтокрила.

Оскільки зі збільшенням висоти в тропосфері (верхня межа 8-18км) температура знижується приблизно на 6 градусів за шкалою Цельсія, а гвинтокрили в деяких випадках мають необхідність підніматися на висоту до 6км, то проблема

підтримання оптимальних температур повітря для пілотів та в деяких випадках для пасажирів є актуальною навіть в літній період.

1.3.1. Традиційне рішення проблеми

На гвинтокрилах які заправляються пальним Т-1, ТС-1, Т-2 для підтримання оптимальних температур використовується гасовий обігрівач КО-50 який рекомендується включати при температурах нижче 10 градусів за шкалою Цельсія.[6]

1.3.2 Головний недолік традиційного рішення проблеми

Використання гасового обігрівача зменшує кількість запасу пального в баках, що призводить до того що паливо витратиться раніше.

1.3.3 Перспективне рішення проблеми

Перед входом до гасового обігрівача на трубопроводі можна встановити витратомір для відстежування витрат палива та корегування плану польоту з врахуванням додаткових витрат палива.

1.4. Проблема фільтрації палива.

Не фільтроване паливо може може бути забруднене частинками фарби чи бруду які можуть попадати в паливний бак при заправці, а також частинками іржі яка з’являється під дією вологи в сталевих баках. Якщо ці частинки не видалити до того як паливо надійде в систему, вони призведуть до швидкого зношування і видоду з ладу паливних насосів та форсунок, через абразивний вплив частинок на високоточні компоненти, що використовуються в сучасних системах вприскування.

Також паливні фільтри підвищують потужність двигуна, тому що при зменшенні кількості забруднюючих речовин горіння палива відбувається ефективніше.**[7]**

1.4.1. Традиційне рішення проблеми

Для механічного очищення палива використовуються авіаційні фільтри ФЕ, ФЕС. Для відділення і видалення частинок вологи використовуються сепаруючі авіаційні фільтри ЕС. Для коагуляції мікрокрапель вільної вологи в паливі використовуються коагулюючі авіаційні фільтри ЕФК [8].

На гвинтокрилах мі-8 та його модифікаціях перед насосами-регуляторами двигунів встановлюється блок фільтрів. Він складається з корпусу, фільтра грубого очищення , фільтра тонкого очищення, перепускного хлипака, двох кранів 600М і 400М для злиття відстою палива, штуцерів входу і виходу палива. У внутрішніх колодязях корпусу встановлені фільтри грубого і тонкого очищення. У нижній частині в корпус блоку фільтрів вкручені зливні крани. У верхній частині корпусу встановлені штуцери входу і виходу палива. Фільтр грубої очистки складається з дискових фільтруючих елементів, зібраних на штоку, який штифтом через втулку і пружину з’єднаний з кришкою. Фільтруючий елемент складається з чотирьох сітчастих і однієї гофрованої шайб. Зовнішні сітчасті шайби є фільтруючими. Вони виготовлені з нікелевої сітки саржевого плетіння. Внутрішні сітчасті шайби є каркасними. Вони виготовлені з латунної сітки і призначені для додання жорсткості і збереження форми фільтруючих шайб. Внутрішні контури сітчастих шайб попарно завальцьовані в алюмінієві кільця. Між парами сітчастих шайб встановлюють гофровану шайбу, що надає жорсткість всьому фільтруючому елементу, після чого зовнішні контури всіх п'яти шайб елемента завальцьовується в алюмінієве кільце. Для запобігання протікання палива між фільтруючими елементами, їх встановлюють на шторку, щільно притиснутими один до одного пружиною через ущільнювальну шайбу. Зверху на штоку встановлена шайба з гумовим кільцем ущільнювача. Для забезпечення щільного стику між верхньою шайбою

фільтруючого пакета і корпусом блоку фільтрів та між корпусом блоку і кришкою фільтра шток фільтруючого пакету кріплять на кришці фільтра таким чином, що фільтруючий пакет може переміщатися уздовж осі штока внаслідок зміни довжини пружини, встановленої на штоку. Фільтр тонкого очищення затримує механічні частинки розміром більше дванадцяти мікрометрів. Фільтруючий елемент його складається з циліндричного каркаса і двох шарів гофрованих сіток. Фільтрація палива здійснюється зовнішньою гофрованою сіткою саржевого плетіння, виготовленої з нікелевого дроту. Внутрішня латунна сітка є каркасом і додає жорсткість гофрі зовнішньої сітки. Торці сіток і циліндра закладені у втулки. Гумові кільця ущільнюють з'єднання фільтруючого елементу з кришкою і корпусом. З корпусу блоку фільтрів фільтруючий елемент виймається разом з кришкою, яка притягається до корпусу траверсою. Паливо входить в корпус блоку фільтрів через вхідний штуцер, просочується всередину фільтруючого пакета фільтра грубої очистки і через канал між колодязями корпусу входить в порожнину фільтра тонкого очищення, а також підходить до запобіжного клапану. Просочуючись всередину фільтруючого елемента фільтра тонкого очищення, паливо по каналу в корпусі надходить до вихідного штуцера. При засміченні фільтру тонкого очищення перепад тиску при досягненні 0,05мегапаскаль відкривається перепускний клапан 5, і паливо минаючи фільтр тонкого очищення, через вихідний штуцер надходить до насоса двигуна.[4]

1.4.2. Головний недолік традиційного рішення проблеми

Якщо фільтр не буде регулярно змінюватися, він може стати причиною засмічення і перестати пропускати потік палива. **[7]**

1.4.3. Перспективне рішення проблеми

Для вчасного виявлення засмічення фільтрів можна встановити датчики тиску на виході з них.

1.5. Проблема безперебійної подачі палива з підвісних паливних баків

Під час польоту гвинтокрил змінює кут тангажу, а отже паливо в середині баків переміщується від передньої частини бака в задню і навпаки, що ускладнює подачу палива до витратного паливного бака. Втрата доступу до палива в підвісних баках призведе до необхідності аварійної посадки.

1.5.1. Традиційне рішення проблеми

Підвісні баки з'єднані між собою двома трубопроводами, що забезпечує рівномірне вироблення палива з лівого і правого баків, а також повне вироблення палива з підвісних баків при відмові одного з насосів. Для безперебійної подачі палива при різних кутах тангажу насоси розташовуються в протилежних сторонах підвісних баків.[4]

При відмові насосу для підвісного паливного бака сигналізатор тиску СД-29А повідомляє про припиненя подачі палива до витратного баку.[9]

1.5.2. Перспективне рішення проблеми

Інформацію від сигналізаторів тиску для більш комфортного сприйняття можна виводити на клієнт клієнт-серверної системи.

Висновки до розділу 1

В результаті проведених досліджень можна сформулювати технічне завдання на розробку клієнт-серверної системи в відповідності з архітектурою представленою на рисунку 1.1.,

Рис. 1.1. Клієнт – серверна архітектура для систем управління.

а також виділити перелік найбільш важливих вимірюваних величин котрі приведені в таблиці 1.1.

Табл. 1.1.

Датчики та вимірювані величини

Тип	Назва	Діапазон	Границі	Тип контроля
Сигналізатор тиску мембранного типу для насосів ЭЦН-40	СД-29А	Тиск спрацювання СД-29А від 0,15 кгс/см^2		При відмові насосів чи закінченні палива «0» При справності насосів та наявності палива «1»

Сигналізатор тиску мембранного типу для насосу ЭЦН-75 правого баку	СД-29А	Тиск спрацювання СД-29А від 0,15 кгс/см^2		При відмові насосів чи закінченні палива «0» При справності насосів та наявності палива «1»
Сигналізатор тиску мембранного типу для насосу ЭЦН-75 лівого баку	СД-29А	Тиск спрацювання СД-29А від 0,15 кгс/см^2		При відмові насосів чи закінченні палива «0» При справності насосів та наявності палива «1»
Електричний важільно-поплавковий паливомір для вимірювання сумарного запасу без врахування додаткових паливних баків	СКЭС-2027В	від 0 до 4000 л.	критичний залишок палива «Залишилося палива 300 л»	Круговий індикатор
Електричний важільно-поплавковий паливомір для вимірювання запасу палива в лівому підвісному баці	СКЭС-2027В	від 0 до 1000 л		Лінійний індикатор
Електричний важільно-поплавковий паливомір для вимірювання запасу палива в правому підвісному баці	СКЭС-2027В	від 0 до 1000 л		Лінійний індикатор
Електричний важільно-поплавковий паливомір для вимірювання запасу палива в витратному баці	СКЭС-2027В	Від 0 до 1030л		Лінійний індикатор
Електричний важільно-поплавковий паливомір для вимірювання запасу палива в додатковому баці	СКЭС-2027В	Від 0 до 915л		Круговой індикатор
перекривний електромагнітний кран	610200А			Відкритий 1, закритий 0
перекривний кран 768600МА трубопроводу подачі палива	768600МА			Відкритий 1, закритий 0

РОЗДІЛ 2
РОЗРОБКА СИСТЕМИ

2.1. Архітектура і поетапна розробка клієнт-серверної системи

2.1.1. Перший етап розробки

Архітектура програмного рішення клієнт-серверної системи для першого етапу розробки зображена на рисунку 2.1. і включає три підсистеми:

- Імітатор сервера, який за запитами імітатора каналу передачі даних формує пакети даних. До складу імітатора сервера входить імітатор вимірювальної підсистеми, який дозволяє програмісту в зручному візуальному режимі задавати довільні тестові значення.
- Імітатор каналу передачі даних, який імітує отримання даних клієнтом від сервера. При цьому такий імітатор повинен:
  - виконувати імітацію запитів клієнта в автоматичному (по таймеру) режимі;
  - отримувати від імітатора сервера пакети з даними і ініціювати їх обробку імітатором клієнта в підсистемі візуалізації.
- Імітатор клієнта, в який входить підсистема візуалізації.

На цьому етапі розробки основним завданням є розробка прикладної частини клієнта, тобто, підсистеми консолі візуалізації, яка призначена для відображення інформації в зручному для користувача ергономічному вигляді.

При розробці консолі візуалізації вирішуються наступні завдання:

- Розробка базового зображення (малюнка або креслення) технічної підсистеми.
- Розміщення на базовому зображенні індикаторів даних.
- Налаштування параметрів кожного індикатора.

Оскільки, підсистема візуалізації є досить складним програмним продуктом для її розробки необхідні допоміжні модулі, які будуть забезпечувати для її інтерфейсів середовище налагодження яке не відрізняються від середовища взаємодії з підсистемою клієнта до складу якої вона входить. В якості таких модулів використовуються готові рішення з бібліотеки кафедри.

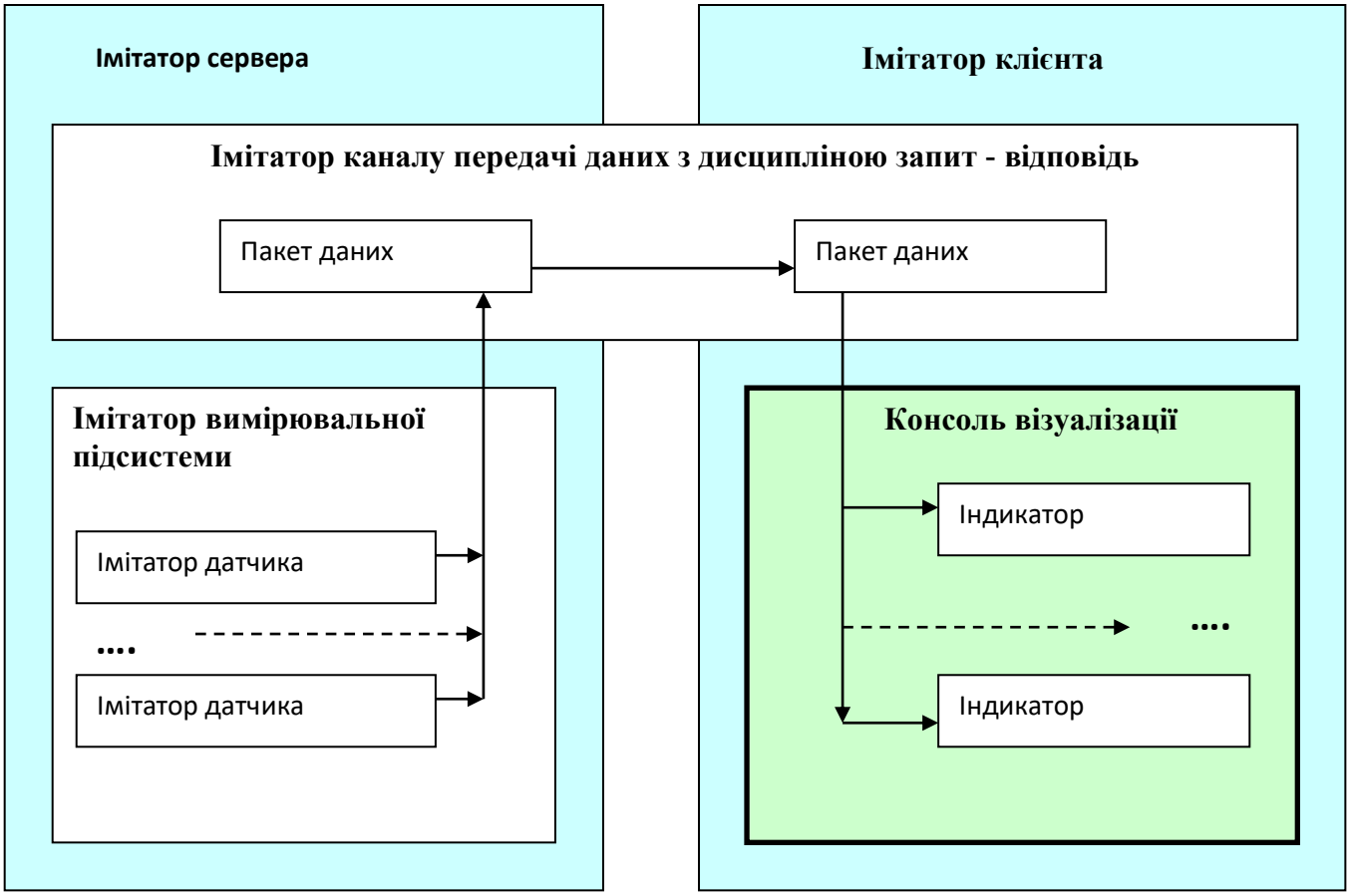

Рис. 2.1. Архітектура програмного рішення клієнт-серверної системи для першого етапу.

