Житомирська політехніка	Міністерство освіти і науки України Державний університет «Житомирська Політехніка»

Затверджено науково-методичною
радою Державного університету
«Житомирська політехніка»

МЕТОДИЧНІ РЕКОМЕНДАЦІЇ

для проведення лабораторних робіт
з навчальної дисципліни

«ОСНОВИ КОМП'ЮТЕРНО-ІНТЕГРОВАНОГО УПРАВЛІННЯ»

для студентів освітнього рівня «бакалавр»
денної та заочної форми навчання

спеціальності «151 Автоматизація та комп'ютерно-інтегровані технології»
освітньо-професійна програма: «Автоматизація та комп’ютерно-інегровані технології»
факультет комп’ютерно-інтегрованих технологій, мехатроніки і робототехніки
кафедра автоматизації та комп’ютерно-інтегрованих технології імені професора Б.Б. Самотокіна

Розглянуто та затверджено на
засідання кафедри автоматизації та
комп’ютерно-інтегрованих технологій
імені професора Б.Б. Самотокіна

протокол від «___» _____ 20___р.
№,____

Розробники:

к.т.н., доцент кафедри автоматизації та комп’ютерно-інтегрованих технології

Ткачук Андрій Геннадійович,

старший викладач кафедри автоматизації та комп’ютерно-інтегрованих

технології Богдановський Мартін Віталійович

Житомир
2020 рік

ЗМІСТ

ВСТУП

Створення сучасних систем управління виробничими процесами ґрунтується на використання комп'ютерної техніки на всіх етапах автоматизації виробництва. Комп'ютерні системи мають здатність гнучко та швидко бути підлаштованими під конкретні задачі управління, натомість класично вимагають узгодженої участі широкої групи спеціалістів на етапах проектування, відлагодження та подальшого супроводження проекту. У випадку складних виробничих процесів, класичними задачами, що підлягають автоматизації, є диспетчеризація та контроль параметрів технологічних процесів в режимі реального часу, що реалізується SCADA системами. Інформація, що поступає на диспетчерський пульт має відповідати канонам ергономіки та уніфікації, бути лаконічною для конкретного процесу, а управління має бути зручним та за можливості простим у маніпулюванні через елементи управління панелі оператора. Створення програмних інтерфейсів, що відповідають зазначеним положенням класично потребують глибинних знать програмування, як то при створенні інтерфейсів, так і простору змінних конфігурації обладнання. Натомість, бурхливий розвиток об'єктний та предметно-орієнтованих мов програмування дозволяє створювати уніфіковані засоби програмування виробничих систем, надаючи розробнику функціональні інструменти для відтворення алгоритмічної частини процесу управління та уніфіковані елементи візуалізації і контролю даних.

Одним з популярних середовищ програмування SCADA систем є продукт компанії National Instruments (NI) LabVIEW, що ґрунтується на використанні ряду мов програмування, основною серед яких є мова діаграм функціональних блоків FBD. Графічний спосіб програмування та уніфікація об'єктних моделей у прив'язі до апаратної частини програмованих логічних контролерів дозволяють значно пришвидшити процес створення та відлагодження SCADA пакету, суттєво зменшуючи необхідність глибоких знань синтаксису та об'єктних моделей мов програмування. Вивчаючи засоби та принципи створення проекту в LabVIEW студент має можливість отримати навички створення програмних продуктів під SCADA інших альтернативних засобів програмування виробничої автоматики.

Лабораторна робота №1

Тема: Інтерфейс та засоби програмування NI LabVIEW

Мета: Ознайомитись з інтерфейсом та основними інструментами створення проекту у програмному середовищі NI LabVIEW

Теоретичні відомості

LabVIEW (Laboratory Virtual Instrument Engineering Workbench - середовище розробки лабораторних віртуальних приладів) є середовищем програмування для створення додатків, використовуючи графічне представлення всіх елементів алгоритму за використанням мови “ G”, що відрізняється від мов програмування, таких як C, C ++ або Java, де програмують, використовуючи текст. Середовище розробки і виконання програм, призначена для дослідників - науковців і інженерів, для яких програмування є лише частиною роботи. LabVIEW функціонує на комп'ютерах під управлінням всіх поширених операційних систем: Windows, MacOS, Linux, Solaris і HP-UX.

Потужна графічна мова програмування LabVIEW дозволяє в сотні разів збільшити продуктивність праці. З LabVIEW на створення проекту додатку зазвичай потрібно кілька годин, оскільки пакет спеціально розроблений для роботи з даними і оформлення результатів. Так як LabVIEW має гнучкий графічний інтерфейс і простий для програмування, він також відмінно підходить для моделювання процесів, презентації ідей, створення додатків загального характеру і просто для вивчення сучасних мов програмування.

Віртуальний (програмний) прилад, створений в LabVIEW, має велику гнучкість в порівнянні зі стандартним лабораторним приладом, щодо відображення процесів та готових інтерфейсів для диспетчеризації даних. За допомогою LabVIEW допустимо створити необхідний тип віртуального приладу при дуже малих витратах в порівнянні зі звичайними інструментами. При необхідності ви можете легко внести в нього зміни.

LabVIEW створений для полегшення роботи з програмування ваших завдань. Для цієї мети є розширена бібліотека функцій і готових до використання підпрограм, які реалізують велику кількість типових задач програмування, зв'язані з автоматизацією у тому числі, і тим самим позбавляють нас від рутинної роботи з покажчиками, розподілом пам'яті та продуктивності. Спеціальні бібліотеки віртуальних приладів для введення / виведення даних з вбудованих апаратних засобів (data acquisition - DAQ), для роботи з каналом загального користування (КОП, General Purposes Interface Bus - GPIB), управління пристроями через послідовний порт RS-232, програмні компоненти для аналізу, уявлення і збереження даних, взаємодії через мережі Internet доповнюють основний інструментар. Бібліотека аналізу (Analysis) містить велику кількість корисних функцій, включаючи генерування сигналу, його обробку, різні фільтри, вікна, статистичну обробку, регресійний аналіз, лінійну алгебру і арифметику масивів.

Програмне середовище має стандартний для операційних систем GUI інтерфейс, який з’являється при запуску. Як і для інших сучасних середовищ програмування, створенню додатку передує створення проекту на базі існуючого чи пустого шаблону. Проект складається з двох основних інтерфейсів:

- лицьова панель (Front Panel) являє собою інтерактивний користувацький інтерфейс SCADA. На ній можуть знаходитися ручки управління, кнопки, графічні індикатори та інші елементи управління (controls), які є засобами введення даних з боку користувача, а елементи індикації (indicators) - вихідні дані роботи програми.

- блок-діаграма (Block Diagram) є основним інтерфейсом FBD програмування, створеним на мові графічного програмування LabVIEW, “G” . Компонентами блок-

діаграми є: віртуальні прилади (ВП) нижчого рівня, вбудовані функції LabVIEW, константи і структури управління виконанням програми. Для того щоб задати потік даних між певними об'єктами (створити зв'язок між ними) від полюсів FBD елементів протягуються лінії (wires). Об'єкти на лицьовій панелі представлені на блок-діаграмі у вигляді відповідних терміналів (terminals), через які дані можуть надходити від користувача в програму і навпаки;

- для того щоб використовувати деякий ВП як підпрограми в блок-діаграмі іншого ВМ, необхідно визначити його іконку (icon) і сполучну панель (connector). Віртуальний прилад, який застосовується всередині іншого ВП, називається, віртуальним підприладом (ВПП, SubVI), який аналогічний підпрограмі в традиційних алгоритмічних мовах. Іконка є однозначним графічним представленням ВП і може використовуватися в якості об'єкта на блок-діаграмі іншого ВП. Сполучена панель являє собою механізм передачі даних в ВП з іншої блок-діаграми, коли він застосовується в якості ВПП. Схоже до аргументів і параметрів підпрограми, сполучна панель визначає вхідні та вихідні дані віртуального модуля. Ієрархічне створення програми полегшує процес відлагодження та програмного опису типових процесів автоматизації.

У кожному з інтерфейсів існує своє контекстне меню, що можна викликати правою кнопкою миші, яке містить основні програмні методи та функції для роботи з елементами панелі (рис. 1.1.)

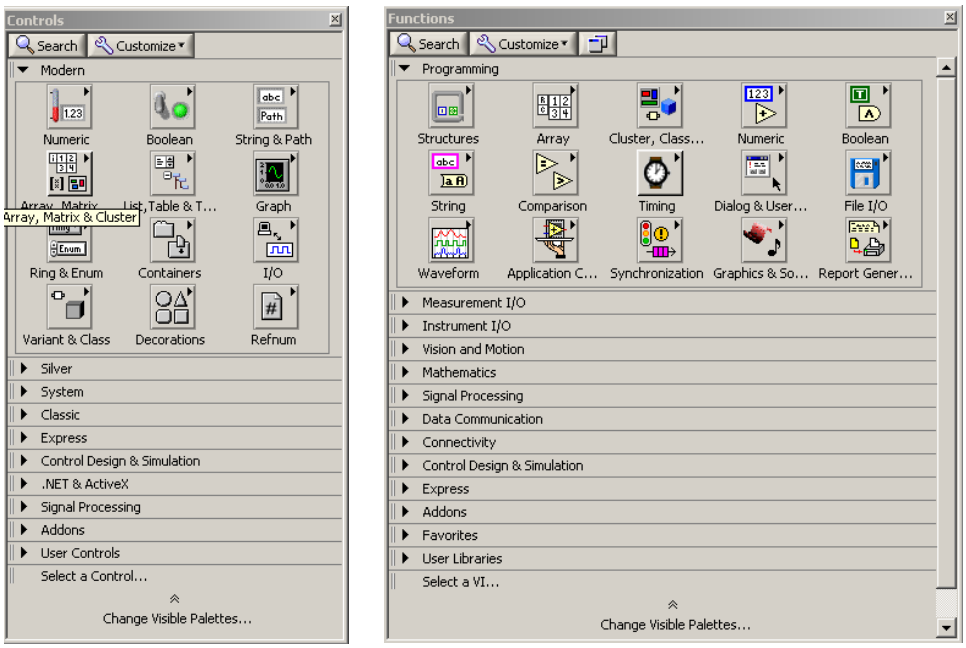


Рис.1.1. Контекстні меню Front panel (зліва) та Blok Diagram (справа).

Всі функції розбиті по окремих категоріях для полегшення навігації, а також можливо відшукати конкретні з них через вкладку **Search**. Кожну функцію супроводжує пояснення — **help**-меню за використанням якого нескладно розібратися з властивостями, методами, параметрами та навіть прикладами застосування. Після створення програного коду та інтерфейсу користувача проект можна запустити і відлагодити через головне меню програми. У лабораторній роботі LabVIEW використовується як засіб симуляції процесу диспетчеризації характеристик процесу. Це означає, що до персонального комп'ютера на якому програма реальний об’єкт не підключається а імітується за використанням математичної його моделі.

Порядок виконання роботи

1. За заданим викладачем варіантом обрати схему та параметри об’єкту імітації, наведені у додатку 1.

2. Розробити математичну модель для обраного об’єкту.

В якості об’єкту диспетчеризації розглянуто два типи схем: електрична та гідравлічна. У ідеалізованому та первинному наближенні обидві схеми підпорядковуються основним законам електротехніки, таким як закон Ома та Кірхгофа для кіл постійного струму та статичного режиму роботи гідравлічних систем із наступними відповідностями, зведеними до таблиці 1.

Таблиця 1. Відповідність основних величин

№	Електрична система	Гідравлічна система
1.	Сила струму, I, A	Витрати рідини, L, куб.м/с
2.	Різниця потенціалів (напруга), U або ЕРС, E, B	Різниця тисків (манометричний тиск), ⊗P, Па
3.	Постійний опір R, Ом	Опір трубопроводу, K, Па·с/куб.м

Основні закони, що можна математично та термінологічно формулюються так:

Закон Ома: Сила струму прямо пропорційна напрузі та обернено-пропорційна опору:

$$I = {}^{U}/_{R} \qquad 1.1$$

Або для гідравлічних схем:

$$L = {}^{\Delta P}/_{K} \qquad 1.2$$

Тут і надалі, враховуючи відповідність величин, наведених у таблиці 1.1, будемо виражати закони лише для електричних систем, розуміючи при цьому, що вони будуть відповідати і Гідравлічним системам.

Визначення опору: опір R прямо пропорційний довжині провідника (трубопроводу) L, питомому опору матеріалу (опору стінок трубопроводу при ламінарному режимі течії) Ro та обернено-пропорційний площі перерізу провідника (внутрішньому діаметру трубопроводу S:

$$R = \frac{Ro \cdot L}{S} \qquad 1.3$$

Закон Кіргофа (перший): сума струмів (витрат) у будь-якому вузлі (розгалуженні) системи дорівнює нулю:

$$\sum I = \mathbf{0} \qquad 1.4$$

Закон Кіргофа (другий): Алгебраїчна сума ЕРС (джерел тиску, таких як насоси) та падінь напруг (споживачів тичку — елементів гідравлічної системи) для будь-якої замкненої частини системи дорівнює нулю:

$$\sum E - \sum U = \mathbf{0} \qquad 1.5$$

Комбінуючи означені закони відповідно схеми за варіантом необхідно провести розрахунок параметрів струму та напруги (витрат та тиску) на означених ділянках систем, як параметрів6 що підлягають диспетчеризації. Приклад розрахунку складного кола методом контурних струмів (потоків) наведеній у додатку 2.