2.1.2 Другий етап розробки системи

Наступним етапом розробки клієнт-серверної системи для технічного персоналу, є інтеграція розробленої підсистеми візуалізації з підсистемами клієнта і сервера. Архітектура відповідної програмної реалізації зображена на рис. 2.2.

На цьому етапі виконується програмування і налагодження підсистем, які відповідальні за стійку взаємодію в реальній локальній комп'ютерній мережі між клієнтськими і серверними вузлами технічної системи без прямого впливу налагодження на реальні фізичні об'єкти. Це дуже важлива особливість, яка дозволяє безпечно моделювати різні ситуації як у фізичному каналі передачі даних, так і різні ситуації, пов'язані з помилками програмування.

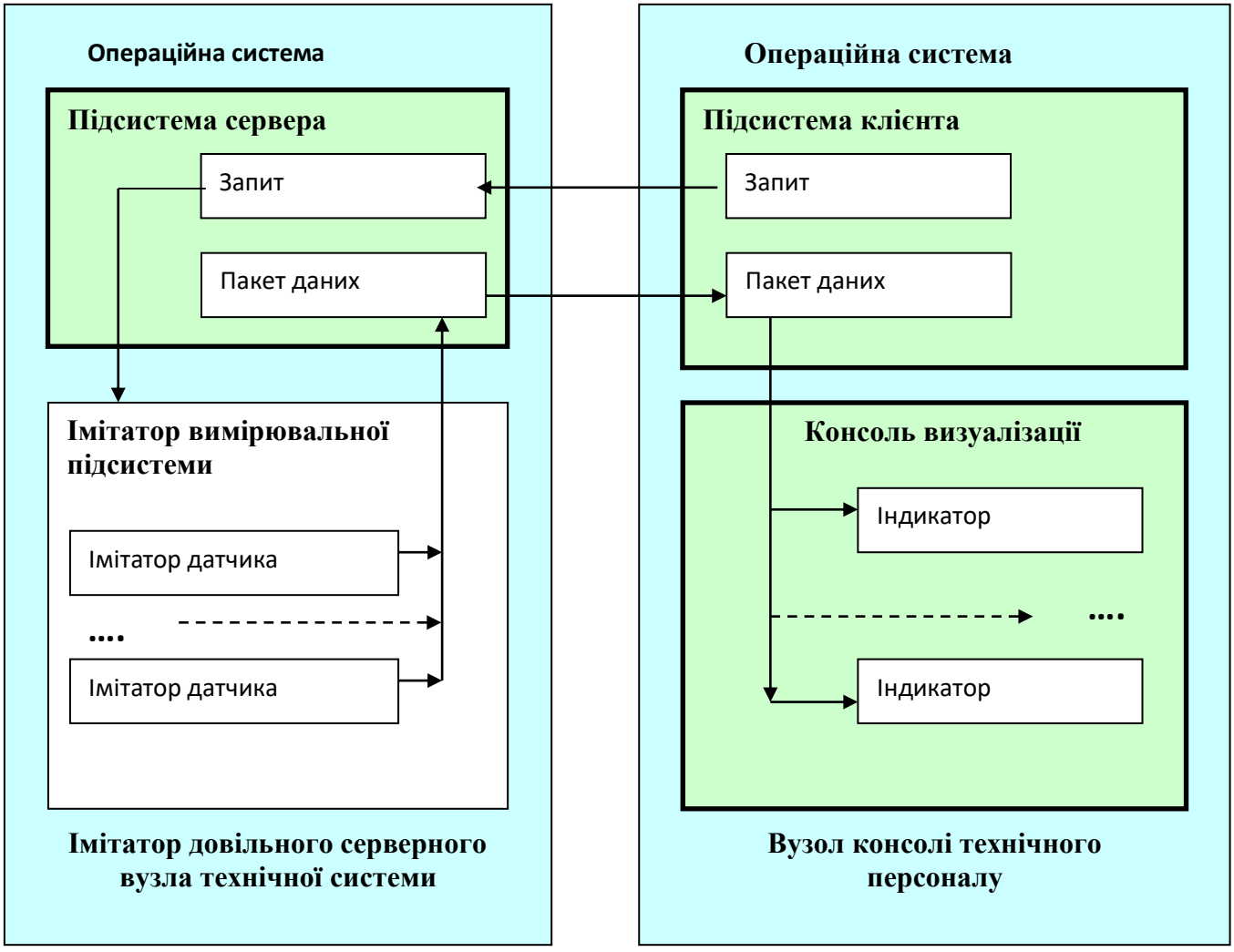

Рис .2.2. Клієнт-серверна архітектура інформаційно-вимірювальної системи з імітаторами датчиків.

2.2. Архітектура підсистем

2.2.1. Особливості блок - алгоритмічногопису програмних підсистем

Відповідно до завдання, реалізація розроблюваних програмних підсистем, повинна базуватися на Delphi технології програмування. У термінах технології програмування Delphi, створені програми називають прикладними програмами або просто додатками.

Важливо підкреслити, що будь-які додатки, створені в системі програмування Delphi, виконуються як деякі програми під управлінням деякого додаткового керуючого об'єкта. Такий керуючий об'єкт створюється системою Delphi автоматично і не вимагає від програміста написання ніяких додаткових програмних текстів. У цьому сенсі, систему Delphi можна розглядати як інструмент по створенню дворівневих програмних систем. Де до систем першого рівня відносять прикладні програми, а до системи другого рівня додатковий керуючий об'єкт. Слід також зазначити, що для різних додатків (прикладного або системного характеру) в Delphi пропонуються різні варіанти такого керуючого об'єкта.

У нашому випадку в якості керуючого об'єкта використовується об'єкт Application, орієнтований на побудову додатків з розвиненим діалогом, орієнтованим на кінцевого користувача.

Об'єкт Application першим отримує управління при запуску операційною системою деякого прикладного застосування. При цьому об'єкт Application:

- Створює і ініціює всі модулі в складі додатка, включаючи:
  - всі інструментальні модулі;
  - всі модулі з формами для взаємодії з кінцевими користувачами і всі компоненти на цих формах;
  - формує зв'язки між подіями, які заявлені на компонентах, і адресами процедур для обробки цих подій.

- Відстежує повідомлення операційної системи, і якщо ці повідомлення ставляться до подій, які задані на компонентах, ініціює виконання відповідних процедур (надалі обробників подій);
- Відстежує аварійні ситуації, які можуть виникнути при виконанні процедур обробки подій і робить можливі дії для збереження стійкої роботи програми або детально інформує про її аварійне завершення;
- Виконує штатне або аварійне завершення роботи програми та звільняє всі ресурси запитані цим додатком.

Оскільки виконання тих чи інших прикладних програм пов'язується з обробниками подій, а ті, в свою чергу, зв'язані з компонентами розміщеними на формах, то центральний алгоритм роботи всіх розроблюваних модулів як першого,

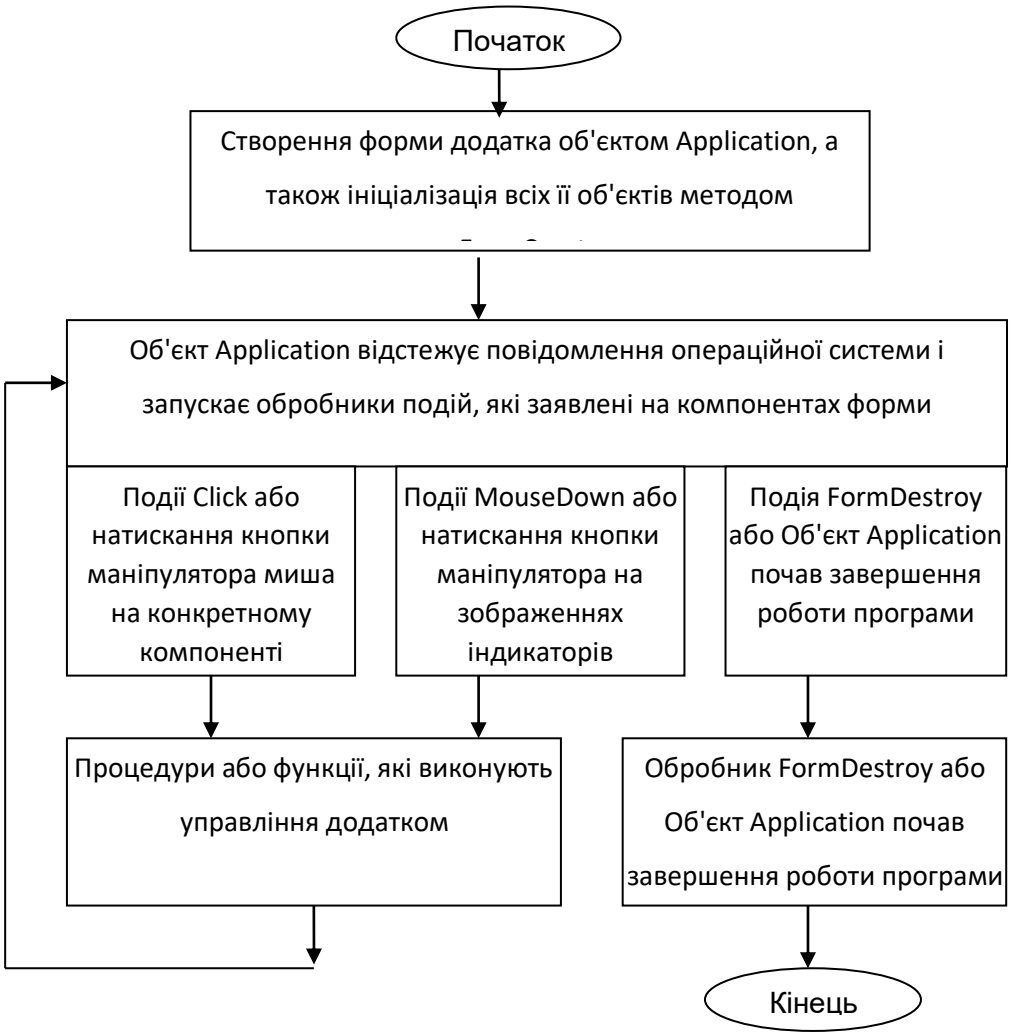

Рис. 2.3. Укрупнений алгоритм роботи Application - додатки Delphi

так і наступних етапів розробки, можна представити як блок - алгоритмічну схему, яка приведена на рисунку 2.3.

Оскільки ми плануємо в розробці системи застосовувати готові компоненти Delphi, то для реалізації клієнта і сервера використовуємо мережеві компоненти

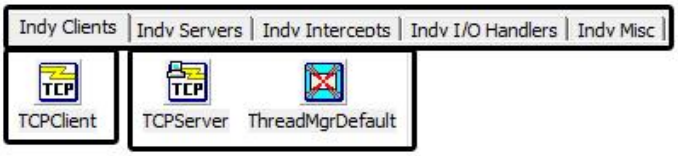

Рис. 2.4

ISO-рівня додатків, реалізовані в технології Indy (рис.2.4).

Використання названих компонентів розширює список подій, мережевими

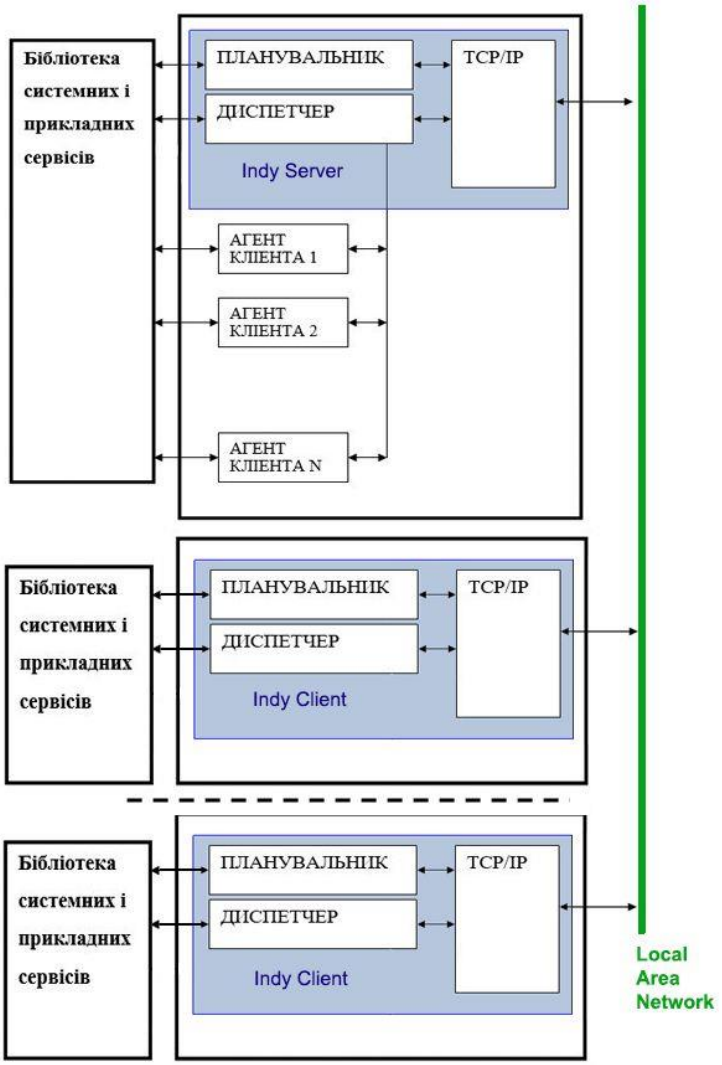

Рис. 2.5. Архітектура інформаційно - вимірювальної системи

подіями.

Відповідні обробники цих подій використовуються для планування (в нашому випадку це планування прикладного сеансового ISO-рівня), а також диспетчерування ресурсів для агентів клієнта на сервері і організації виконання прикладних або системних запитів від клієнтів.

Таким чином, архітектуру інформаційно - вимірювальної системи можна представити в наступному вигляді (рис. 2.5):

2.2.2. Архітектура підсистеми сервера

В рамках такої архітектури, при підключенні чергового клієнта до сервера планувальник - диспетчер сервера створює паралельний потік (нитка) приєднуючи до неї функції управління і блок даних.

У нашому випадку такий блок даних на сервері описується класом:

```
// -----------------------------
// Агент клієнта на сервері
type TAgent  = class(TObject)
// Нитка сервера, яка обслуговує клієнта
   AThread       : TIdPeerThread;
   IP            : string;         // IP - адреса клієнта
   Port          : integer;        // Port - клієнта
   Name          : string;         // Імя клієнта
   Password      : string;         // Пароль клієнта
   ClientMsg     : string;         // Повідомлення від клієнта серверу
   ServerMsg     : string;         // Повідомлення сервера клієнту
   LastIndx      : Integer;        // Індекс останній даних в DataBuf
   DataBuf       : TDataBuffer;    // Буфер відповіді від сервера
end;
```

Об'єкт, створений за цим класом, приєднується до виділеної клієнту нитки. Така нитка активується подією TCPServerExecute за запитом від конкретного клієнта. В обробнику подій TCPServerExecute виконуються наступні завдання:

```
procedure TServerForm.TCPServerExecute(AThread: TIdPeerThread);
begin
```

```
   // Отримати посилання на агента
   // Прочитати повний текст заявки від клієнта
   // Виділити в заявці системну командну частину
   // Зберегти повний текст заявки від клієнта
   // --------------------------
   // Запит від клієнта на підтвердження конекту
   // --------------------------
   // Запит від клієнта на дисконнект
   // ==========================
   // Запит від клієнта на отримання сервіса
      // Виконати запит від клієнта (ServiceDispatch)
         // Надіслати клієнту позитивну квитанцію
         // Надіслати клієнту результат обслуговування
         // АБО
         // Надіслати клієнту негативну квитанцію
   // ==========================
      // Зберегти останнє повідомлення клієнту від сервера
end;
```

Представлений перелік завдань, фактично описує планування прикладного сеансового ISO-рівня. Сукупність методів процедур або функцій, які викликаються позиціями цього плану можна розглядати як методи диспетчеризації. Основним диспетчируючим виконавцем в цій схемі, є функція:

```
// Диспетчер прикладних сервісів
function ServiceDispatch(var Request : TRequest) : boolean;
begin
   // Виконання планування та диспетчеризації прикладного обслуговування
end;
```

Іншими словами, функція ServiceDispatch є кореневою функцією виконання прикладної задачі.

На рисунку 2.6 представлена копія екрану сервера з двома підключеними клієнтами.

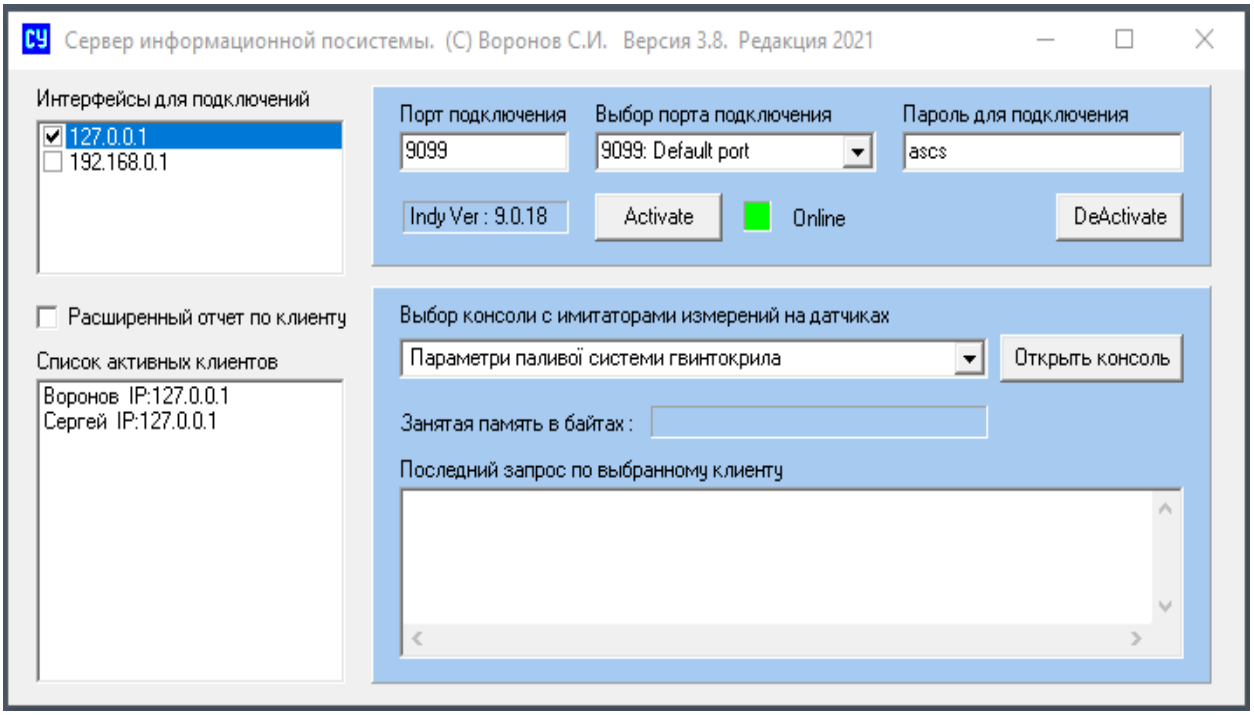

Рис. 2.6. Копія екрану сервера з двома підключеними клієнтами

За запитами клієнта, функція ServiceDispatch зчитує поточні значення з імітаторів датчиків, доступ до яких можна отримати через кнопку «Відкрити консоль». Копія екрану консолі «Імітатор датчиків» представлена на рис.2.7.

Сервер, при виборі конкретного технічного додатка і відкритті відповідної консолі, автоматично створює і розміщує на формі консолі всі об'єкти імітаторів датчиків, які описані в обраному технічному додатку. Об'єкт для будь-якого датчика описується класом, публічний сегмент якого має наступний вигляд:

```
// Класс сенсоров для объектов имитации (датчиков)
type TSensor = class(TObject)
public
  constructor Create(RqWinControl : TWinControl; XB,YB : integer);
  procedure Free;
  // ----------------------
  property Indx    : integer  read GetIndx    write SetIndx;
  property Title   : string   read GetTitle   write SetTitle;
  property Min     : integer  read GetMin     write SetMin;
  property Max     : integer  read GetMax     write SetMax;
  property RqNoise : boolean  read GetRqNoise write SetRqNoise;
  property Noise   : integer  read GetNoise   write SetNoise;
  property Value   : extended read GetValue   write SetValue;
  // ----------------------
```

```
  // Обработчик - Изменилось значение на сенсоре
  property OnSensor : TOnSensor read fOnSensor    write fOnSensor;
  // Присоединенный к сенсору индекс 1
  property LinkInd1 : integer   read fLinkInd1    write fLinkInd1;
  // Присоединенный к сенсору индекс 2
  property LinkInd2 : integer   read fLinkInd2    write fLinkInd2;
  // Присоединенный к сенсору объект
  property LinkOBJ  : TObject   read fLinkOBJ     write fLinkOBJ;
  // ----------------------
  // Установка цвета панели
  property Color : TColor       read fColor       write SetColor;
  // Видимость органов управления fTrackBar, fEdMin, fEdMax
  property CVisible   : boolean  read fCVisible   write SetCVisible;
  // Видимость органов управления шумом fChBoxNoise, fEdNoise
  property NVisible   : boolean  read fNVisible   write SetNVisible;
end;
```

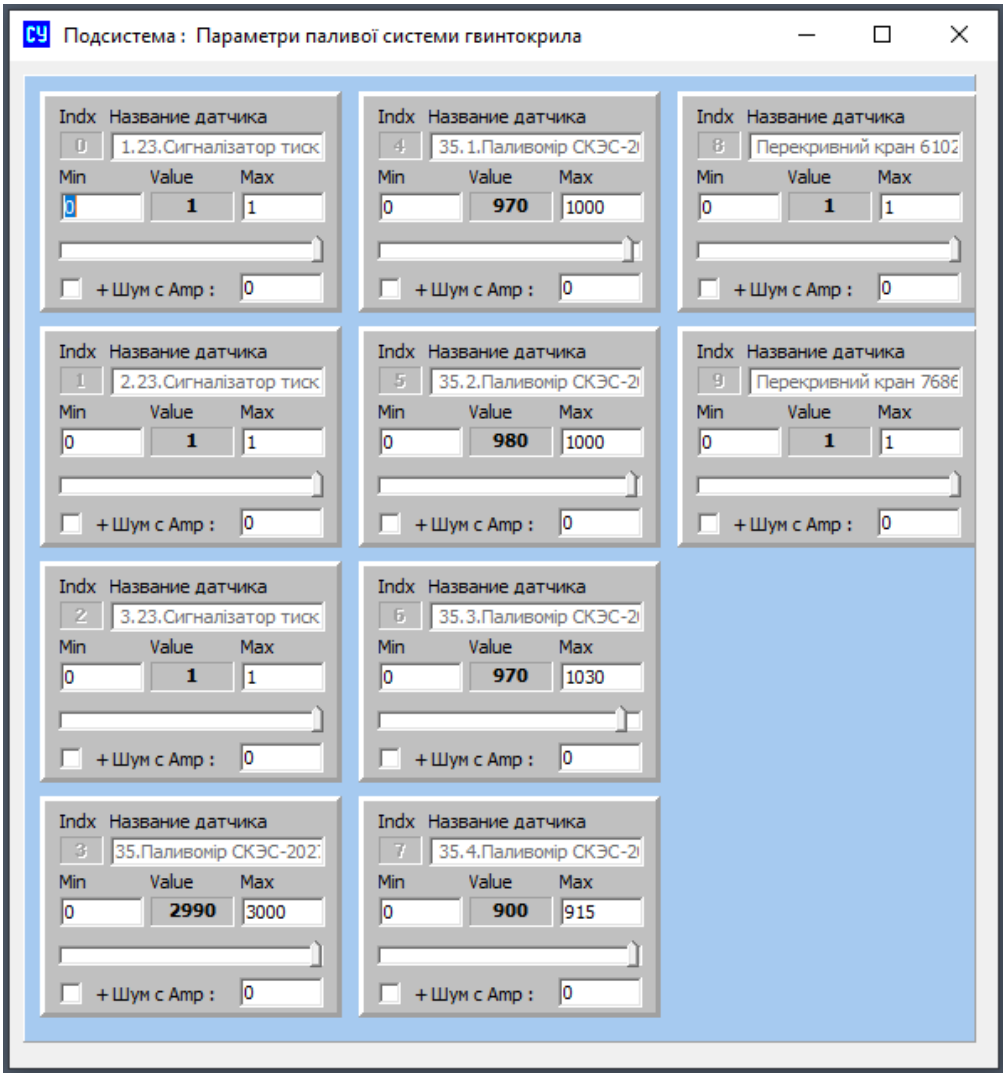

Рис. 2.7. Копія екрану консолі «Імітатор датчиків»

Консоль «Імітатор датчиків» підтримується бібліотечним інструментарієм, який входить до складу двох модулів:

- Sensor03 - Інструментальний клас для динамічного створення об'єктів імітації сенсорів (датчиків). Створення об'єкта сенсора реалізується тільки з прив'язкою до вже існуючого компоненту типу панель (TPanel) на формі додатка.
- SensorGroup03 - призначений для динамічного створення імітаторів датчиків (сенсорів), візуалізації таких імітаторів на формі консолі, а також є джерелом значень для консолі візуалізації даних клієнтом (ConsoleMAIN).