3. Створити новий проект, використовуючи пустий шаблон **blank VI**.

4. Створити власний графічний інтерфейс SCADA диспетчеризації параметрів. Для цього створити мнемосхему системи, використовуючи елементи малювання каталогу **Modern / Decorations** у контекстному меню **Front Panel**. Під мнемосхемою розуміється

інтуїтивне, функціональне представлення об’єкту диспетчеризації, що відповідає канонам ергономіки, впливаючи на швидкодію та зручність прийняття рішень оператором. Приклад мнемосхеми із елементами управління та індикації наведено на рисунку 1.2.

5. Створити елементи контролю (введення даних в програму) та індикації (виведення даних з програми на панель) згідно схеми розрахунку за варіантом. Має бути створено мінімум 2 елементи індикації (струм та напруга чи витрати тиск у розгалужених ділянках системи) та мінімум один елемент контролю (джерело напруги чи джерело тиску). Для цього скористуйтесь каталогом M**odern / Numeric**. Використайте різні способи представлення числових значень (аналогові та цифрові) і оцініть різницю у налаштуванні їх параметрів у контекстному меню, натискаючи правою кнопкою миші на доданому елементі контролю чи індикації на **Front Panel**. Приклад аналогових елементів контролю та індикації наведений на рисунку 1.2, під надписами Вентиль 1_1 (контроль), Резервуар1 (індикація) та аналогічні інші.

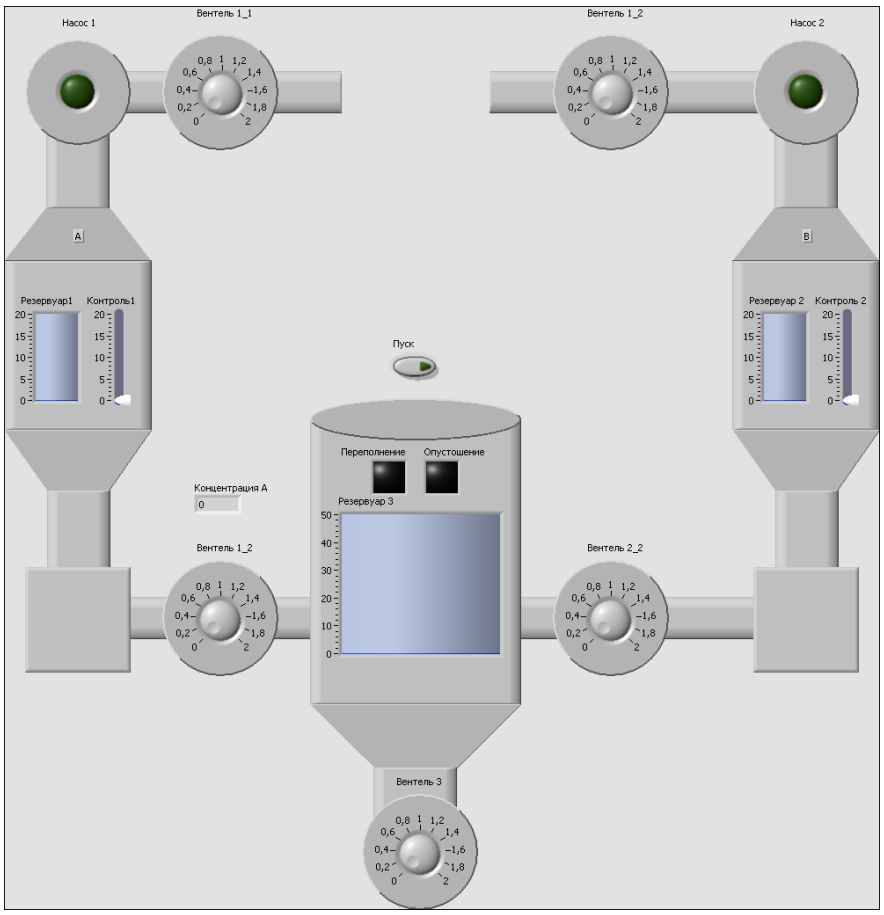


Рис.1.2. Приклад користувацького інтерфейсу SCADA LabVIEW.

6. Створити програму імітації зміни параметрів диспетчеризації на основі математичної моделі об’єкту, отриманої в пункті 2. Для цього перемикніться чи відкрийте через головне меню форму **Block Diagram**. Ви зможете побачити, що для всіх вами створених елементів контролю та індикації з’явились відповідні, однойменні термінали. Термінали, що містять стрілку зліва — термінали вводу даних, варто розташувати в лівому боці форми а термінали із стрілкою справа — термінали виведення даних — в правому боці. Так варто зробити, оскільки природно потоки (лінії зв’язку між терміналами) орієнтовані зліва направо. Далі потрібно скористатися бібліотекою стандартних, математичних функції, що реалізують математичну моль об’єкту. Оскільки означені закони базуються на 4 операціях: додавання, віднімання, множення та ділення, всі вони можуть бути знайдені в контекстному меню форми **Block Diagram** що

викликається правою кнопкою миші (рисунок 1.1., права частина) у каталозі **Numeric**. Означені термінали мають 2 входи і один вихід кожний, тому для отримання складених операцій вони використовуються багаторазово у каскадах, на зразок як то робиться з простими електронними схемами. Приклад додавання каскадом трьох чисел наведений на рисунку 1.3.

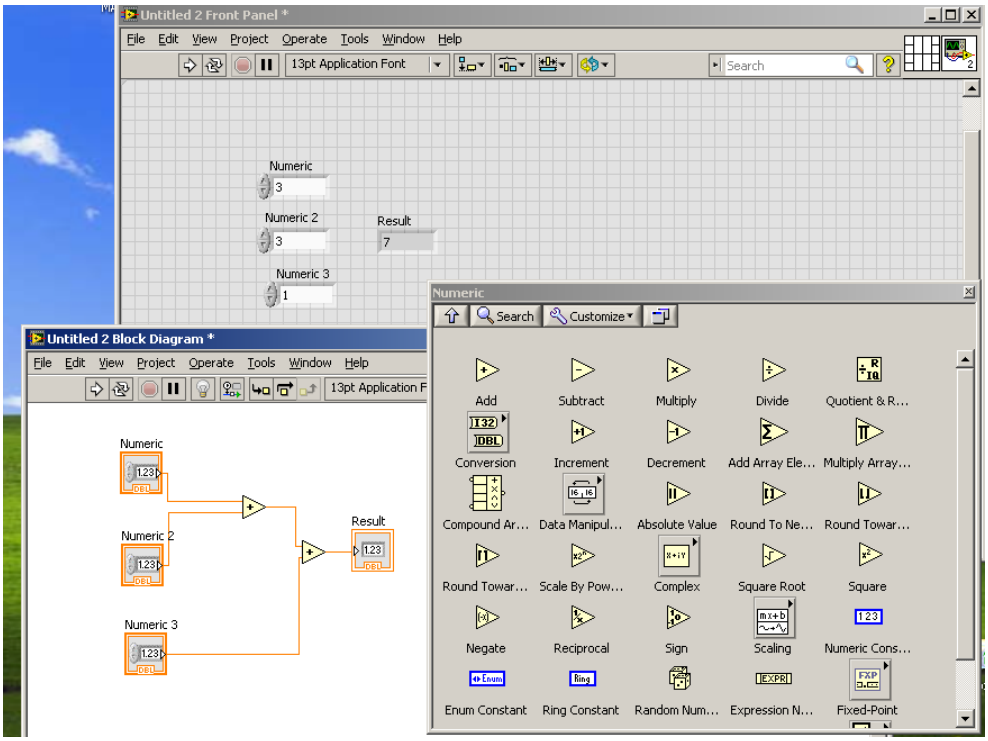


Рис.1.3. Додавання каскадом трьох чисел у LabVIEW.

7. Перевірити працездатність створеного проекту. Для перевірки запустити проект на циклічне виконання, витиснувши кнопку **Run Continuously** (дві стрілки, розташовані по колу в графічному, головному меню). Якщо маркована сітка на панелі інтерфейсу зникла і немає помилок компілювання, використовуючи елементи контролю внести зміни у запущеному стані та оцінити правильність отриманих даних на елементах індикації відповідно до моделі. У противному випадку виконати відлагодження програмної моделі до отримання позитивного результату. Зберегти проект для наступного використання.

8. Навести у звіті тему і мету роботи. У порядку виконання роботи навести опис математичної моделі системи (розрахунки на підставі законів, згідно схеми за варіантом з обраними числовими значеннями). Навести екранні форми розробленого продукту та зробити текстовий опис основних елементів та їх параметрів налаштування, відмінних від значень за замовчуванням.

9. Зробити висновки про відповідність розрахованих та отриманих програмно параметрів диспетчеризації створеної системи.

Контрольні питання

1. Яка мова програмування використовується в NI LabVIEW і в чому її переваги?
2. Якій каталог містить графічні елементи, що слугують для поліпшення представлення та ергономіки інтерфейсу користувача?
3. Скільки полюсів (входів та виходів) містить елемент ділення в каталозі **Numeric**?
4. Перерахуйте режими запуску проекту з основного меню та їх відмінності?

5. Яким чином використовуючи елементи множення каталогу **Numeric**, можна помножити 3 числа? Проілюструйте схематично.

6. Яке призначення форми **Front Panel** у проекті?

7. Використовуючи які команди можна вирівняти положення графічних елементів?

8. Які переваги мають аналогові індикатори над цифровими?

9. Якими чином у LabVIEW виконується сполучання обчислювальних операцій?

10. Що таке SCADA і які основні задачі нею вирішуються?

Лабораторна робота №2

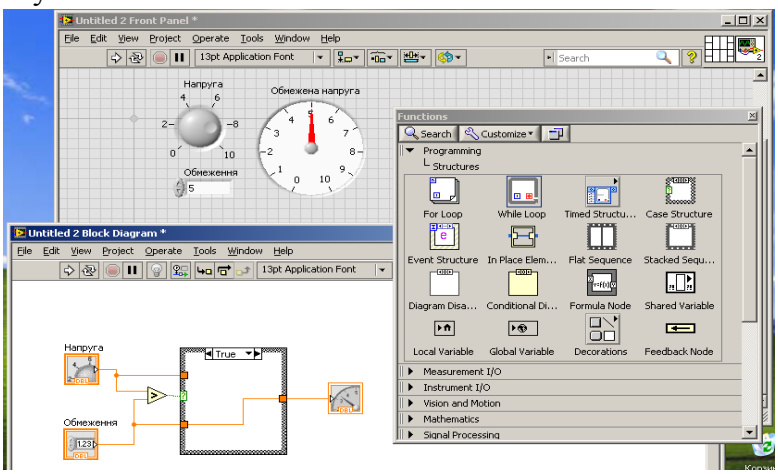
Тема: Оператори управління та способи відображення даних в NI LabVIEW

Мета: Ознайомитись з основними операторами управління та графічного відображення даних у програмному середовищі NI LabVIEW.

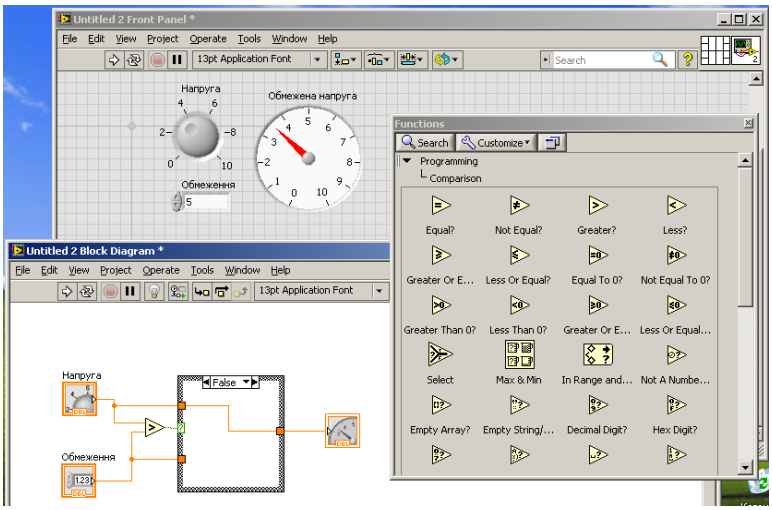
Теоретичні відомості

Автоматизація технологічних процесів із використанням SCADA у тому числі передбачає, що система має управління має здатність та властивості процедурно та алгоритмічно реагувати на зовнішні та внутрішні (програмні) події, що виникають під час роботи системи. LabVIEW містить чисельну кількість класичних та специфічних операторів, які дозволяють керувати ходом виконання програми. Перерахуємо основні з них, що знаходяться у каталозі **Programming / Structures** контекстного меню форми **Block Diagram**:

Case structure — оператор розгалуження, який дозволяє визначити гілку розвитку процесу при досягненні простої чи складеної умови. Аналог в класичних мовах програмування - оператор If чи Case. Приклад застосування оператора **Case Structure** наведено на рисунку 2.1.



а)



б)

Рис.2.1. Реалізація оператора case structue: а) вкладка реалізації події true; б) вкладка реалізації події false.

Оператор застосовується разом з терміналами порівняння із каталогу **Programming / Comparison** результат виконання яких є булевою функцією та слугує для вибору вкладок оператора **Case Structure**.