2.2.3. Архітектура підсистеми клієнта

Архітектурна реалізація клієнта передбачає в його мережевій частині використання одного паралельного процеса (нитки). Така нитка керується подіями, які задаються на компонентах основної форми клієнта, оброблювачем таких подій є метод RunRequest, що виконує наступні завдання:

```
// Виконати запит клієнта на обслуговування
procedure TClientForm.RunRequest(RqCmdInd : integer);
begin
    // ---------------------------
    // Скасування запитів до сервера, якщо додаток не завантажено
    // Контроль підключення
    // ---------------------------
    // Підготувати блок управління запитом до сервера
    // ---------------------------
    // Початкові установки
    // Сформуємо текст запиту до сервера
    // ---------------------------
    // Відправимо заявку на сервер
    // ---------------------------
    // Читаємо квитанцію за заявкою
```

```
        // Якщо сервер відповів позитивною квитанцією
        // ---------------------------
        // Читати результати обслуговування
        // Звіт про виконання команди
        // ---------------------------
end;
```

Після завершення роботи методу RunRequest управління передається в метод візуалізації даних в складі консолі візуалізації (ConsoleMAIN), який вирішує такі завдання:

```
// Показати нове значення на всіх індикаторах
procedure TConsoleForm.ShowSensors();
begin
   // -------------------
   // Вхідний контроль
   // -------------------
   // Показати всі датчики
   // -------------------
   // Додати точку і відобразити осцилограму
   // -------------------
   // Додати точку ALARM - панель
   // -------------------
   // Записати точку GRID - панель
   // -------------------
   // Якщо спектрометер включений
      // ---------------------------------
      // Додати точку в накопичувач вибірки для побудови спектра
      // По завершенню побудови вибірки відобразити спектр
   // -------------------
end;
```

2.2.4. Модульна структура клієнт-серверної підсистеми

Згідно з поетапним планом розробки розглянемо підсистему візуалізації, модульна структура якої представлена на рисунку 2.8.

Головний модуль підсистеми візуалізації ConsoleMAIN призначений для візуалізації необхідної кількості індикаторів заданого виду на зображенні конструкції або кресленні конкретного вузла технічної системи. Модуль надає

кінцевому користувачеві діалогову форму для управління особливостями візуалізації по кожному індикатору (завдяки обробці подій маніпулятора мишки на малюнку або кресленні технічного вузла).

При цьому, будь-який з індикаторів може підключатися до однопроменевого осцилографа, ALARM або GRID панелям, а також до спектрометру. Якщо використовуються спектральні вимірювання на обраному індикаторі, то спектр

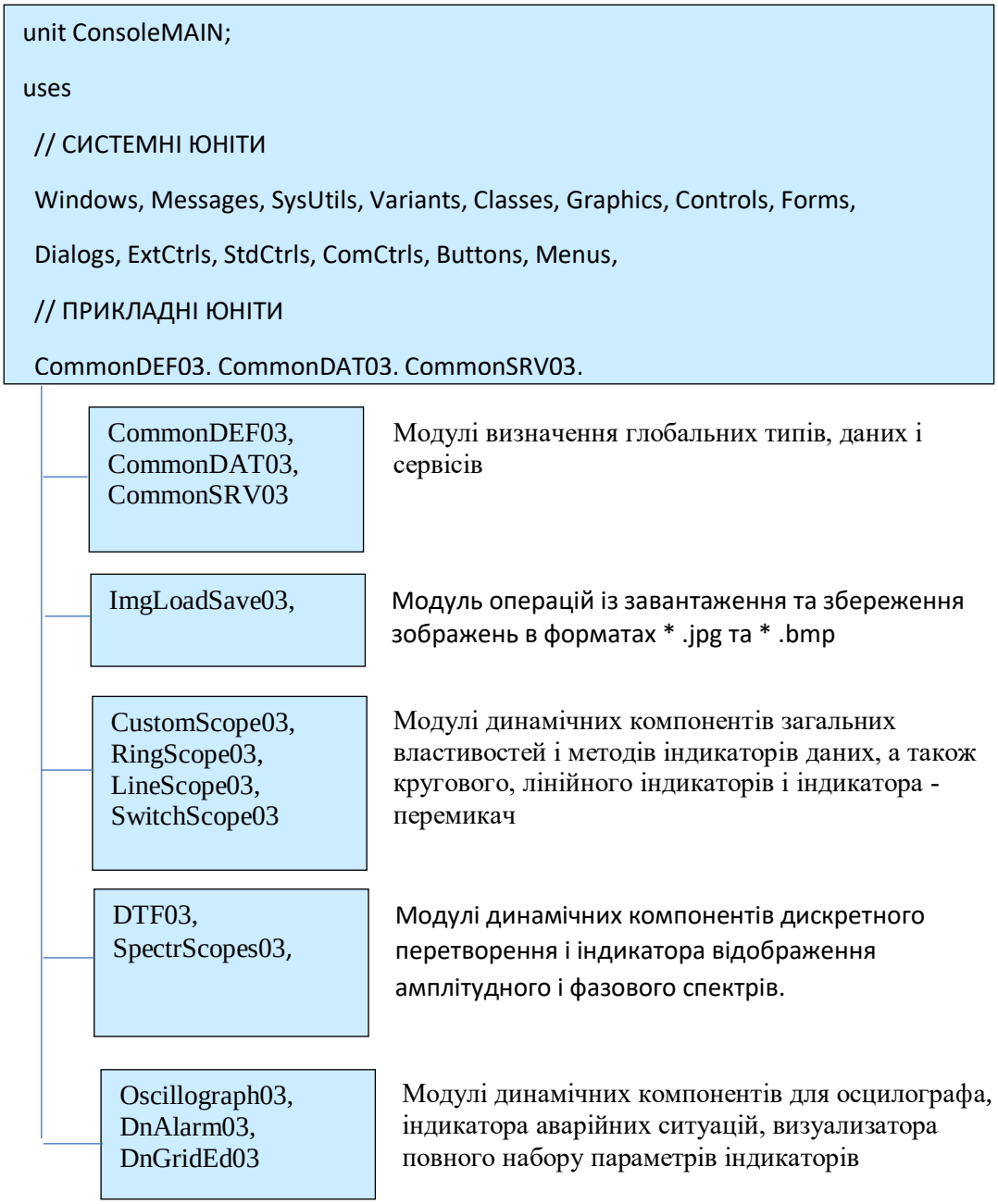

Рис. 2.8. Модульна структура підсистеми візуалізації

сигналів можна експортувати в Excel, а історію обраної на спектрі гармоніки можна відстежувати в вигляді осцилограми і записувати сигнал і (або) гармоніку в текстовий файл. Також, за допомогою осцилографа сигналу, можна записувати в текстовий файл відповідний вихідних сигнал.

У модулі ConsoleMAIN також виконується остаточна інтеграція (збірка) всіх засобів візуалізації та налаштування їх параметрів. Список основних сервісів, які використовує модуль ConsoleMAIN включає:

Лінійний індикатор миттєвих значень (модуль LineScope03; Клас TLineScope) Даний клас є прямим потомком базового класу TCustomScope для індикаторів миттєвих значень (див. Unit CustomScope03). Клас TLineScope призначений для візуалізації дійсних чисел (властивість Value) у формі текстового відображення і прямокутного графічного представлення. Позитивні значення відображаються в графічному поданні Value "з ліва на право" зеленим кольором, а негативні "з права на ліво" синім кольором. Індикатору можна задати колірні схеми і режими для допускового або порогового контролю значень, а також встановити режим цифрової фільтрації (тобто, індикацію усереднення заданого числа значень). Створення об'єкта індикатора виконується тільки з прив'язкою до вже існуючого компоненту TImage (орендованого Image).

Кільцевій індикатор миттєвих значень (модуль RingScope03, клас TRingScope) Даний клас є прямим потомком базового класу TCustomScope для індикаторів миттєвих значень (див. Unit CustomScope03). Клас TRingScope призначений для візуалізації дійсних чисел (властивість Value) у формі текстового відображення і кільцевого графічного представлення у вигляді дуги або сектора, розмір яких відповідає значенням Value. Позитивні значення відображаються на індикаторі зеленим кольором "проти годинникової стрілки", а негативні "за годинниковою стрілкою" синім кольором. Індикатору можна задати режими для допускового або порогового контролю значень, а також встановити режим цифрової фільтрації (тобто, індикацію усереднення заданого числа значень). Створення

об'єкта індикатора виконується тільки з прив'язкою до вже існуючого компоненту TImage (орендованого Image).

SWITCH - індикатор або індикатор індексів станів (модуль SwitchScope03, клас TSwitchScope). Даний клас є прямим потомком базового класу TCustomScope для індикаторів миттєвих значень (див. unit CustomScope03). Клас TSwitchScope призначений для візуалізації індексів різних станів, наприклад індексів станів дискретних об'єктів, кінцевих автоматів, індексів положення роторних перемикачів і т.п. Створення об'єкта індикатора виконується тільки з прив'язкою до вже існуючого компоненту TImage (орендованого Image).

Осцилограф (модуль Oscillograph03 ;, клас TOscillograph. Інструментальний клас для візуалізації результатів вимірювань у вигляді однопроменевий осцилограми. Клас TOscillograph для створення свого об'єкта допускає два варіанти:

- Оренда вже існуючого об'єкта типу панель (TPanel)
- Самостійне створення динамічної форми.

У будь-якому випадку клас самостійно виконує динамічне створення необхідних компонентів і виконує візуалізацію осцилограми. Для управління параметрами і режимами осцилографа використовується вбудоване в клас контекстне меню, що дозволяє керувати стилями осцилограми, а також здійснювати запис точок осцилограми в текстовий файл.

Панель відстеження і контролю значень (модуль DnAlarm03, клас TDnAlarm). Інструментальний клас для візуалізації результатів порогового або допускового контролю сигналів. Клас самостійно виконує динамічне створення форми, на якій розміщує необхідні компоненти і виконує візуалізацію лічильників перевищення порогів або виходу значень за допуск. Для управління параметрами і режимами панелі використовується вбудоване в клас контекстне меню.

Сітка на динамічній формі (модуль DnGridEd03, клас TDnGridEd). Інструментальний клас для візуалізації довільних текстових полів або одновимірних і двовимірних масивів. Динамічне створення несучої форми, а також візуалізація вхідних даних реалізується автоматично класом TDnGridEd.

2.2.5. Процедурна структура головного модуля клієнт-серверної підсистеми

Подання структури головного модуля підсистеми в формі переліку процедур і функцій, а також короткого опису виконуваних ними дій, можна розглядати як довідкову інформацію по запозиченому модулю, або як постановку задачі на їх програмування. У нашому випадку має місце обидва випадки. В даному підрозділі наводиться зведена таблиця основних процедур та функцій приватного і публічного сегментів головного модуля:

```
private
    // ---------------------------------
    // Режим використання границь для контролю поточних значень
    procedure SetCTRLs (RqSOBJ : TObject; RqCTRL : integer);
    // ---------------------------------
    // Установка SelectInd і параметрів панелей
    procedure SetSelectInd(Indx : integer);
    // ---------------------------------
    // Установка загальних параметрів індикаторів
    procedure SetCommonScopeParam (ScopeIndx : integer);
    // Установка параметрів в лінійний індикатор датчика
    procedure SetLineScopeParam (ScopeIndx : integer);
    // Установка параметрів в кільцевої індикатор
    procedure SetRingScopeParam (ScopeIndx : integer);
    // Установка параметрів в Switch індикатор
    procedure SetSwitchScopeParam (ScopeIndx : integer);
    // Установка параметрів в осцилограф сигналу
    procedure SetOscParam (RqOsc : TOscillograph; ScopeIndx : integer);
    // Установка параметрів сигналу в ALARM - панель
    procedure SetAlarmParam (RqAlarm : TDnAlarm; ScopeIndx : integer);
    // Установка параметрів в GRID - Панель
    procedure SetGRIDParam (RqGRID : TDnGridEd;  ScopeIndx : integer);
    // Установка параметрів в осцилограф гармонік в спектрі
    procedure SetGrmlOscParam (RqOsc     : TOscillograph;
                               ScopeIndx : integer;
                               GrmIndx   : integer);
    // ---------------------------------
    // Обробник події "Виконано завантаження масиву гармонік"
```

```
    procedure  SpectrLoad(Sender : TObject; ArrGrm : TArrSpectr);
    // Обробник події "Розпочато виконання методу Free"
    procedure SpecrtDestroy(Sender : TObject);
    // Оброблювач подій "Виконана відмітка гармоніки"
    procedure ShowSelectGrm(Sender: TObject;
                            Num : integer; ArrGrm : TArrSpectr);
    // Додати одну точку в масив накопичення вибірки для побудови спектра
    procedure AddSignalPoint(RqVal : extended;
                         var RqSpectrData : TSpectrData);
    // Скопіювати спектр у вхідний масив індикатора і відобразити спектр
    procedure ShowSpectrScope(var RqSpectrData : TSpectrData);
    // ---------------------------------
    procedure CreateSpectrograph();
    // Початкова ініціалізація вимірювача спектра і
    // осцилографа історії обраної гармоніки
    procedure InitSpectrograph(SelectInd : integer);

public
    // ---------------------------------
    // Показати нове значення на всіх індикаторах
    procedure ShowSensors();
    // Ініціювати консоль
    function OpenConsole(RqAppInd : integer) : integer;
    // Закрити консоль
    procedure CloseConsole();
```

2.2.6. Опис окремих процедур клієнт-серверної підсистеми

Приватна група:

Встановлення загальних параметрів для L, R, S - індикаторів

procedure TConsoleForm.SetCommonScopeParam (ScopeIndx: integer);

Опис:

Метод перевіряє існування об'єкта індикатора в масиві ArrSOBJ, який розташований в структурі даних адресованої AllPtScopeArr (масив покажчиків на додатки в unit CommonDAT03). Якщо об'єкт індикатора з індексом ScopeIndx існує, то виконується установка параметрів в цей об'єкт через групу батьківських

властивостей (unit CustomScope03). Джерелом параметрів є структура даних, яка адресується через покажчик PtAllConst в складі AllPtScopeArr.

Встановлення параметрів в лінійний Line - індикатор

procedure TConsoleForm.SetLineScopeParam (ScopeIndx: integer);

Опис:

Метод викликає метод SetCommonScopeParam і додатково встановлює індивідуальні параметри лінійного індикатора.

Встановлення параметрів в кільцевої Ring - індикатор

procedure TConsoleForm.SetRingScopeParam (ScopeIndx: integer);

Опис:

Метод викликає метод SetCommonScopeParam і додатково встановлює індивідуальні параметри лінійного індикатора.

Встановлення параметрів в Switch - індикатор

procedure TConsoleForm.SetSwitchScopeParam (ScopeIndx: integer);

Опис:

Метод SetSwitchScopeParam викликає метод SetCommonScopeParam і додатково встановлює індивідуальні параметри лінійного індикатора.

Встановлення параметрів в осцилограф сигналу

procedure TConsoleForm.SetOscParam (RqOsc: TOscillograph;
ScopeIndx: integer);

Опис:

Метод перевіряє існування об'єкта осцилографа (параметр RqOsc) і якщо об'єкт існує, то виконується установка параметрів осцилографа. Джерелом параметрів є структура даних, яка адресується через покажчик PtArrDESC в складі AllPtScopeArr (unit CommonDAT03).

Встановлення параметрів сигналу в ALARM - панель

procedure TConsoleForm.SetAlarmParam (RqAlarm: TDnAlarm;
ScopeIndx: integer);

Опис:

Метод перевіряє існування об'єкта ALARM - панель (параметр RqAlarm) і якщо об'єкт існує, то виконується установка параметрів інформаційної панелі ALARM. Джерелом параметрів є структура даних, яка адресується через покажчик PtArrDESC в складі AllPtScopeArr (unit CommonDAT03).

Встановлення параметрів сигналу в GRID - панель

procedure TConsoleForm. SetGRIDParam (RqGRID: TDnGridEd;
ScopeIndx: integer);

Опис:

Метод перевіряє існування об'єкта GRID - панель (параметр RqGRID) і якщо об'єкт існує, то виконується установка параметрів інформаційної панелі GRID. Джерелом параметрів є структура даних, яка адресується через покажчик PtArrDESC в складі AllPtScopeArr (unit CommonDAT03).

Встановлення параметрів в осцилограф історії гармонік в спектрі

procedure TConsoleForm.SetGrmlOscParam (RqOsc: TOscillograph;
ScopeIndx: integer;
GrmIndx: integer);

Опис:

Метод перевіряє існування об'єкта осцилограф (параметр RqOsc) і якщо об'єкт існує, то виконується установка параметрів. Джерелом параметрів є структура даних, яка адресується через покажчик PtArrDESC в складі AllPtScopeArr (unit CommonDAT03). Параметр GrmIndx задає індекс обраної в спектрі гармоніки.

<u>Публічна група:</u>

Видалити додаток з консолі

```
procedure TConsoleForm.CloseConsole ();
Опис:
procedure TConsoleForm.CloseConsole ();
begin
    // -------------------
    // Вхідний контроль
```

```
        // --------------------
        // Видалити об'єкти індикаторів
        // Видалити панелі і осцилограф сигналу
        // Видалити ALARM - панель
        // Видалити GRID - панель
        // Очистити екран спектрометра
        // Очистити осцилограф історії гормонікі
    end;
```

Створення і початкова ініціалізація параметрів всіх індикаторів обраного технічного додатка

function TConsoleForm.OpenConsole(RqAppInd : integer) : integer;

Опис:

```
function TConsoleForm.OpenConsole(RqAppInd : integer) : integer;
begin
   // Закрити консоль поточного додатка
   // --------------------
   // Вхідний контроль запиту додатка
   // Перевірка цілісності описувача додатка з індексом RqAppInd
   // --------------------
   // Завантаження програми в консоль
       // Завантажити зображення об'єкта
       // Очистити селектор вибору індикатора
       // Заповнити селектор вибору індикатора
       // Створити об'єкти зазначених індикаторів
       // Ініціювати параметри кожного індикатора
       // Встановити стартовий індекс обраного датчика
       // Показати найменування обраного датчика
       // Початкова ініціалізація вимірювача спектра
end;
```

Відобразити нові значення на всіх індикаторах

procedure TConsoleForm.ShowSensors();

Опис:

```
procedure TConsoleForm.ShowSensors();
begin
   // --------------------
```

```
    // Вхідний контроль
    // -------------------
    // Зафіксувати стартову тимчасову позначку
    // -------------------
    // Показати всі датчики
    // -------------------
    // Додати точку і відобразити осциллограмму
    // Додати точку ALARM - панель
    // Записати точку GRID - панель
    // Якщо спектрометер включений
       // ---------------------------------
       // Додати точку в накопичувач вибірки для побудови спектра
       // По завершенню побудови вибірки відобразити спектр
    // -------------------
    // Зафіксувати фінішну тимчасову позначку і відобразити інтервал
    // -------------------
end;
```

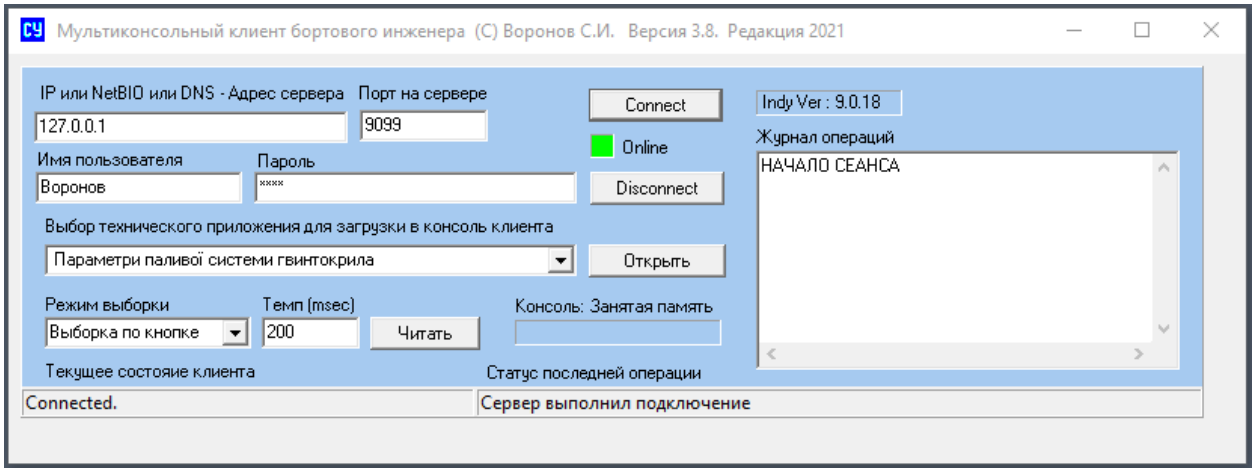

Рис. 2.9. Копія екрану консолі «Клієнта системи»

Функція TConsoleForm.OpenConsole (RqAppInd: integer): integer параметризується індексом технічного додатка який вона отримує по «Вибір технічного додатка для завантаження» і викликається за подією Click кнопки «Відкрити». Візуалізація даних виконується методом procedure TConsoleForm.ShowSensors() після успішного завершення заявки клієнта до сервера. Результат виконання заявки представлений на рис. 2.9

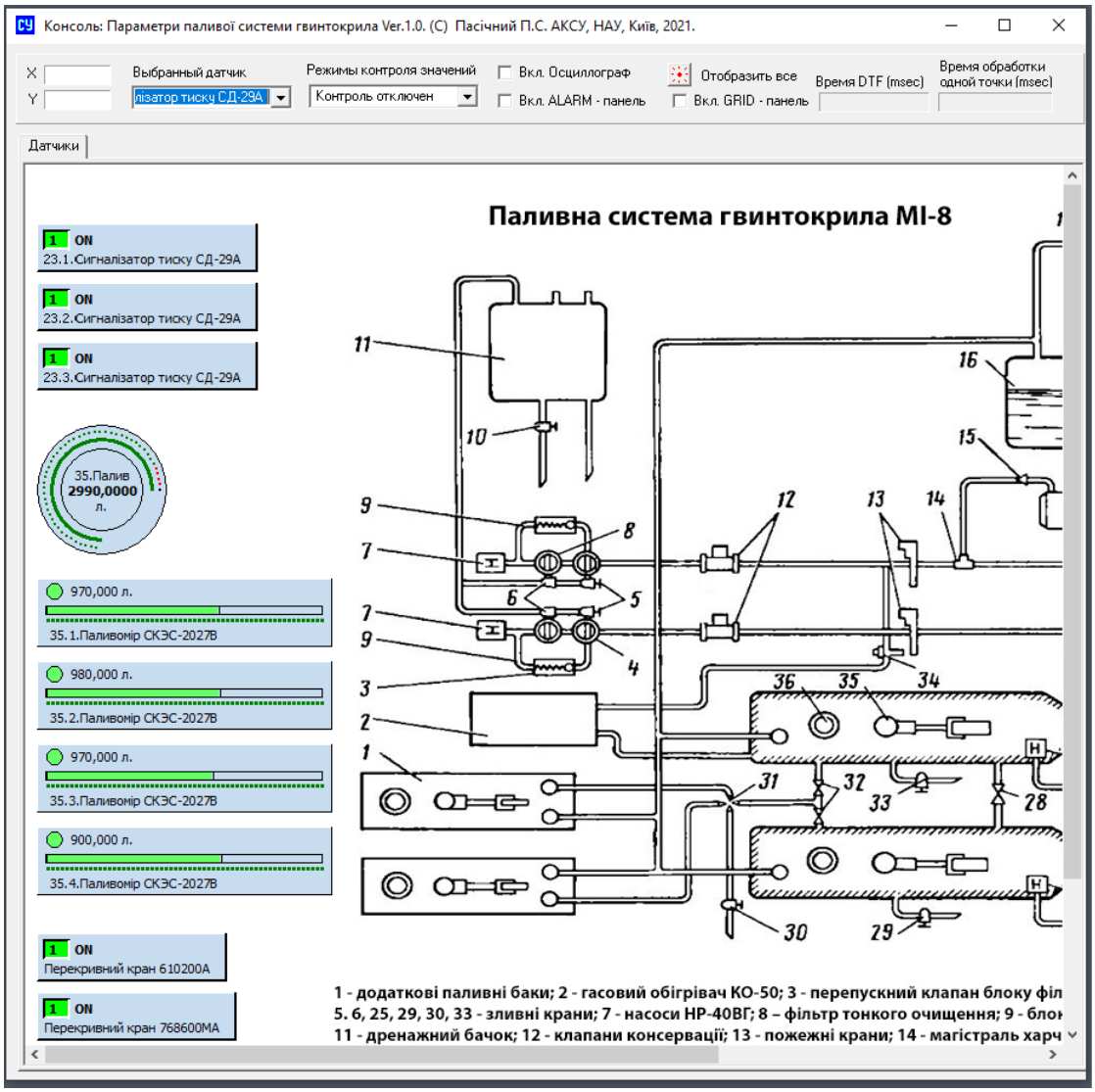

Рис. 2.10. Копія екрану консолі «Консолі візуалізації»

2.2.7. Структура представлення даних в системі

Оскільки комп'ютерні програми є засобом обробки даних, структура представлення таких даних в системі вирішальним чином впливає на архітектуру системи. У нашому випадку, структура даних та засоби їх ініціалізації визначаються в модулях CommonDEF03, CommonDAT03, CommonSRV03.