For Loop — оператор циклу, в якому тіло циклу виконується наперед визначену кількість разів. Приклад застосування циклу наведено на рисунку 2.2.

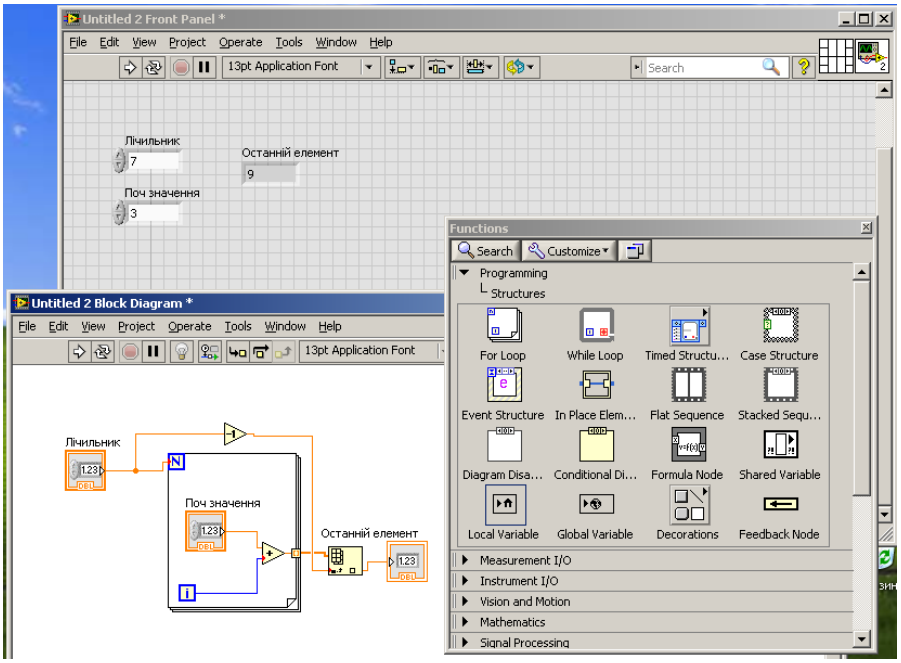


Рис.2.2. Реалізація циклу **For Loop**.

Оскільки виходом циклу є масив значень (позначено жирною лінією), для того? Щоб виділити та відобразити останній елемент масиву використано термінал **Index Array**. Треба зауважити, що початковий індекс **i** в циклі дорівнює нулю, тому для виділення останнього елементу дуло також використано термінал **Decrement**.

While Loop — оператор циклу, в якому тіло циклу виконується доти, доки не буде досягнута умова виходу з циклу. Приклад оператору наведено на рисунку 2.3.

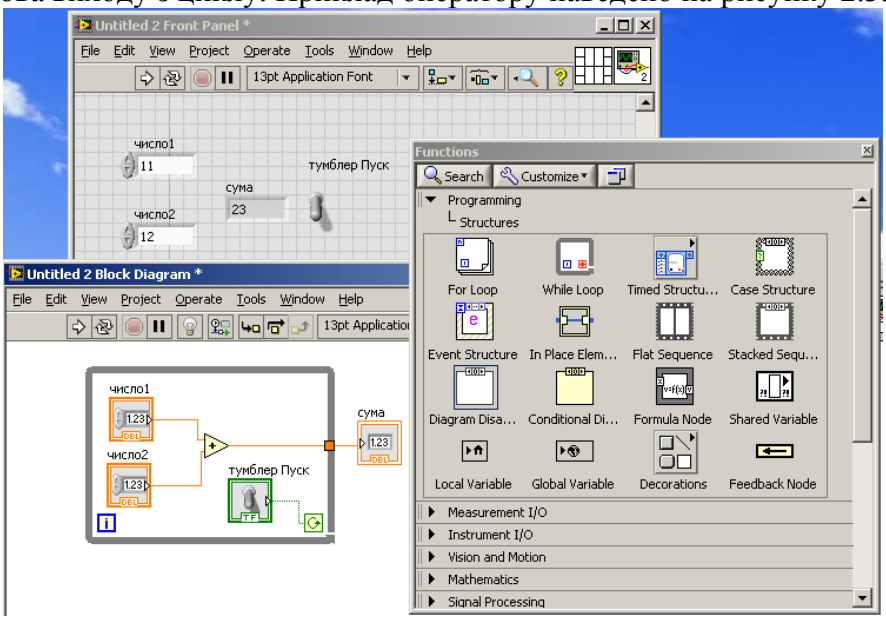


Рис.2.3. Реалізація циклу **While Loop**.

Умовою зупинки циклу є булевий покажчик, що керується вручну через термінал **Toggle Swich**. Треба зауважити, що для того, щоб спостерігалась імітація ігнорування зміни чисел, потрібно термінал індикації суми винести за межи циклу.

Sequence — цикл, що складається з декількох тіл (фреймів), що виконуються у послідовності один раз, після чого цикл розпочинається знову. Особливість в тому, що під час виконання даного циклу ми маємо доступ до змінних фреймів і можемо їх передавати між фреймами. Приклад циклу наведений на рисунку 2.4.

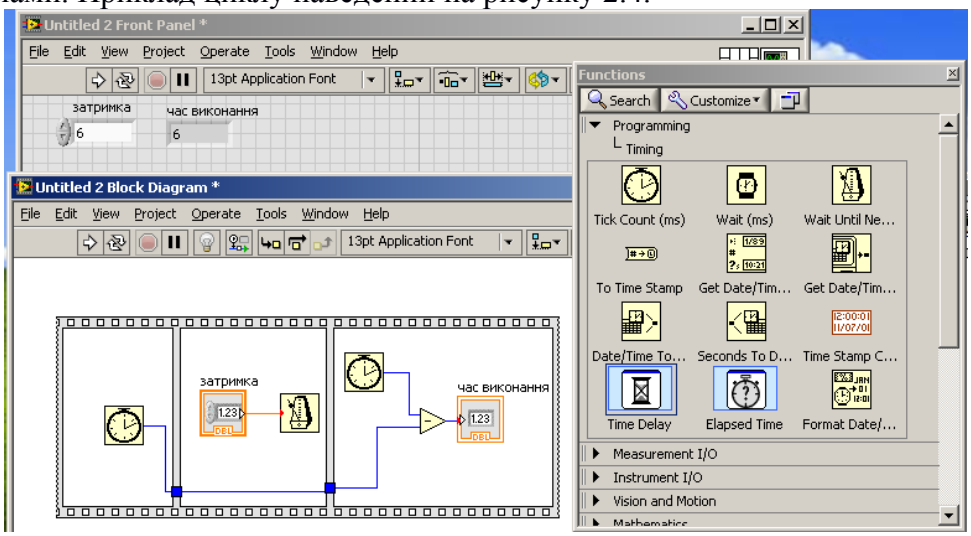


Рис.2.4. Реалізація циклу Sequence.

Цикл використано для розрахунку часу виконання програми у другому кадрі (фреймі). Для цього було використано термінал **Tick Count** із каталогу **Timing** що таймер реального часу значення якого різняться між третім та першим кадрами через затримку на виконання програми у другому кадрі. Для чистої затримки було обрано термінал **Wait Until Next**.

Formula Node — використовується для швидкої символічної реалізації бажаної функції або складеної функції на базі простих. Приклад структури наведено на рисунку 2.5.

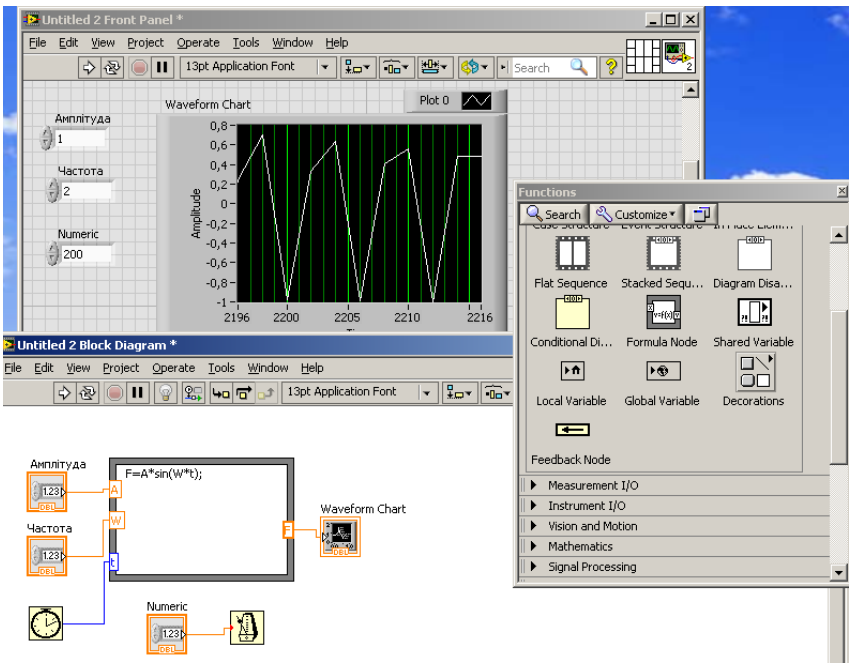


Рис.2.5. Реалізація складеної функції **Formula Node**.

Для ефективної диспетчеризації даних їх представляють графічно.

Waveform Chart — компонент, що використовується для побудови часових характеристик системи. Може відображати один або декілька процесів у обраному масштабі часу. У контекстному меню компоненту доступні опції способів відображення даних, масштабування та часової розгортки. Приклад застосування компоненту наведено на рисунку 2.5.

Waveform Graph — компонент, що використовується для відображення процесів як послідовностей (дискрет) відносно довільного базису. Налаштування компоненту можливе через його контекстне меню а опції налаштування подібні до **Waveform Chart**. Приклад реалізації послідовностей А та В наведено наведено на рисунку 2.6.

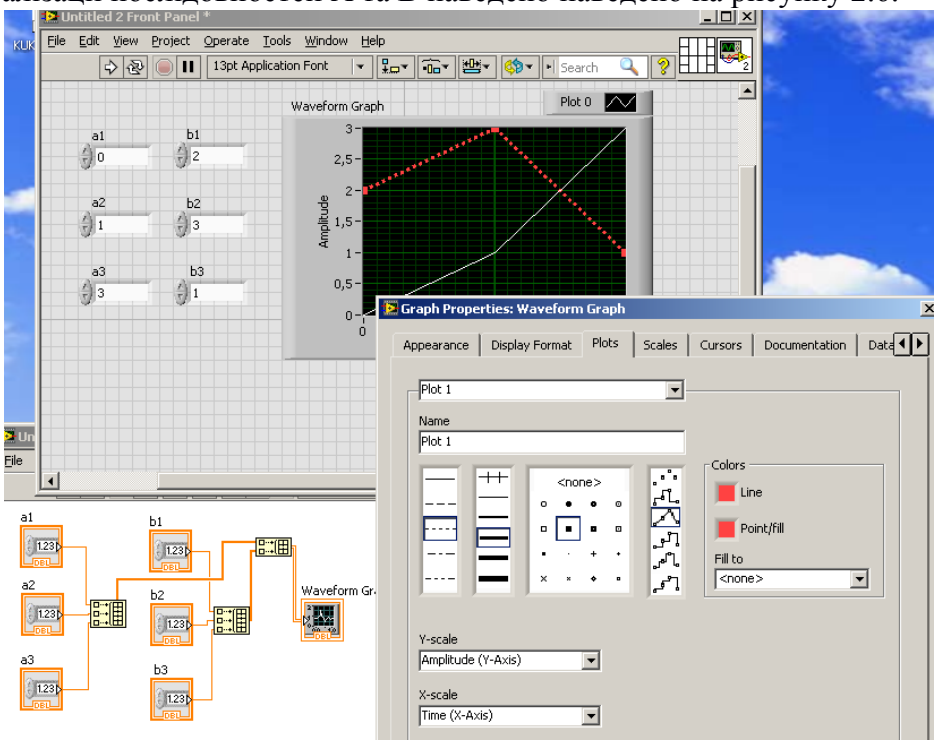


Рис.2.6. Реалізація та відображення послідовностей А та Б з використанням **Waveform Graph**.

Для відображення послідовності необхідно сформувати масив даних, який реалізуються використовуючи компонент **Build Array** із каталогу **Arrays**. Для відображення декількох послідовностей на одному полі потрібно створити двовимірний масив шляхом каскадування, як то показано на малюнку 2.6.

XY Graph — компонент, що слугує для відображення графічних залежностей одного потоку чи потоків даних від іншого. Приклад реалізації **XY Graph** для побудови квадратичної залежності наведено на рисунку 2.7.

Треба зазначити, що при створенні нового екземпляру компоненту, одразу з'являється декілька зв'язаних потоками елементів що налаштовані по замовченню на відображення одновимірних величин. При протягування потоків масивів на входи компоненту **XY Graph** автоматично створюються компоненти конвертування типів даних.

Для формування випадкових чисел із нормальним законом розподілу використовується термінал **random**, який знаходиться в каталозі **Numeric** панелі **Block Diagram**. Він генерує випадкові числа в діапазоні від нуля до одиниці (нормовані).

При створенні змінних чи терміналів, як і у класичних мов програмування існує типізація даних. Типи даних легко розрізняти за кольорами, легенду яких можна подивитись, якщо зайти в контекстне меню терміналу, обрати пункт властивості та перейти на вкладку **Data Type**. Для приведення типів даних та їх кластерів використовується компоненти конвертації, розташовані у каталозі **Programming / Numeric / Conversion** та **Programming / Array**.

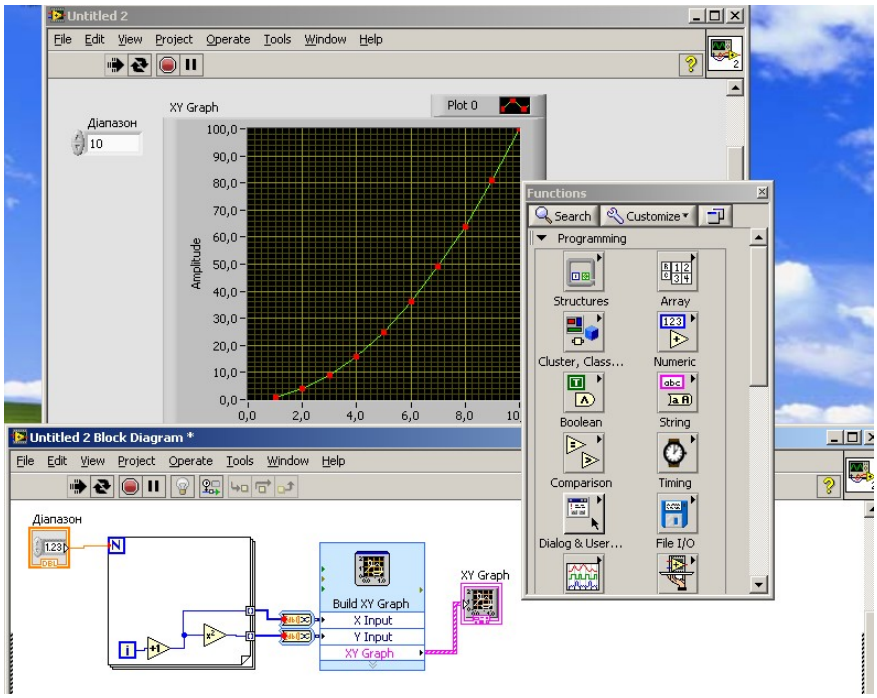


Рис.2.7. Реалізація квадратичної залежності між масивами даних за використанням компоненту **XY Graph**.

Порядок виконання роботи

1. Відкрити проект попередньої лабораторної роботи у програмному середовищі LabVIEW.

2. Використовуючи компонент **Case Sequence** реалізувати розгалуження програми імітації показів приладів:

А) Імітувати процес обриву паралельної гілки електричної системи чи повне перекриття паралельної ділянки трубопроводу гідравлічної системи шляхом миттєвого зростання значення опору ділянки до величини, що на 3 чи чотири порядки вище за розраховане, нормальне значення. Підставкова нескінченних значень не рекомендується у випадку використання вихідної математичної моделі системи без істотних доробок. Керування реалізується вруну (кнопкою або тумблером)

Б) Імітувати перемикання одного з аналогових приладів на інший, додатковий прилад того ж типу, діапазон вимірювання та ціна поділки якого вдвічі більша. Якщо на первинному приладі стрілка потрапляє у діапазон максимально-допустимих значень вхідної величини, (80-90% правої частини шкали) потрібно обнулити значення першого та відобразити поточні показники на другий (каскад приладів). Процес має бути зворотнім. Керування реалізується по рівню сигналу.

3. Використовуючи компонент **While Loop** імітувати відключення довільного приладу в процесі роботи програми а також запуск та зупинку програми вручну (кнопкою чи тумблером)

4. Реалізувати імітацію нестабільності джерела напруги чи тиску (на насосі) використовуючи компоненти **Formula Node** (для імітації періодичного збурення) та **Random** (для імітації випадкового збурення) Обидва збурення впливають адитивно (додаються) а величина сумарного збурення не повинна перевищувати 10% від рівня (амплітуди) ідеального сигналу (сигналу до збурення). Зробити можливість відключати збурення на джерелі тумблером чи кнопкою.

5. Реалізувати з використанням компоненту **Sequence** розрахунок часу роботи основної програми. Використовуючи цикл **For Loop** отримати усереднене значення часу роботи програми на 100 вимірах. Для цього в циклі просумувати 100 значень часу роботи і поділити на 100.

6. Відобразити у часі покази двох аналогових приладів, використовуючи **Waveform Chart**. Налаштувати властивості трендів так, щоб один був суцільною лінією а другий пунктирною.

7. Відобразити залежність напруги від струму (ВАХ — вольт-амперна характеристика) (тиску від витрат) чи тиску від витрат (характеристика насосу) використовуючи компонент **XY Graph** та цикл **For Loop** для перших 20 значень від пуску програми.

8. Навести у звіті тему і мету роботи. У порядку виконання роботи навести екранні форми розробленого продукту для кожного пункту виконання та зробити текстовий опис основних елементів, що використовувались, та їх параметрів налаштування, відмінних від значень за замовчуванням.

9. Зробити висновки про адекватність імітації роботи технічної системи.

Контрольні питання

1. Скільки станів (графічних вкладок)має оператор розгадування в **LabVIEW** по замовченню?

2. Скільки разів виконається тіло циклу оператора **For Loop**, якщо на вхід **N** подати 10?

3. В чому полягає різниця між операторами цикл **For Loop** та **While Loop**?

4. Що реалізує термінал **Tick Count**?

5. Який тип змінних повертає термінал **Push Button** (кнопка) чи **Toggle Swich** (тумблер)?

6. За використанням якого компоненту можливо реалізувати складену символьну функцію?

7. Чим принципово відрізняються компоненти **Waveform Chart** та **Waveform Graph** у способі відображення характеристик систем?

8. Який компонент потрібно обов’язково використовувати, щоб побудувати 2 графіка на одній площині?

9. В якому діапазоні генерується випадкове число компонентом **Random**?10. Який оператор реалізує визначену наперед послідовність виконання в нескінченному циклі?

Лабораторна робота №3

Тема: Організація роботи з масивами та рядками даних в середовищі NI LabVIEW

Мета: Ознайомитись з основними компонентами формування та роботи з масивами та символьними рядками в програмному середовищі NI LabVIEW.

Теоретичні відомості

На практиці, розробка SCADA оправдана необхідністю обробки значної кількості інформаційних потоків, впорядкування даних багатьох одночасних процесі, надання зручних механізмів контролю та спостереження за ними. Оперативне опрацювання даних базується на використанні таких структур, як масиви та кластери, що дозволяють алгоритмічно та програмно їх впорядкувати. Типові операції роботи із даними в означених структурах реалізується в NI LabVIEW через множину функціональних блоків представлених представлених графічними компонентами бібліотек. Базові каталоги, що містять основні компоненти є **Array, Matrix & Clusters** та **String & Path**, доступні через контекстне меню **Front Panel** та **Array**, **Cluster** та **String**, доступні через контекстне меню **Block Diagram**. У продовженні проекту реалізуємо частину доступного інструментарію.

Масиви. У випадку розробки інструментарію користувача для контролю та спостереження за масивами даними, зручно починати із додавання елементів інтерфейсу **Front Panel**. Для цього розпочнемо з об'єкту-контейнеру **Array** з каталогу **Array, Matrix & Clusters**. Сам по собі контейнер не має ані типу та представлення даних ані типу терміналу (введення чи виведення). Для того щоб ініціалізувати типи та представлення, необхідно перетягнути в контейнер одномірний елемент контролю чи індикації. Приклад ініціалізації масивів введення та виведення даних зображено на рисунку 3.1.

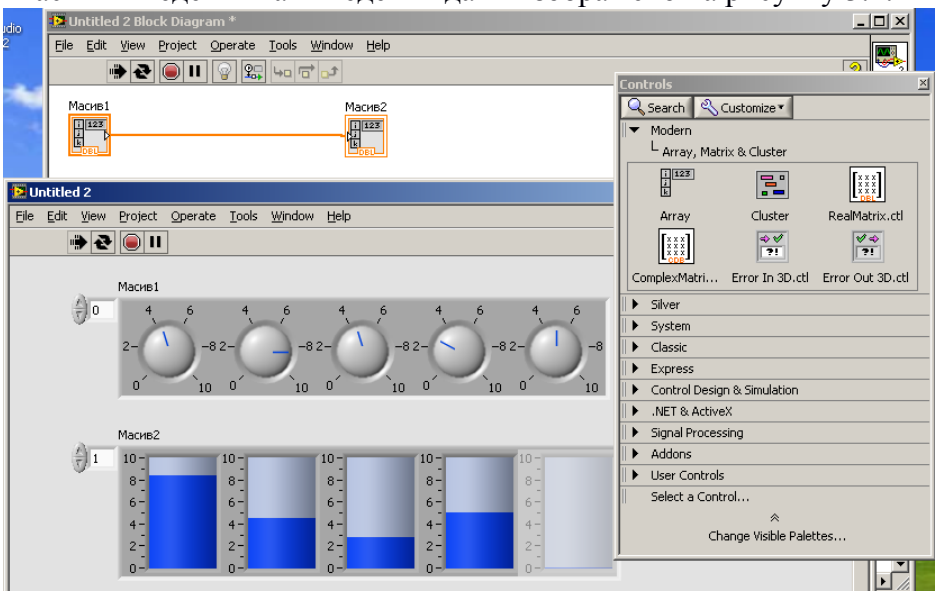


Рис.3.1. Ініціалізація масивів даних в **NI LabVIEW**.

Треба зазначити, що в контейнері масив реалізується динамічно, тобто розмір масиву змінюється в залежності від ініціалізації його елементів. Змінити розмірність масиву можна через контекстне меню контейнеру, пункт **add demention**. Наочне представлення масиву можна забезпечити розтягуючи контейнер по ширині та прокручуючи за використанням індексу першого елементу масиву, що спостерігається, зліва, в межах розтягненого вікна. На рисунку видно, що масив 2 як раз зсунуто через індекс у видимому діапазоні наліво (за межами відображення, зліва), тому першому

відображеному елементові масиву 2 відповідає другий елемент масиву. Треба розуміти, що це тільки впливає на відображення і жодним чином не впливає на зміну індексації елементів. Також справа у масиві 2 видно не ініційований елемент, тому що йому немає відповідного з масиву 1.

Особливим типом масивів є кластер, який дозволяє використовувати елементи з різними типами даних. Приклад ініціалізації кластерів наведений на рисунку 3.2.

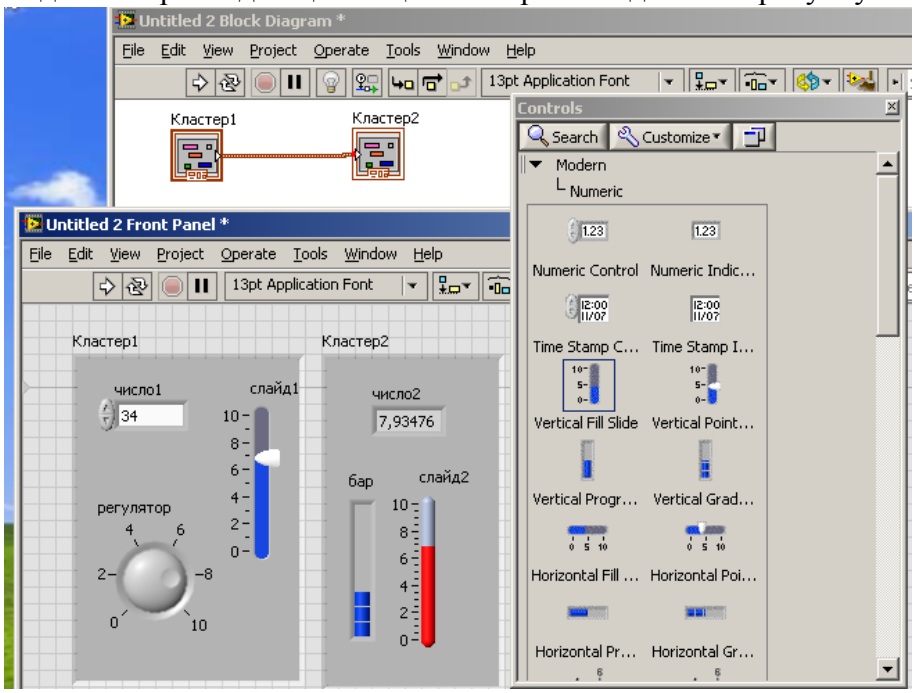
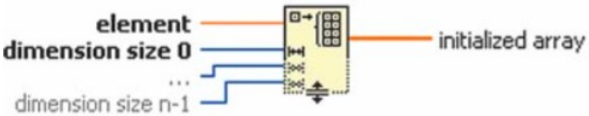


Рис.3.2. Ініціалізація кластерів з трьома елементами.

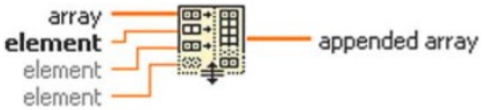
Зазначимо, що при прямому зв'язку кластерів, як то показано на рисунку 3.2., їх елементи поєднуються у порядку їх створення у кластерах: перший створений у кластері 1 (число1) з першим створеним у кластері 2 (бар) і так далі.