Модуль CommonDEF03. Даний модуль визначає основні типи представлення даних в системі. Ключовими типами є типи:

```
// ================================================================
// ОПИСУВАЧ ДЛЯ ВСІХ ДОДАТКІВ І ВІДПОВІДНИХ МАСИВІВ ІНДИКАТОРІВ
// ================================================================
```

```
// Індекс масиву - це індекс тех. додатка
type TAllPtScopeArr = array of TPtApp;
// ---------------------------------------------
// Описувач покажчика на заголовок додатка
type TPtApp = ^TApp;
// ---------------------------------------------
// Описувач заголовка додатка
type TApp = record
  PtAllConst  : TPtAllConst;
  MaxIndx     : integer;
  PtArrDESC   : pointer;
  ArrSOBJ     : TArrScopeOBJ;
end;
// ---------------------------------------------
// Опис заголовка додатка
type TAppConst = record
      // ------------------------------
      // Заголовок форми додатка
      AppType, AppName, AppVers, AppAutor, AppCopyR : string;
      // ------------------------------
      // Ім'я файла базової картинки (креслення)
      ImgFileName : string;
      ImgWidth    : integer;  // Ширина в пікселях
      ImgHeight   : integer;  // Висота в пікселях
      // ------------------------------
      // Піксельні габарити осцилографа сигналу
      Osc01Width  : integer;  // Ширина в пікселях
      Osc01Heigth : integer;  // Висота в пікселях
      // ------------------------------
      // Піксельні габарити ALARM - панелі
      AlarmWidth  : integer;  // Ширина в пікселях
      AlarmHeight : integer;  // Висота в пікселях
      // ------------------------------
      // Номер останньої гармоніки в обчислюваних спектрах
      MaxGrmNum   : integer;
      // ------------------------------
      // Число точок в масиві вибірки для MaxGrmNum
      // числа гармонік
      MaxSampNum  : integer;
end;
// ---------------------------------------------
```

```
// Описувач покажчика на запис з константами додатка
type TPtAllConst  = ^TAppConst;
// ---------------------------------------------
// Описувач масиву індикаторів в додатку
type TArrScopeOBJ = array of TObject;
```

Особливо необхідно звернути увагу на покажчик PtArrDESC в складі описувача заголовка програми. Після створення і ініціалізації системи даних, покажчики PtArrDESC всіх підключених технічних додатків будуть містити адреси статичних масивів - констант (типу TDescScope). Елеметами таких масивів є записи -константи типу TAppScope, в яких описуються параметри конкретних індикаторів:

```
// Описувач одного елемента масиву індикаторів
type TAppScope = record
     SType   : char;       // Тип індикатора
                           // L-лінійний R-кільцевої S-перемикач
     // --------------------
     Title   : string;     // Назва датчика
     Xb, YB  : integer;    // Піксельна прив'язка індикатора на екрані
     MinLX   : integer;    // Мін. ширина в пікселях (тільки для L-
                           //  індикаторів)
     // --------------------
     Meas    : string;     // Розмірність вимірюваної величини
     // ====================
     // ПОРОГОВИЙ КОНТРОЛЬ
     //  Позитивний піддіапазон Max > P2R > P1Y > 0
     //  Якщо P2R = P1Y = 0, то  пороговый контроль на
     //  піддіапазоні виключений
     Max     : extended;   // Кінець позитивного діапазону датчика
     P2R     : extended; // Початок позитивної червоної зони
     P1Y     : extended;   // Початок позитивної жовтої зони
     // --------------------
     // Негативний піддіапазон  Min < N1Y < N2R < 0
     //  Якщо N1Y = N2R = 0, то  пороговий контроль на
     //  піддіапазоні виключений
     N1Y     : extended;   // Початок негативною жовтої зони
     N2R     : extended;   // Початок негативною червоної зони
     Min     : extended;   // Кінець негативного діапазону датчика
```

```
    // ====================
    // ДОПУСКОВИЙ КОНТРОЛЬ
    // LB < LE у всьому діапазоні
    //  Якщо LB = LE = 0, то  допусковий контроль виключений
    LB      : extended;   // Початок допускового області
    LE      : extended;   // Кінець допускового області
    // --------------------
    WS      : TWallStyle; // Режим контролю значень
    // ====================
    Compact : boolean;    // Компактний вид відображення (тільки L-
                          // індикатор)
    Transp  : boolean;    // Прозорість
    // --------------------
    NumLPF  : byte;       // Довжина вибірки для цифрового фільтра
    Grm1P   : extended;   // Період першої гармоніки в (msec)
                          // Відключити аналіз спектра (-1 або 0)
    // --------------------
    Value   : extended;   // Стартове значення
    // --------------------
end;
```

Відповідні конкретному технічного додатка масиви константа з описами індикаторів кодуються в модулях AppDataXX, де XX позначає індекс технічного додатка.

```
// ============================================== ===============
// ОПИСУВАЧ ДЛЯ ВСІХ ДОДАТКІВ ВІДПОВІДНИХ МАСИВІВ ДАТЧИКІВ
// ============================================== ===============
// Індекс масиві - це індекс додатка
type TAllSensorArr = array of record
  AppName     : string;
  AppSensorArr: TAppSensorArr;
end;
// ------------------------------------------
// Описувач всіх датчиків однієї програми
// Індекс в масиві - це індекс датчика в додатку
type TAppSensorArr = array of THeadSensors;
// ------------------------------------------
// ОПИСУВАЧ ЗАГОЛОВКА СЕНСОРА
type THeadSensors = record
```

```
    // Опис датчика
    Title   : string; // Найменування датчика
    Min     : extended; // Мінімальне значення для датчика
    Max     : extended; // Максимальне значення для датчика
    Value   : extended; // Початкове значення для датчика
    RqNoise : boolean; // Запит підключення шуму
    Noise   : extended; // Амплітуда шуму
    // Вибірка для спекрографа
    SampInd : integer; // Поточний індекс в вибірці
    SampArr : TSampleArr; // Масив значень вибірки
end;
// ---------------------------------------------
// Описувач масиву значень вибірки
type TSampleArr = array of extended;
```

Модуль CommonDAT03. Даний модуль визначає основні змінні для представлення даних в системі. Ключовими змінними є змінні:

```
// ===============================================
// ВСІ ТЕХ. ДОДАТКИ КЛІЄНТА
// ===============================================
// Індекс масива - це індекс додатка
var AllPtScopeArr: TAllPtScopeArr;
```

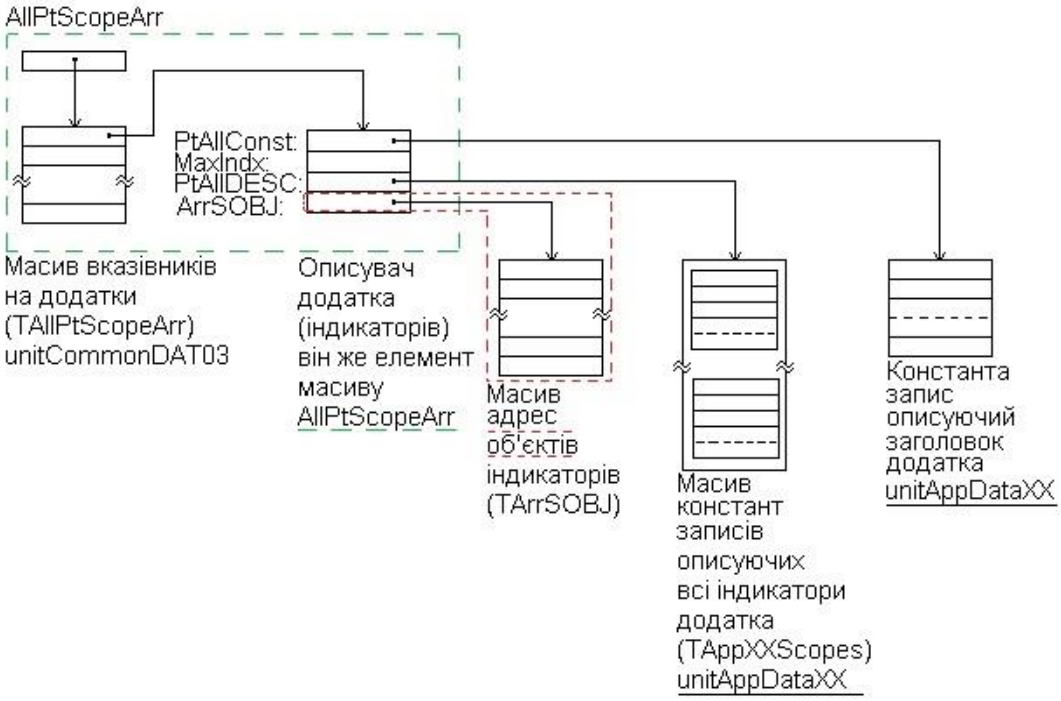

Рис. 2.11. Структура даних клієнта

```
// ============================================
// ВСІ ДАТЧИКИ СЕРВЕРА
// ============================================
// Індекс масива - це індекс додатка
var AllSensorArr: TAllSensorArr;
```

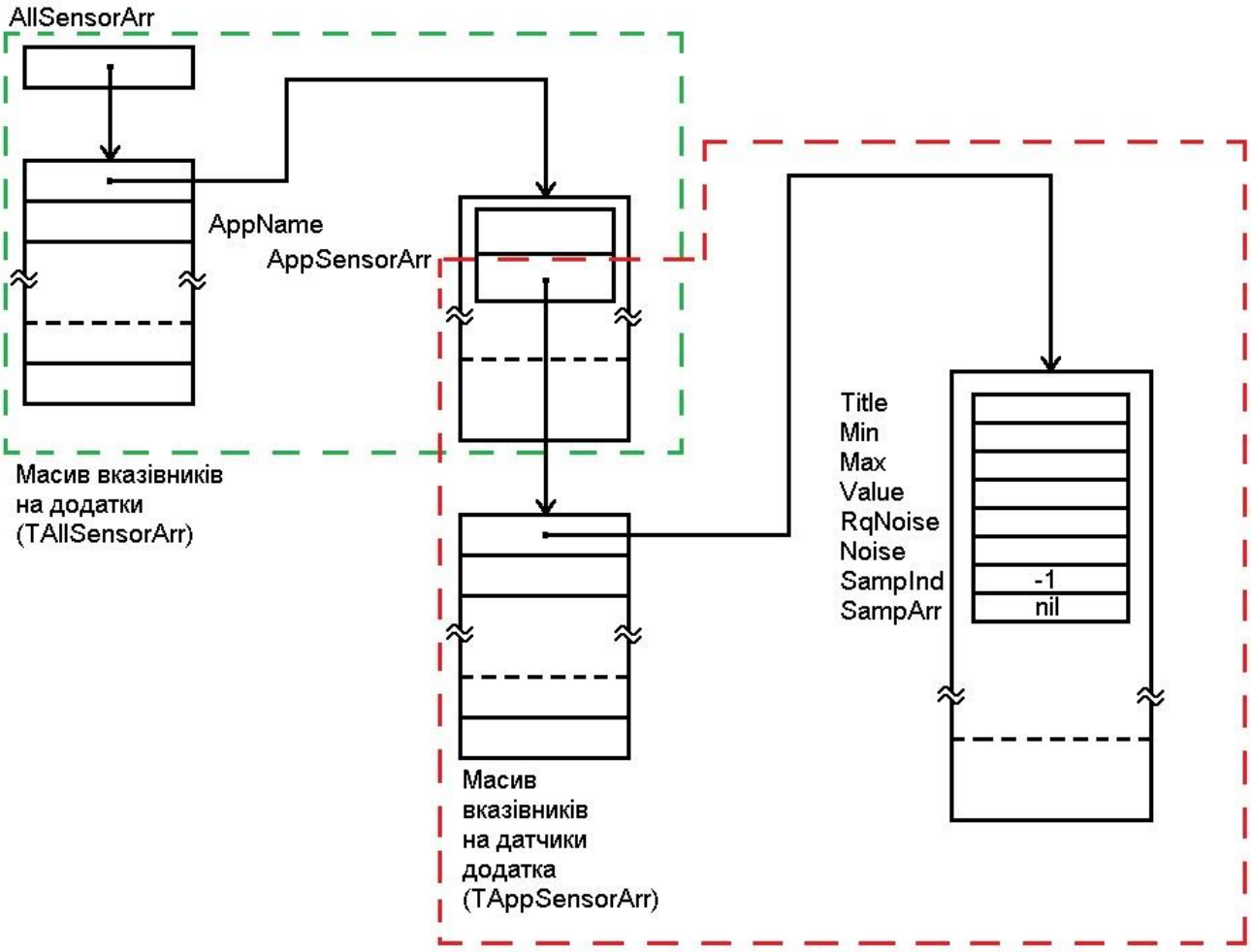

Рис. 2.12. Структура даних сервера

Названі змінні є корінними для структури представлення даних як для клієнта так і для сервера. Створення всієї структури даних, її ініціалізація і інструменти доступу до окремих елементів всередині структури виконують інструментарієм в складі модуля CommonSRV03.

Модуль CommonSRV03. Даний модуль містить основний інструментарій для роботи з даними як для сервера, так і для клієнта. Ключовою функцією модуля є функція:

```
// Загрузить описания всех индикаторов и сенсоров дя клиента и сервера
function LoadAllScopeAndSensorArr() : boolean;
```

Ця функція повністю створює структури даних актуалізуючи кореневі посилання AllPtScopeArr і AllSensorArr, а також ініціалізує всі вихідні дані всіх технічних додатків як для клієнта так і для сервера.