Основними програмними елементами роботи з масивами, що доступні у контекстному меню **Block Diagram** є наступні:

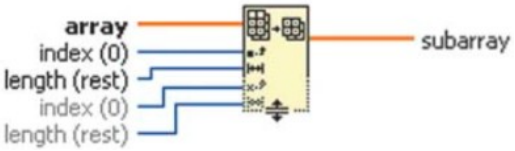
Array Size — повертає розмір масиву по кожній з розмірностей (у випадку багатовимірного);



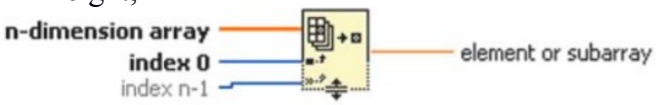
Initialize Array — створює масив зазначеної розмірності, заповнений значенням elements;



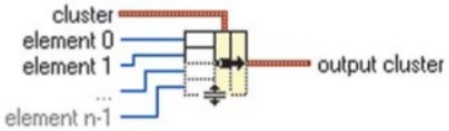
Build Array — додає до вже існуючого масиву array потоки даних element;



Subset Array — вирізає підмасив, починаючи по кожній розмірності з елементу index кількість елементів leight;



Index Array — виділення елемента чи підмасиву по індексах в порядку формування розмірностей index;



Bundle — додає до існуючого кластеру нові елементи elenent.

З іншими функціями можна ознайомитись, користуючись контекстною довідкою help для компонентів зазначених вище категорій.

Для світлової індикації поточних та критичних значень параметрів чи станів системи використовується компоненти LED (круглий та квадратний), що знаходяться в каталозі **Modern / Boolean / LEDs**. Через контекстне меню їм можна задати колір, маскування, режими спрацьовування тощо. Для реалізації булевих операцій слугує каталог програмних компонентів **Boolean**. Зміст компонентів зрозумілий із графічного представлення та назв однойменних булевих операцій ТА, АБО, НЕ тощо. Приклад формування та обробки даних масиву наведено на рисунку 3.3.

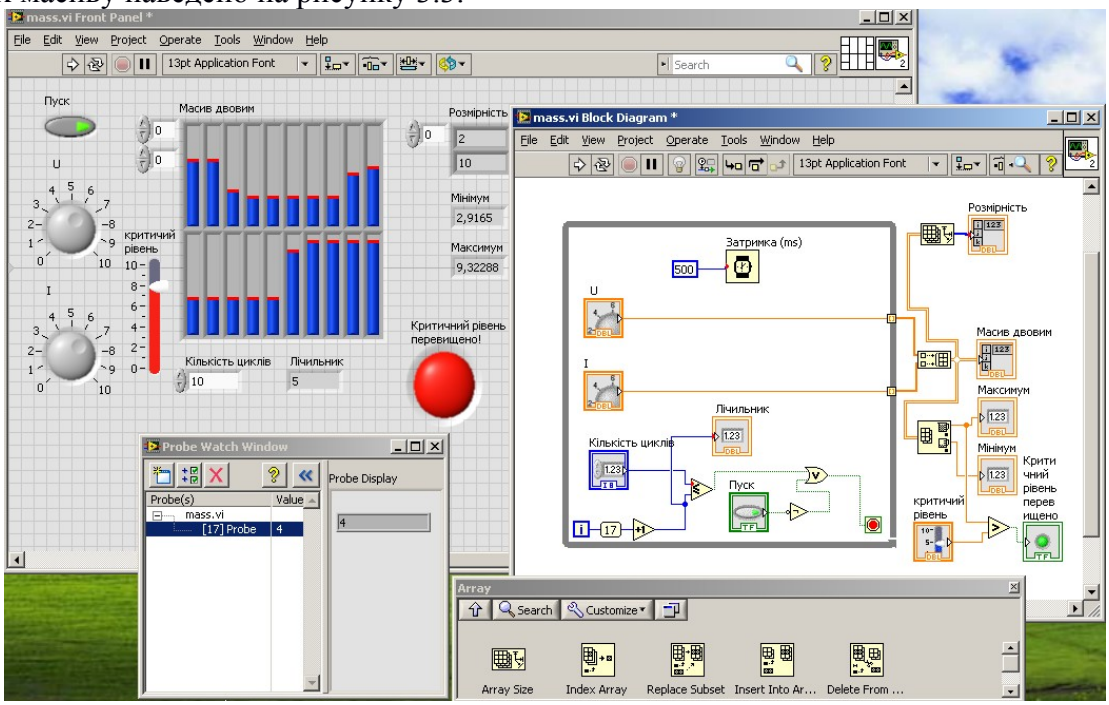


Рис.3.3. Формування та обробка даних масиву в NI labVIEW.

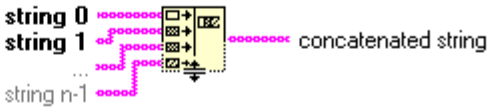
Для формуванні масиву із даними роботи програми реального часу, при виході потоків за межі циклу (перетин межі циклу **While Loop**) необхідно активувати індексацію у контекстному меню на помаранчевих квадратиках — **Enable Indexing**. При цьому вони отримують біле забарвлення, як то показано на рисунку 3.3. Для сповільнення формування даних та наочного спостереження заповнення масиву всередині циклу компонент затримки **Wait** із параметром 500 мс. Затримка впливає лише на реальний час всередині циклу. Цикл зупиняється по досягненню покажчика “Кількість циклів” або деактивації кнопки “Пуск”. Для цього використані логічні оператори **OR** та **NOT**.

Корисним інструментом відлагодження програми є Probe Watch, аналогічний по функціям відладчикам класичних засобів програмування. Його активувати можна під час

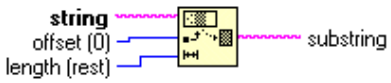
роботи програми, підносячи курсор до будь-якого потоку (лінії зв'язку) та натискаючи ліву кнопку миші. При цьому на лінії з'явиться круг з цифрою — номером зонду, та відкриється вікно Probe Watch Window, в якому можна спостерігати зміну даних у потоці в реальному часі усіх встановлених зондів.

Організація людино-машинного інтерфейсу (HMI) на більш глибокому рівні ґрунтується на обміні, валідація та інтерпретація текстових повідомлень. Для цього у NI LabVIEW існує множина типових програмних компонентів роботи із рядками, діалоговими формами і іншими структурами представлення. Як і у випадку компонентів для роботи з числовими даними, компоненти для роботи з рядками діляться на елементи контролю та індикації, доступні у каталозі **String & Path**. Наведемо основні програмні компоненти роботи з рядками:

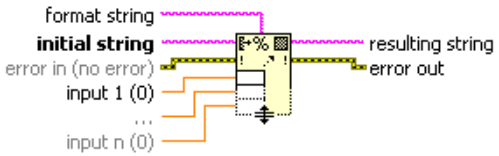
String Length — визначає довжину рядка та повертає кількість символів числом;



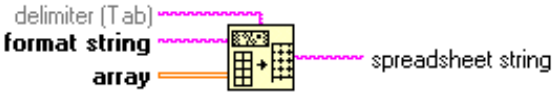
Concatenate strings — об'єднує декілька рядків в один рядок;



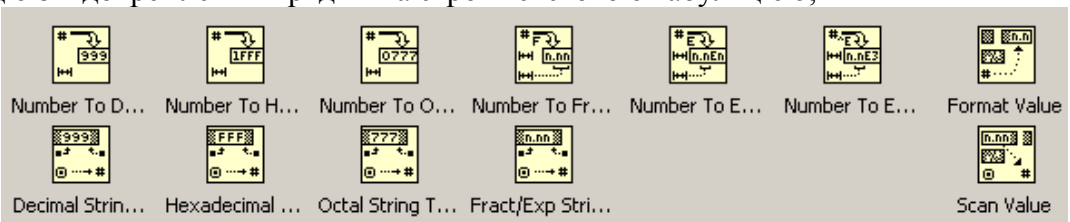
Srting Subset — Вирізає із рядка підрядок, починаючи з символу з порядковим номером offset та довжиною light;



Format Into String — форматує та перетворює числові та символьні типи даних у рядок;



Array To Spreadsheet String — перетворює масив числових чи символьних даних на таблицю з відокремленням рядків та строк текстовою табуляцією;



String/Number conversion — підкаталог конвертації різних типів даних у строковий та навпаки.

Приклад застосування окремих візуальних та програмних компонентів роботи із рядками наведено на рисунку 3.4. Для реалізації ефекту постійного затискання кнопки ОК для виконання програми при правильному паролі, було змінено стандартний режим спрацювання — тригер — з короткого одноразового та постійний. Доступ до зміни режиму тригерів для кнопок та тумблерів знаходиться в їх контекстному меню, пункт **Properties**, у закладці **Operations**. У списку **Button Behavior** можна обрати один з шести режимів спрацювання, протестувавши у тій же закладці, знизу, поведінку на імітаторі. Для

того, щоб у рядку введення паролю символи відображались зірочками, необхідно в контекстному меню компоненту рядка вибрати опцію **Password Display**.

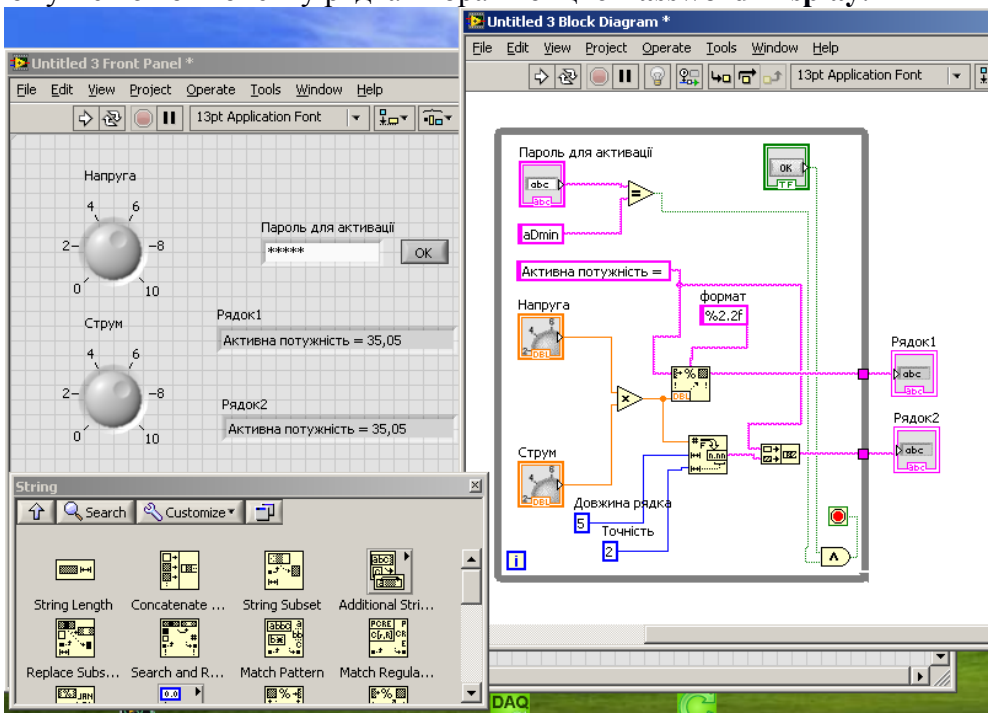


Рис.3.4. Робота із рядками в середовищі LabVIEW.

Більш детально ознайомитись з властивостями компонентів та іншими компонентами каталогу роботи з рядками можна через контекстну довідку LabVIEW.

Порядок виконання

1. Відкрити проект попередньої лабораторної роботи у програмному середовищі LabVIEW.

2. Використовуючи масиви чи кластери, впорядкувати прибори (індикатори) та елементи контролю як елементи масиву. Сповільнити темп реального часу в основному циклі програми до 0.2-0.5 секунд. Створити буфер даних з 20 елементів (див. варіант на рис.3.3.), відшукати значення та номери у послідовності найбільшого елементу.

3. Вивести данні на екранну форму у табличному вигляді із буферу, створеному у пункті 2.

4. Зробити текстове повідомлення та сповіщення двома лампочками (двох кольорів) про нормальне і критичне значення буд-якого приладу. Текст має бути складений з трьох частин, що виводяться у рядок: “Система”+”у нормальному”+“режимі” або “Система”+”у критичному”+”режимі” Вивести на панель реальний час, коли сталось останнє критичне значення.

5. Зробити доступ по паролю до запуску вашої програми.

6. Навести у звіті тему і мету роботи. У порядку виконання роботи навести екранні форми розробленого продукту для кожного пункту виконання та зробити текстовий опис основних елементів, що використовувались, та їх параметрів налаштування, відмінних від значень за замовчуванням.

7. Зробити висновки про переваги та недоліки використання структурованих типів даних у розробленій програмі.

Контрольні питання

1. Яка принципова відмінність у структур даних масив та кластер?

2. Перерахуйте основні способи ініціалізації масивів в програмі NI LavVIEW?

3. Опишіть процес створення масиву в елементів, сформованих вручну?
4. Яким чином можливо сповільнити реальний час частини процесу чи програми?
5. Які операції над булевими змінними (булевої алгебри) реалізовано в LabVIEW?
6. Опишіть дії, необхідні формування рядка, що містить числові дані?
7. Що буде в результаті форматування числа 112.23, якщо задати параметр %7.1f?
8. Як реалізується відображення символів в 16-річному коді при наборі?
9. Який тип та структуру даних позначає подвійна синя лінія (потік)?
10. Що таке тригер кнопки і які доступні у LabVIEW?

Лабораторна робота №4

Тема: Створення додатку із діалоговими вікнами, засобами представлення, створення та збереження даних у файловій системі з NI LabVIEW

Мета: Ознайомитись з елементами представлення та діалоговими формами користувача , засобами створення та збереження даних у файловій системі із використанням програмного середовища NI LabVIEW.

Теоретичні відомості

Для ефективної взаємодії користувача і системи при диспетчеризації значної кількості даних, оправданим є фрагментація інформації для переставлення із використанням діалогів та форм а також її збереження на носіях даних. Компоненти для створення елементів відображення даних у табличній формі а також для роботи із каталогами файлової системи знаходяться в контекстному меню **Front Panel**, в каталозі **List, Table & Tree**. Компоненти для способів упорядкування та представлення екранних форм - в каталозі **Containers**. Приклад застосування візуальних і програмних компонентів у двох режимів відображення даних наведено на рисунках 4.1. та 4.2.

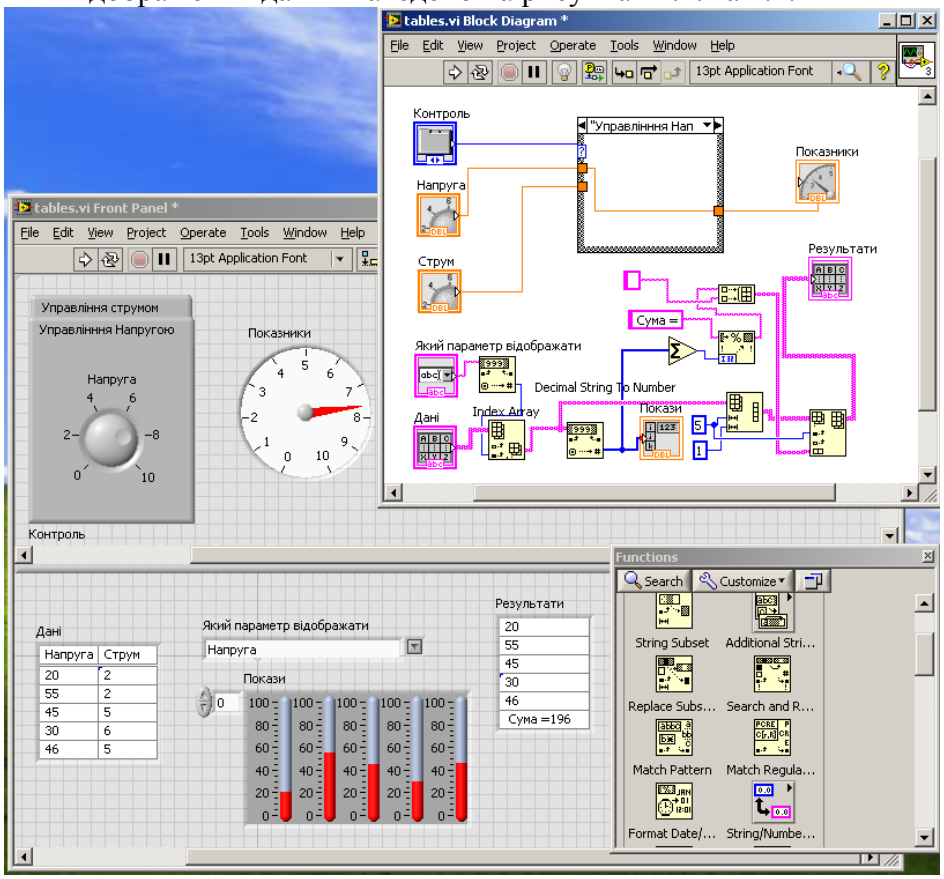


Рис.4.1. Формування та представлення даних системи, зв'язаних з напругою.

Для більш ефективного розділення простору відображення даних, головну екранну форму було поділено на верхню частину для відображення характеристик реального часу та нижню частину для відображення статистичної обробки даних, використовуючи компонент **Hor Splitter Bar** з каталогу **Containers**. Інструмент дозволяє проглядати статистику, прокручуючи повзунки нижньої частини не зачіпаючи верхню, яка має бути весь час перед очима. Враховуючи інші обставини безпеки та ергономіки відображення даних, користувач може самостійно групувати компоненти у вертикальних чи

горизонтальних фреймах екранної форми. Наступним кроком є скорочення інформації та більш ефективне використання основної екранної форми. Зручним компонентом, який може допомогти відображати лише релевантну інформацію є **Tab Control**. Крім того, що він дозволяє організувати доступ до екранних форм по закладках, він може дозволити скоротити кількість терміналів для відображення, як то продемонстровано на прикладі.

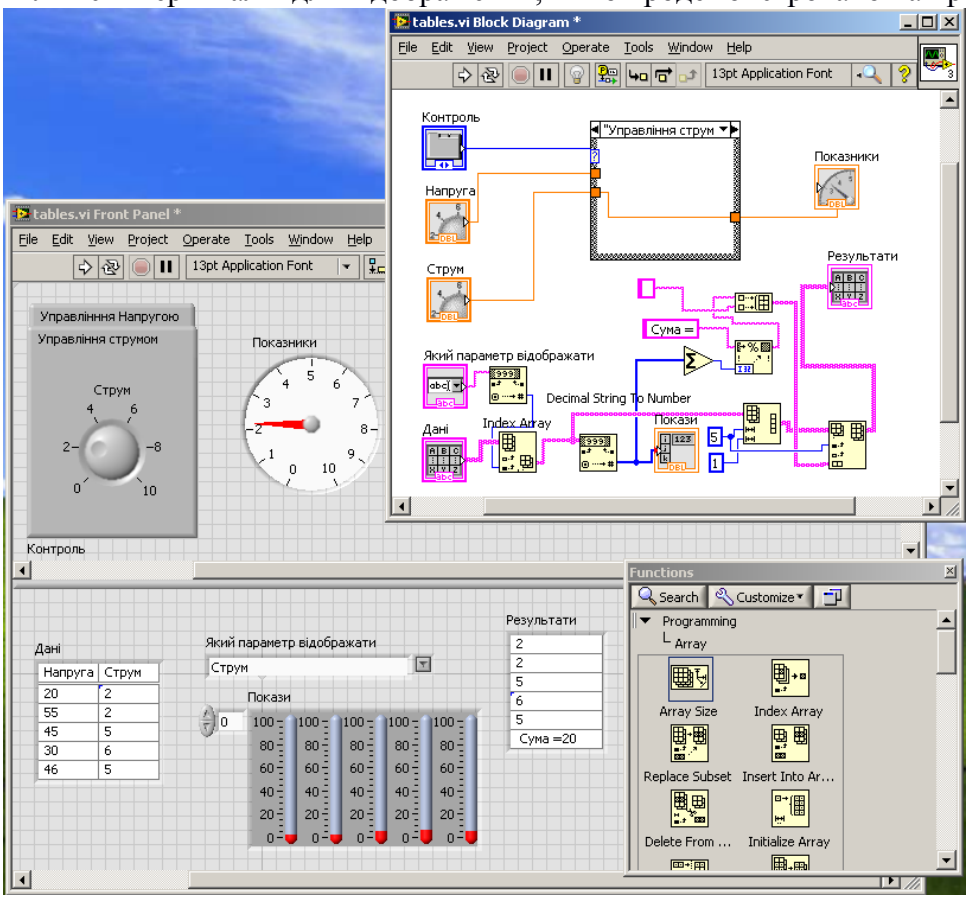


Рис.4.2. Формування та представлення даних системи, зв'язаних із струмом.

При перемиканні вкладок, данні які в них опрацьовуються, передаються в основну програму, відображаючись на приладі “Показники”. Часто роблять і навпаки: в основній формі виставляють режим роботи системи, а її характеристики переглядають по вкладках. Для цього в контекстному меню **Tab Control** треба вибрати опцію **Change to Indicator** і передавати номер режиму (номер вкладки) в компонент.

Нижня частина екрану слугує для відображення статистичних даних, в даному випадку визначених оператором. Зручним способом взаємодії з оператором при впорядкованих структурах даних є табличні форми. В програмному середовищі NI LabVIEW для створення табличних форм існує декілька компонентів, розташованих у каталозі меню **List, Table & Tree**. У додатку використано компонент **Table**, що функціонувати в режимі **control** чи **indicator**, обрані через контекстне меню компоненту. Таблиця у випадку додатку містить 5 значень напруги та струму, тому для зручності у властивостях візуального компоненту було виключено можливість прокрутки і встановлено читку кількість для відображення рядків та стовпчиків. Задачею додатку було перетворити дані одного з параметрів в числові для обробки та відображення “Показів”. Для того, щоб обрати один параметр (струм чи напругу), було використано візуальний компонент **Combo Box**, у властивостях якого, доступних через його контекстне меню, було прописано два параметри стану із зрозумілими символьними позначеннями. Налаштування параметрів наведено на рисунку 4.3.

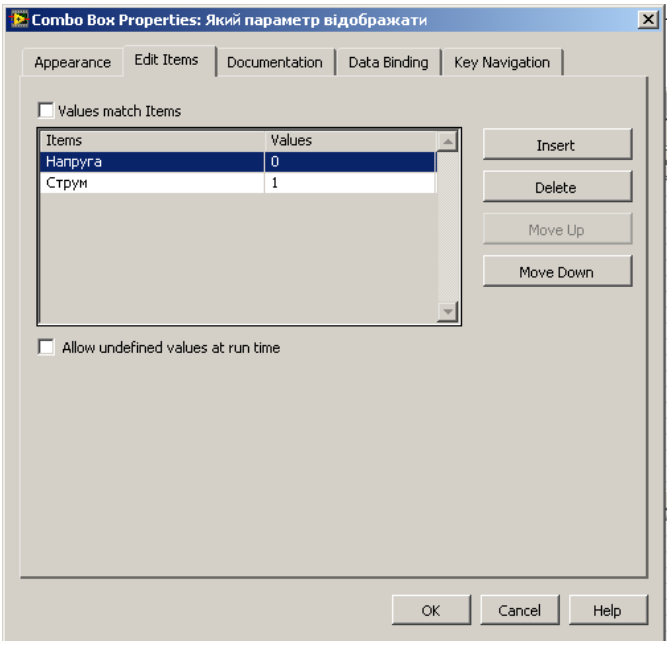


Рис.4.3. Налаштування властивостей компоненту **Combo Box**.

Для зведення результатів по одному параметру було також використано таблицю у режимі індикації із додаванням суми значень вибірки. При цьому було використано компоненти приведення типів даних, роботи із строками та масивами.

При прийнятті рішення оператором, особливо у відповідальних діях, широкого вжитку отримали діалогові вікна. Приклад діалогового вікно, що попереджає про критичну зупинку програми наведено на рисунку 4.4.

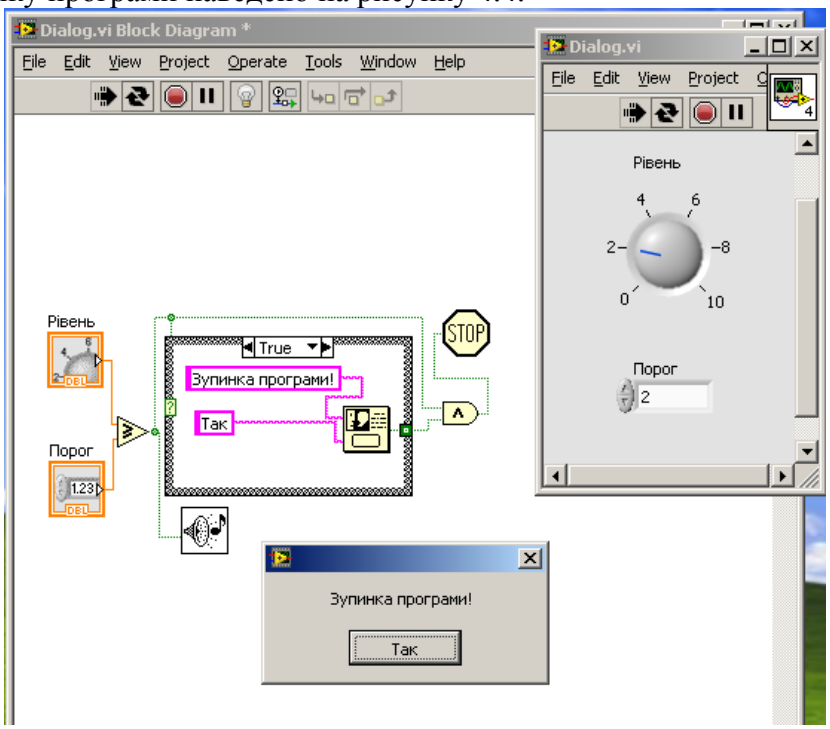


Рис.4.4. Діалог попередження про зупинку програми.

Повідомлення супроводжується системним звуковим сигналом, який додатково привертає увагу користувача. Компонент звукового оповіщення знаходиться у каталозі **Graphics & Sounds**.

Створені програми можуть бути збережені, як підпрограми — нові компоненти що зручно багаторазового використання при виконанні однотипних операцій. Приклад створення підпрограми наведено на рисунку 4.5.