2.2.8. Порядок підключення нового технічного додатка

1) Підготувати модуль AppDataXX.pas, де XX-черговий номер модуля в папці AppData. Число індикаторів в AppDataXX.pas не повинно перевищувати значення, визначене в константі BufferMaxIndx + 1 юніта CommonDEF03.
2) Додати юніт AppDataXX.pas в проект клієнта і проект сервера
3) Додати юніт AppDataXX в список uses модуля CommonSRV03
Наприклад, якщо XX = 01 то:
4) Додати змінну App01 в модуль CommonDAT03

```
// Заголовок додатку модуля AppData01
var App01 : TApp;
```

5) Редагувати константу AllAppCount в юніт CommonDAT03

```
// Загальна кількість додатків
const AllAppCount = 2;
```

6) Додати в код функції function LoadAllPtScopeArr() : boolean модуля CommonSRV03 наступний фрагмент кода

```
// -------------------------------
// Указатели для тех.приложения App01
   if LoadAppLINKs(Addr(App01),
                   Addr(App01Const),
                   App01MaxIndx,
                   Addr(App01Scopes))
   then AllPtScopeArr[1] := Addr(App01)
   else AllPtScopeArr[1] := nil;
// -------------------------------
```

7) Виконати build-компіляцію клієнта і сервера

2.3. Тестування системи

Щоб переконатися, що система працює правильно потрібно провести її тестування.

Запуск та активація сервера

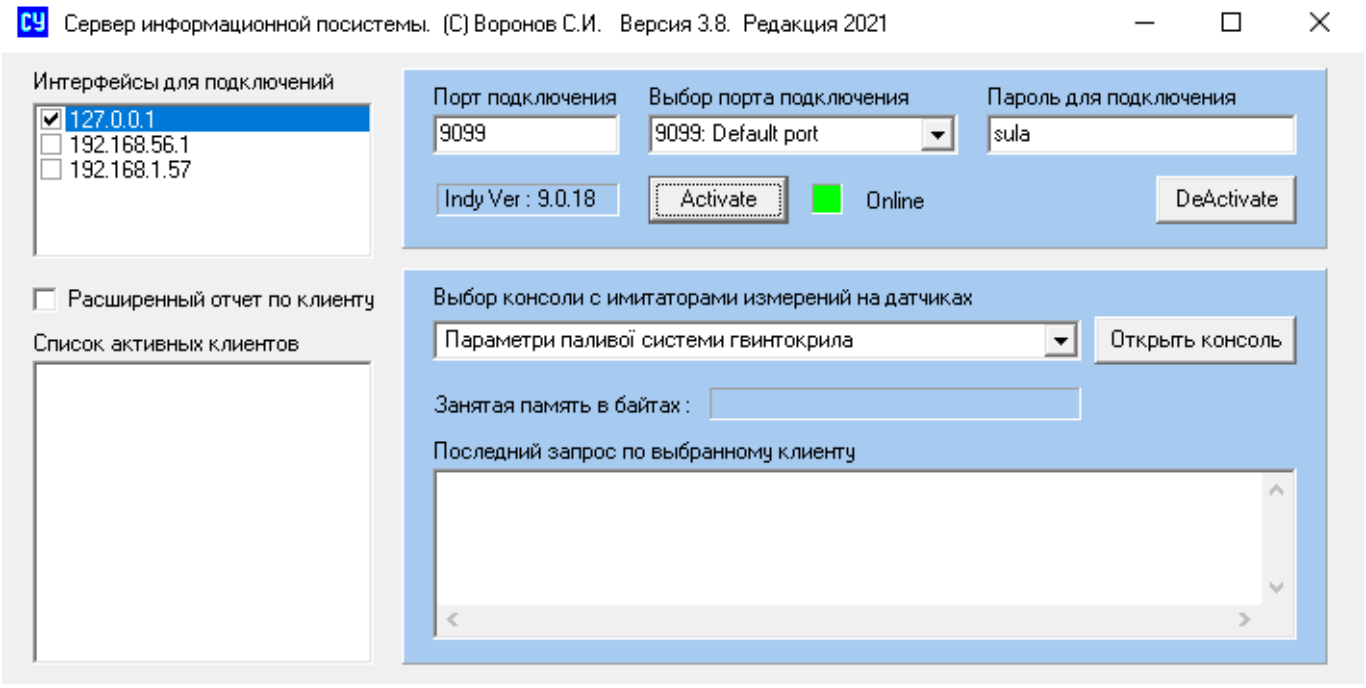

Рис. 2.13. Копія екрану активованого сервера

Активація сервера виконалась без виникнення помилок.

Запуск клієнта та підключення до сервера

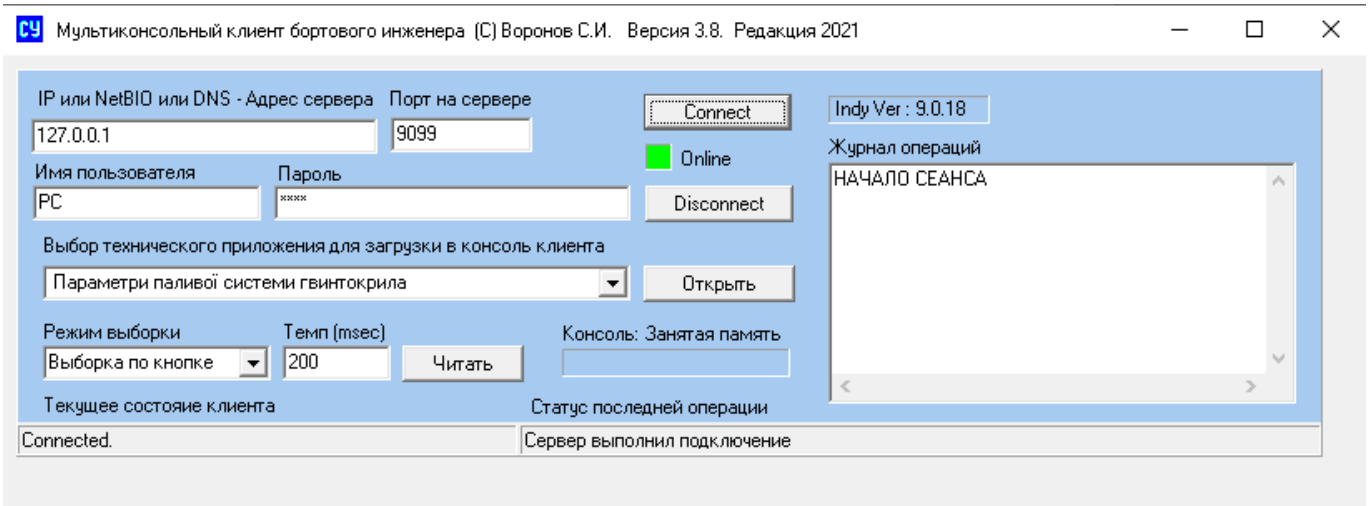

Рис. 2.14. Копія екрану клієнта підключеного до сервера

Активація клієнта виконалась без виникнення помилок. Після підключення клієнта до серверу в полі вікна сервера «Список активних клієнтів» появилася інформація про підключення клієнта.

Відкриття консолі імітаторів датчиків для паливної системи гвинтокрила

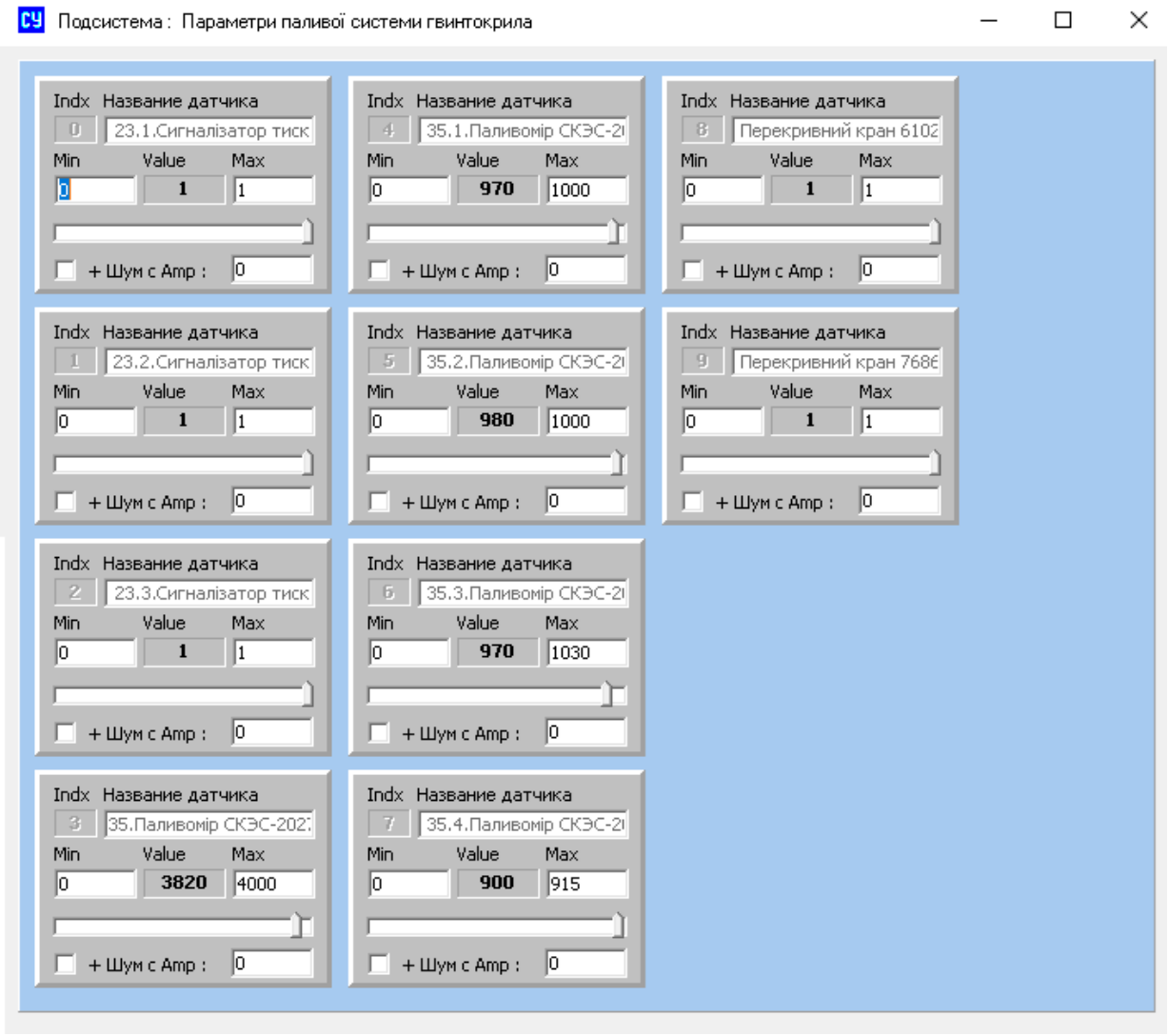

Рис. 2.15. Копія екрану консолі імітаторів датчиків

Імітатори всіх запланованих для відображення датчиків присутні та працюють як повинні.

Відкриття консолі параметрів паливної системи гвинтокрила та перевірка всіх правильної роботи всіх її елементів.

Всі елементи консолі відображення параметрів паливної системи гвинтокрила працюють як повинні. Відображувані параметри датчиків відповідають значенням заданим в консолі імітаторів датчиків на сервері та змінюються при оновленні.

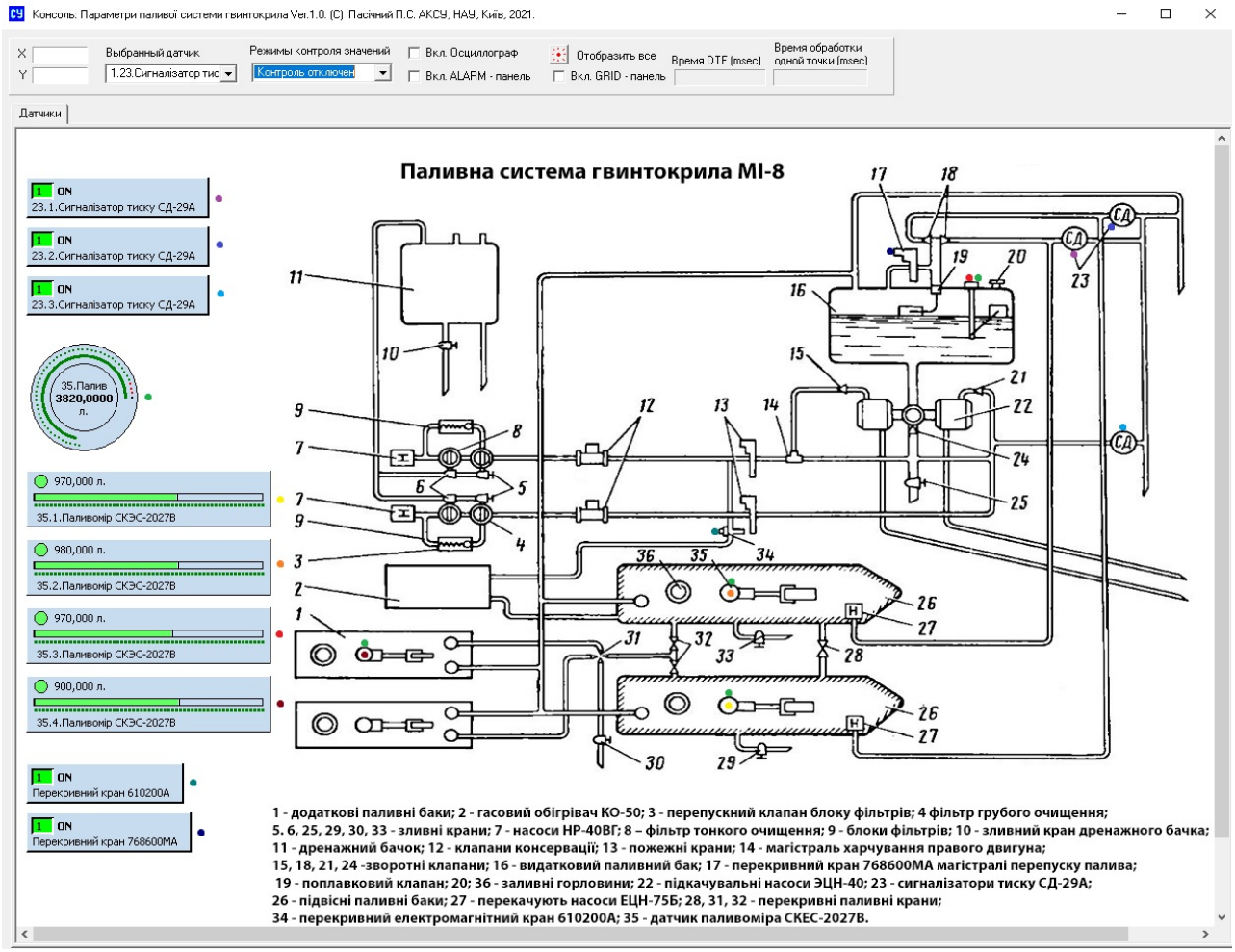

Рис. 2.16. Копія екрану консолі клієнта паливної системи гвинтокрила

Перевірка блокування завершення роботи

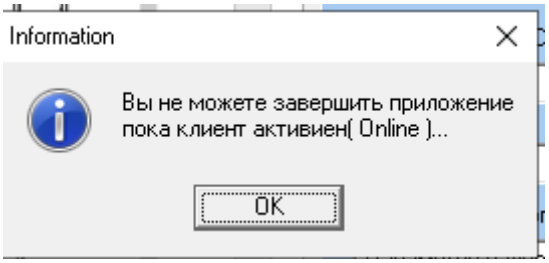

Рис. 2.17. Копія екрану відмови завершення роботи клієнта підключеного до сервера

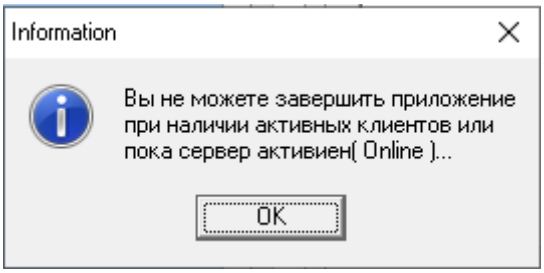

Рис. 2.18. Копія екрану відмови завершення роботи сервера при наявності підключеного клієнта

Перевірка допускового контролю паливоміра сумарної кількості палива

Встановивши значення імітатора датчика менше нижньої границі допуску та включивши ALARM-панель маємо результат зображений на рис.

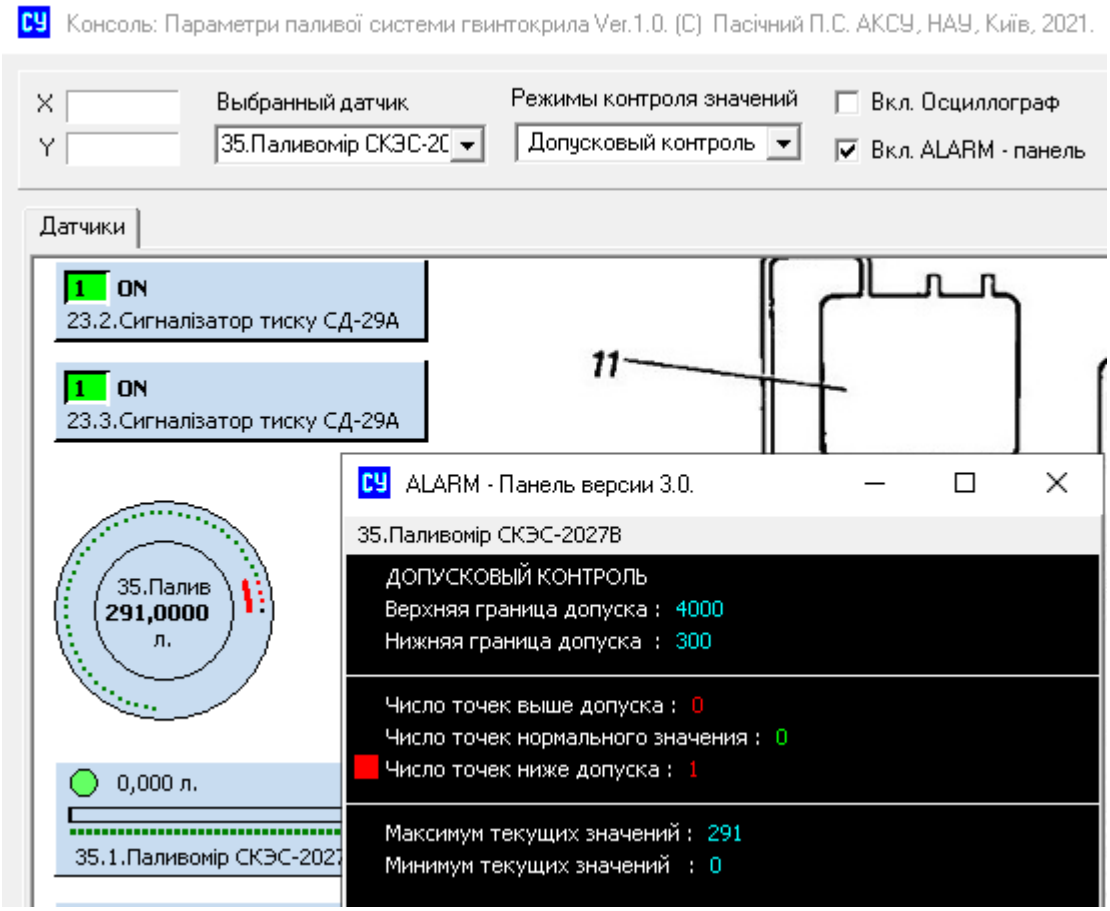

Рис 2.19. Датчик паливоміра сумарної кількості палива та ALARM-панель при залишку палива менше нижньої границі допуска

2.4. Пропозиції подальшого розвитку проекту

На цьому етапі виконується остаточна інтеграція (збірка) повнофункціонального серверного вузла технічної системи і виконується комплексне налагодження (верифікація взаємодії вузлів системи). Особливо необхідно підкреслити, що така інтеграція можлива тільки якщо інша група розробників реалізувала підсистему локального управління і/або спостереження за аналогією двохетапної схеми.

В рамках завдання розробки вузла консолі технічного персоналу, опис даного етапу є винятково інформаційним і приведено тільки для цілісного сприйняття цілей розробки такої консолі.

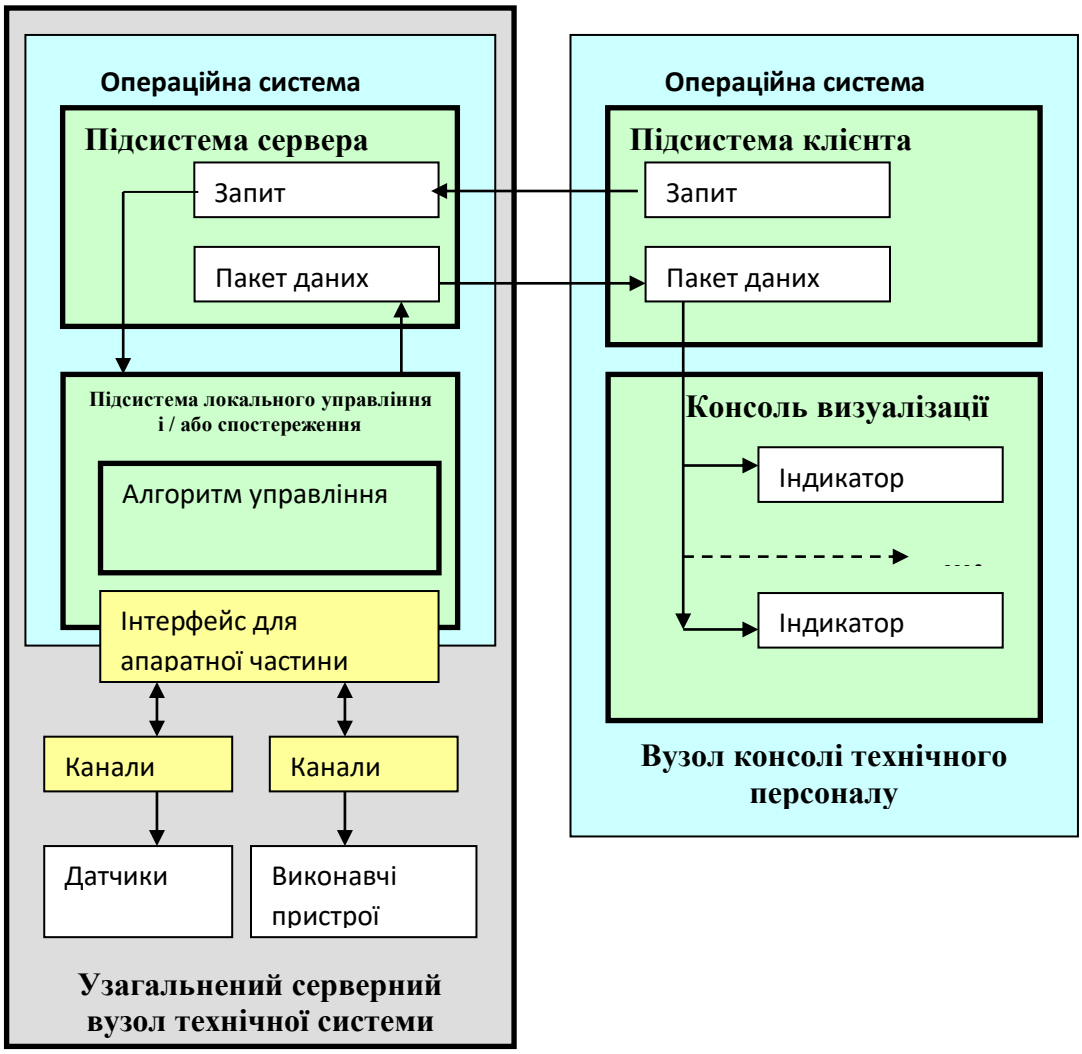

Рис. 2.20. Архітектура системи після повного складання конкретного серверного вузла технічної системи.

Висновки до розділу 2

В результаті тестування розробленої клієнт-серверної програми контролю та відображення параметрів паливної системи гвинтокрила можна зробити висновок, що програма працює так як було заплановано хоча в майбутньому може бути вдосконалена та розширена.

ВИСНОВКИ

В результаті теоретичних досліджень та розробки клієнт-серверної програми контролю та відображення параметрів паливної системи гвинтокрила я визначив деякі недоліки вирішення проблем які виникають в паливній системі гвинтокрила, та запропонував альтернативні рішення, розробив клієнт-серверну систему автоматизованого контролю та відображення параметрів технічного стану паливної системи гвинтокрила.

Тестування системи пройшло успішно, в функціонуванні програм сервера та клієнта помилок не виявлено що можна побачити на рис., а отже поставлена задача виконана але програма потребує подальшого вдосконалення та підбору реальних промислових комп'ютерів для її прикладного використання.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. С. И. Воронов. Выбор платформы для реализация систем управления, 2018,– 17 с.

2. КОНСЕРВАЦИЯ, РАСКОНСЕРВАЦИЯ. И ХРАНЕНИЕ ВЕРТОЛЕТА: [Електронний ресурс]. Режим доступу до ресурсу:

http://ooobskspetsavia.ru/2015/10/17/konservaciya-raskonservaciya-i-xranenie-vertoleta/

3. Сливные краны 601100М. : [Електронний ресурс] . Режим доступу до ресурсу:

http://cnit.ssau.ru/vertolet/mi8/mi7_6_12.htm.

4. Данилов В.А. Вертолет Ми-8:(Устройство и техническое обслуживание). – М.:Транспорт, 1988.—278 с.

5. Внешние дополнительные топливные баки ВДБ.6130.000: [Електронний ресурс] . Режим доступу до ресурсу: https://helicmi.ru/production/fueltanks.

6. Керосиновый обогреватель. : [Електронний ресурс] . Режим доступу до ресурсу: http://cnit.ssau.ru/vertolet/mi8/mi12_2_17.htm

7. Фильтры для авиационной промышленности: [Електронний ресурс]. Режим доступу до ресурсу: http://pgsystems.ru/avia-filtri-stendi

8. Фільтри для авіаційного палива І Фільтрування авіаційних олив: [Електронний ресурс]. Режим доступу до ресурсу: https://newfilter.com.ua/ua/palyvni/filtri-dlya-aviatsiynogo-paliva.html

9. Управления топливной системой вертолета ми – 8: [Електронний ресурс]. Режим доступу до ресурсу: https://studfile.net/preview/2584189/page:7/