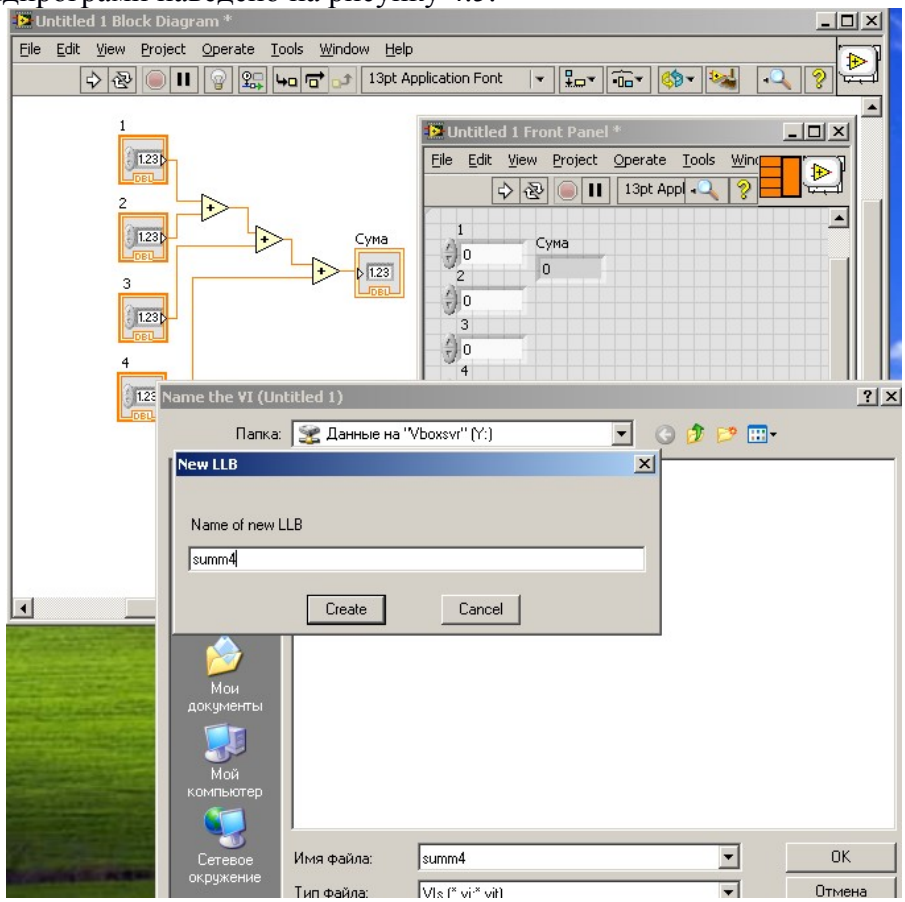


Рис.4.5. Створення підпрограми сумування чисел.

Після створення програми у формі **Front Panel**, у її лівому верхньому кутку є дві квадратні іконки (ліва із білими прямокутниками і права з піктограмою логотипу LabVIEW). Натискаючи правою кнопкою миші на лівій іконці, можна через опцію Patterns змінити кількість та конфігурацію розташування по периметру ніжок майбутнього візуального компоненту — підпрограми. Для того щоб сполучити полюса (ніжки) компоненту з потоками, потрібно лівою кнопкою миші натиснути на обраному прямокутнику в лівій іконці, і після того на потоці, з яким потрібно його асоціювати. При цьому прямокутник зафарбується у колір, який відповідає кольору потоку (типу даних). Операцію слід повторити з усіма прямокутниками і потоками. На рисунку 4.5 ліва іконка отримала помаранчеве забарвлення після означених операцій. Далі за бажанням можна змінити малюнок представлення самого компоненту. Для цього потрібно правою кнопкою миші викликати через контекстне меню правої іконки пункт **Edit Icon**. При цьому з'явиться графічний редактор із базовим зображенням, яке можливо доповнити чи замінити на власне. Після завершення роботи над іконкою, викликаючи діалог зберігання проекту **Save As**, Потрібно обрати в ньому пункт збереження **New LLB** (нова бібліотека) та зберегти під новим ім'ям. Після створення нового проекту, в контекстному меню **Block Diagram** обрати категорію **Select VI** to open та відкрити компонент через діалог. Приклад нової програми з підпрограмою сумування чотирьох чисел наведено на рисунку 4.6. Використовуючи підпрограми в такий спосіб можливо значно пришвидшити створення нових програм, зменшити вірогідність помилок та спростити візуальне представлення в основній форми програмного забезпечення NI LabVIEW.

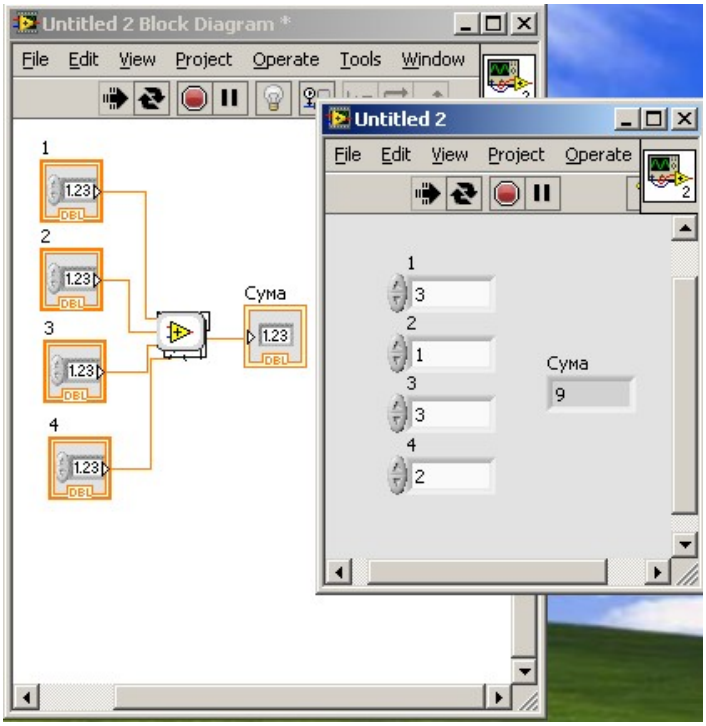


Рис.4.6. Програма із підпрограмою сумування чисел.

Збереження файлів у файлові формати для подальшого використання, це також одна з важливих функцій SCADA. В NI LabVIEW компоненти, що реалізують роботу з файловою системою знаходяться в каталозі **File I/O**. Приклад програмної реалізації запису та зчитування даних файлової системи наведено на рисунку 4.7.

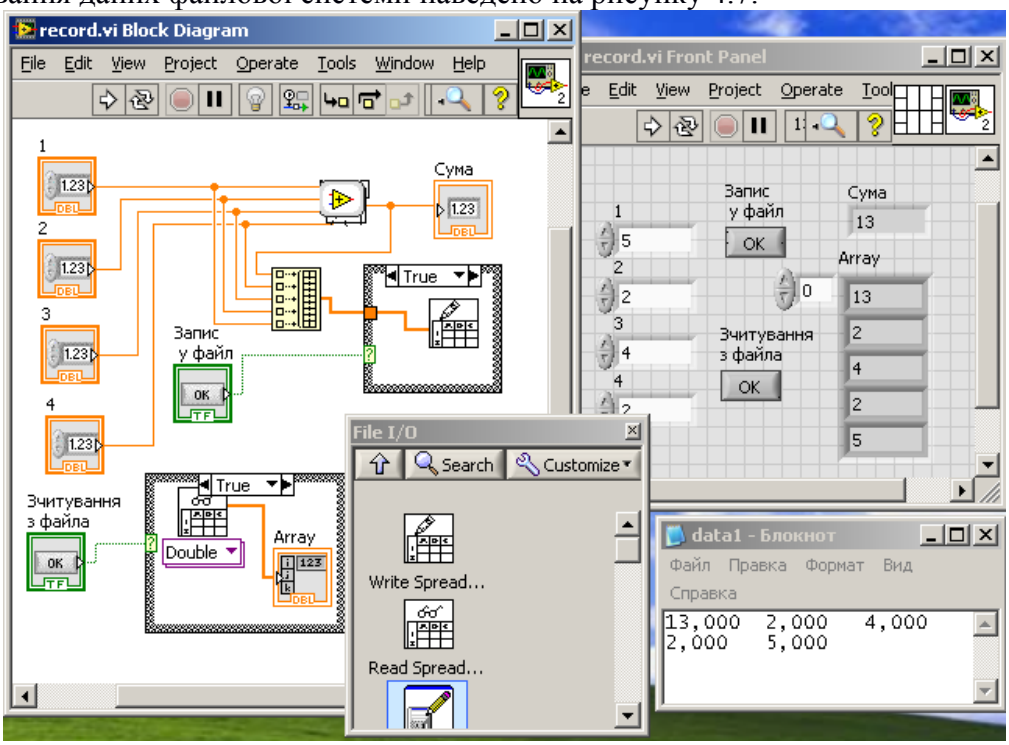


Рис.4.7. Запис та зчитування даних з текстового файлу.

Компонент для запису **Write Spread** та зчитування **Read Spread** потребують упорядкування даних у вигляді одномірного чи двомірного масиву. Якщо не пописувати шлях зберігання чи зчитування, що подається як строкова змінна на вхід блоків, то відкривається стандартний діалог для зазначення чи обирання назви файлу з якім потрібно

проводити операції. У каталозі також присутні інші компоненти для роботи з файлами і даними різних форматів та представлень. Детальніше про їх властивості можна дізнатись через контекстну довідку **NI LabVIEW**.

Кінцева компіляція розробленого додатку реалізується через головне меню програми, пункт меню **Tools**, підпункт **Build Application (exe) from VI**. Після компіляції буде створено виконавчий (exe) файл та додаткові файли конфігурації.

Порядок виконання.

1. Відкрити проект попередньої лабораторної роботи у програмному середовищі LabVIEW.
2. Використовуючи компоненти каталогів **List, Table & Tree** та **Containers**, провести модифікацію інтерфейсу, реалізуючи дві панелі (розділенням), форму із закладками для представлення даних про робочі параметри системи, статичні чи реального часу на власний розсуд. Використати табличні форми для введення чи відображення параметрів роботи системи. Долучити до табличних даних характеристики сумарної потужності (гідравлічної енергії), як суму добутків параметрів струму і напруги (витрат та тиску на насосі). Кількість використаних елементів має бути не меншою, ніж то показано в прикладах.
3. Реалізувати мінімум дві діалогові форми з двома чи трьома варіантами відповіді та зміною алгоритму роботи програми при цьому. Варіант діалогу критичної зупинки, діалогу зміни характеру протікання процесу по досягненню обраних значень параметрів чи власний варіант.
4. Створити мінімум дві підпрограми, замінивши частину програмного коду, що повторюється. Варіантом можуть бути процедури формування та відображення даних (однотипні) та їх математична обробка “по шаблону”. При цьому підпрограми мають мати різу кількість входів та виходів із власними відрисованими іконками.
5. Використовуючи компоненти роботи з файловою системою реалізувати операції запису та зчитування частини чи усіх даних. Варіантом може бути збереження частини послідовності даних масивів, реалізованих у третій лабораторній роботі та зчитування із по місцю статистичної обробки даних, розрахунку потужності тощо.
6. Навести у звіті тему і мету роботи. У порядку виконання роботи навести екранні форми розробленого продукту для кожного пункту виконання та зробити текстовий опис основних елементів, що використовувались, та їх параметрів налаштування, відмінних від значень за замовчуванням.
7. Зробити висновки про особливості застосування та можливості компонентів HMI та роботи з файловою системою.

Контрольні питання.

1. Які переваги дає розбиття екрану на частини з використанням **Splitter Bar**?
2. Чим відрізняється стандартний компонент N**umeric** C**ontrol** (**Edit Box**) від компонента **Combo Box**?
3. Яка структура і тип даних використовується компонентом **Table**?
4. Скільки полюсів має двокнопочний (так, ні) компонент діалогу?
5. Які параметри налаштування є у компоненту **Sound**?
6. Що таке підпрограма і які переваги вона надає при створенні додатку?
7. Як змінити кількість полісів у підпрограмі?
8. Яки структури та типи даних використовуються компонентом **Write Spread**?
9. Які типи даних можна задати для представлення компонентом **Read Spread**?
10. Як створити програмний exe-модуль додатку в **NI LabVIEW**?

ЛІТЕРАТУРА

1. LabVIEW для всех / Джеффи Тревис. Пер. с англ. Клушин Н.А. ДМК пресс. ПриборКомплект, 2005. - 544 с.: ил.

2. Пейч Л.И., Точилин Д.А., Поллак Б.П. LabVIEW для новичков и специалистов. - М. Горячая линия — Телеком, 2004. - 384с.: ил.

3. Суранов А. Я. LabVIEW 7: справочник по функциям. – М.: ДМК Пресс, 2005. – 512 с

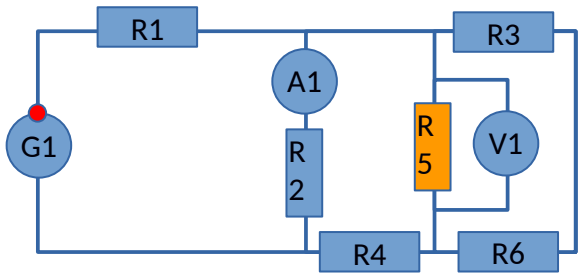
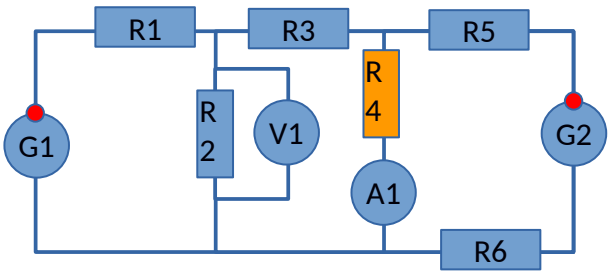
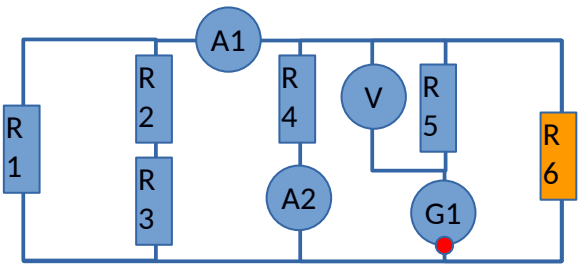
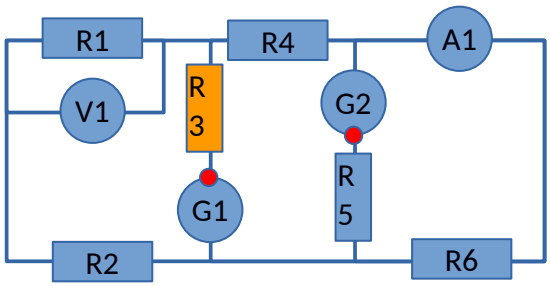
4. Уроки по LabVIEW [Електронний ресурс]: — режим доступу: http://www.picad.com.ua/lesson.htm – Назва з екрана.

ДОДАТОК 1

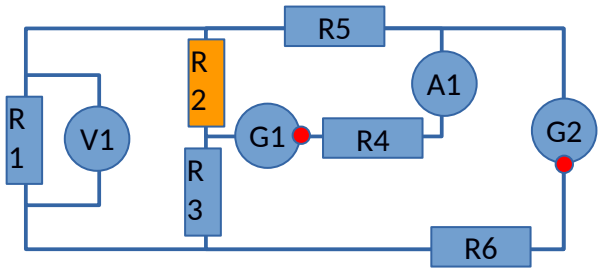
Лабораторний модуль з дисципліни «Основи комп'ютерно-інтегрованого управління» складається з 4 лабораторних робіт, спрямованих на здобуття практичних навичок створення та програмування SCADA в середовищі NI LabVIEW. Студент отримує індивідуальне завдання на розробку, що являє собою конкретну структурну, електричну чи гідравлічну принципові або інші види схем, погоджені з викладачем, із зазначенням вихідних потоків і параметрів до створення програмного додатку засобами LabVIEW, зазначеними у теорії та порядку виконання лабораторних робіт. Програмний продукт удосконалюється по ходу виконання лабораторного практикуму в реалізаціях додаткових задач та програмних компонентів та отриманням кінцевого додатку. Захист лабораторного модуля полягає у складанні письмового звіту та демонстрації розробленого додатку.

Типовими варіантами розробки є автоматизовані робочі місця диспетчеризації змінних параметрів електричної чи гідравлічної системи.

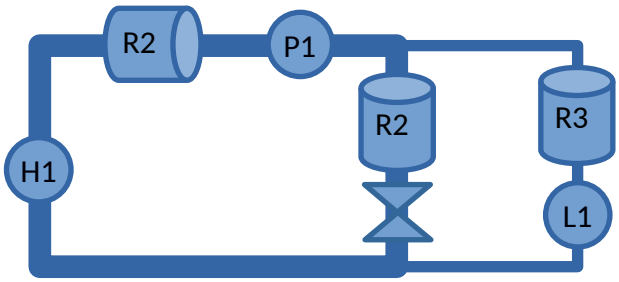
Таблиця Д1: Варіанти схем

1 	2 
3 	4 

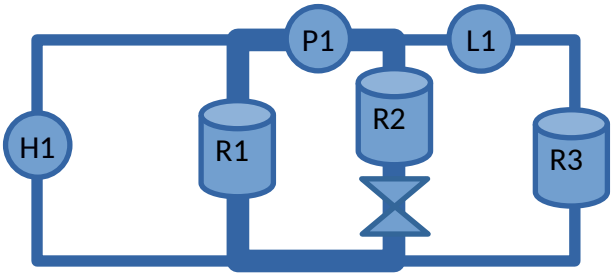
5



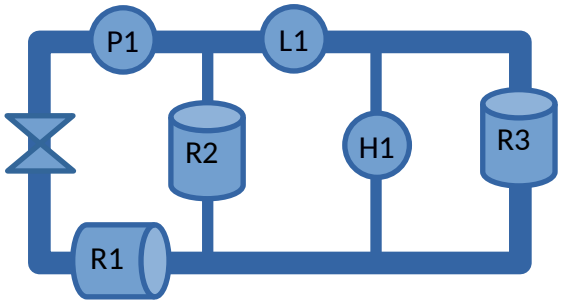
6



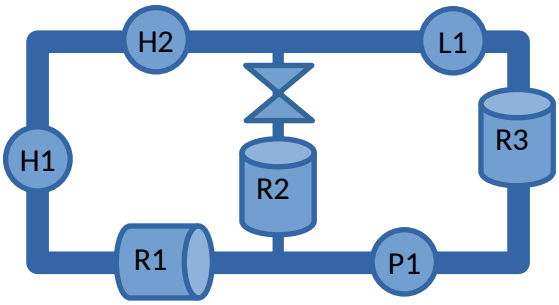
7



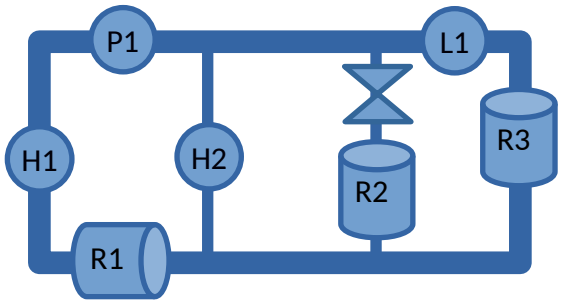
8



9



10



Позначення електричні:
G — генератор напруги із зазначенням напряму протікання струму (симфазність);
R — опір ділянки (помаранчевим кольором позначені елементи, що переривають ланцюг з використанням ключа — тумблера);
A — амперметр;
V — вольтметр;

Позначення гідравлічні:
H — насос; R — радіатор (характеризується усередненим значенням довжини Lr та діаметром перерізом патрубків Dr)
L — витратомір;
P — манометр;
|><| — запораний вентиль, що перериває магістраль, керуємий тумблером;
На схемах позначено лініями товстий трубопровід T1 з діаметром перерізу Dr1 та тонкий T2 з діаметром перерізу Dr2.
Довжини ділянок трубопроводів обрати самостійно, приблизно до масштабу зображення схем.

Таблиця Д2 :Параметри схем:

№	Електрична схема							
	R1 (Ом)	R2 (Ом)	R3 (Ом)	R3 (Ом)	R5 (Ом)	R6 (Ом)	G1 (В)	G2* (В)
1.	2	4	6	4	3	5	20	30
2.	5	7	8	3	5	6	10	35
3.	4	6	4	6	7	8	40	20
4.	3	5	7	4	4	8	10	30
5.	6	4	6	6	3	4	20	80
6.	2	4	5	6	7	8	30	40
7.	3	4	6	4	6	3	25	35
8.	4	7	6	5	3	5	55	45
9.	4	6	7	3	4	6	20	90
10.	4	5	6	3	6	5	10	70

Примітка 1*: якщо у наданому варіанті на схемі відсутнє друге джерело, прийняти G2 = 0.

Таблиця Д3 :Параметри схем:

№	Гідравлічна схема							
	R1	R2	R3	T1	T2	H1	H2**	
	Lr/Dr (м)	Lr/Dr (м)	Lr/Dr (м)	Dr1 (м)	Dr2 (м)	(кПа)	(кПа)	
1.	2/0.2	3/0.2	9/0.1	0.05	0.01	2000	800	
2.	3/0.1	2/0.15	8/0.3	0.03	0.02	3000	400	
3.	5/0.1	4/0.1	3/0.2	0.02	0.01	900	300	
4.	3/0.2	7/0.2	2/0.15	0.015	0.005	700	500	
5.	2/0.15	6/0.1	4/0.1	0.04	0.03	1100	400	
6.	4/01	2/0.2	7/0.2	0.02	0.01	1300	300	
7.	7/0.02	3/0.1	6/0.1	0.04	0.02	1600	700	
8.	6/0.1	5/0.1	7/0.25	0.04	0.03	1400	1000	
9.	5/0.25	8/0.1	6/0.1	0.05	0.03	1500	1100	
10.	3/0.3	2/0.08	2/0.2	0.04	0.01	1700	500	

Примітка 2**: якщо у наданому варіанті на схемі відсутній другий насос, прийняти H2 = 0. Насоси працюють узгоджено (в одному напрямку), напрямок обирається довільно.

ДОДАТОК 2

Приклад розрахунку параметрів електричних та еквівалентних гідравлічних кіл методом контурних струмів (потоків).

На рисунку 1. зображено 3 контури А,Б,В, множина опорів R та джерел енергії Е.

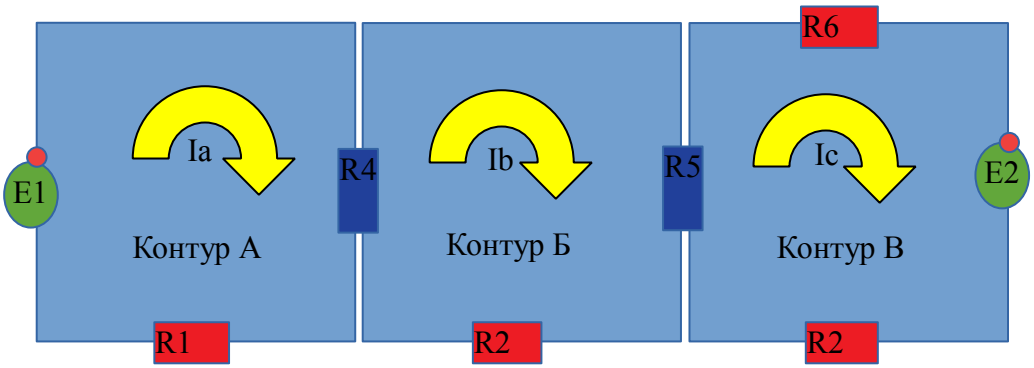


Рис Д1. Складне коло

Відповідно до обраних напрямків протікання контурних струмів Ia, Ib, Ic за годинниковою стрілкою, визначимо матрицю опорів відповідно до наступних правил:
1) Арифметично сумуємо опори кожного контуру, записуємо на перетинах відповідних стовпчиків-рядків матриці: А-А, Б-Б, В-В;

Проміжна матриця Д.

Контури	А	Б	В
А	R1+R4		
Б		R2+R4+R5	
В			R2+R5+R6

2) Сумуємо опори, що знаходяться на дотиканні контурів А-Б, А-В, Б-В і записуємо їх з від'ємними знаками у перетинах відповідних стовпчиків-рядків матриці: А-Б, А-В, Б-В, Б-А, В-А, В-Б;

Матриця Д.

Контури	А	Б	В
А	R1+R4	-R4	0
Б	-R4	R2+R4+R5	-R5
В	0	-R5	R2+R5+R6

Якщо контури не дотикаються, у перетин записуємо 0.

Визначимо стовпчик джерел потенційної енергії (електричної чи гідравлічної) для кожного контуру.

Сумуємо алгебраїчно джерела енергії в кожному контурі, з урахуванням того, що якщо полярність джерела збігається з напрямком контурного струму — до стовпчика заносимо із знаком “+”, якщо не збігається — із знаком “-”. Якщо в контурі джерело відсутнє, записуємо 0.

Стовпчик К.

А	E1
Б	0
В	-E2

Розраховуємо детермінант матриці Д (матриця 3х3), detД

Підставляємо замість першого стовпчика матриці Д стовпчик К та розраховуємо детермінант матриці Д1, detД1:

Матриця Д1.

Контури	А	Б	В
А	E1	-R4	0
Б	0	R2+R4+R5	-R5
В	-E2	-R5	R2+R5+R6

Підставляємо замість другого стовпчика матриці Д стовпчик К та розраховуємо детермінант матриці Д2, detД2:

Матриця Д2.

Контури	А	Б	В
А	R1+R4	E1	0
Б	-R4	0	-R5
В	0	-E2	R2+R5+R6

Підставляємо замість третього стовпчика матриці Д стовпчик К та розраховуємо детермінант матриці Д3, detД3:

Матриця Д3.

Контури	А	Б	В
А	R1+R4	-R4	E1
Б	-R4	R2+R4+R5	0
В	0	-R5	-E2

Тоді контурні струми за методом Крамера знайдуться наступним чином:

Ia = detД1/detД, Iб = detД2/detД, Iв = detД3/detД.

Струми у незалежних гілках контурів будуть визначатися контурними струмами, а у залежних (дотичних) їх алгебраїчною сумою (різницею).

Для визначення визначників в середовищі LabView, скористуйтесь користувацькою бібліотекою Determ1.lbb. Для розрахунку, схема підключення терміналів до компоненту, що міститься у бібліотеці наступна:

X1	X4	X7
X2	X5	X8
X3	X6	X9

Приклад інтерфейсу і фрагменту програми наведено на рисунку Д2.

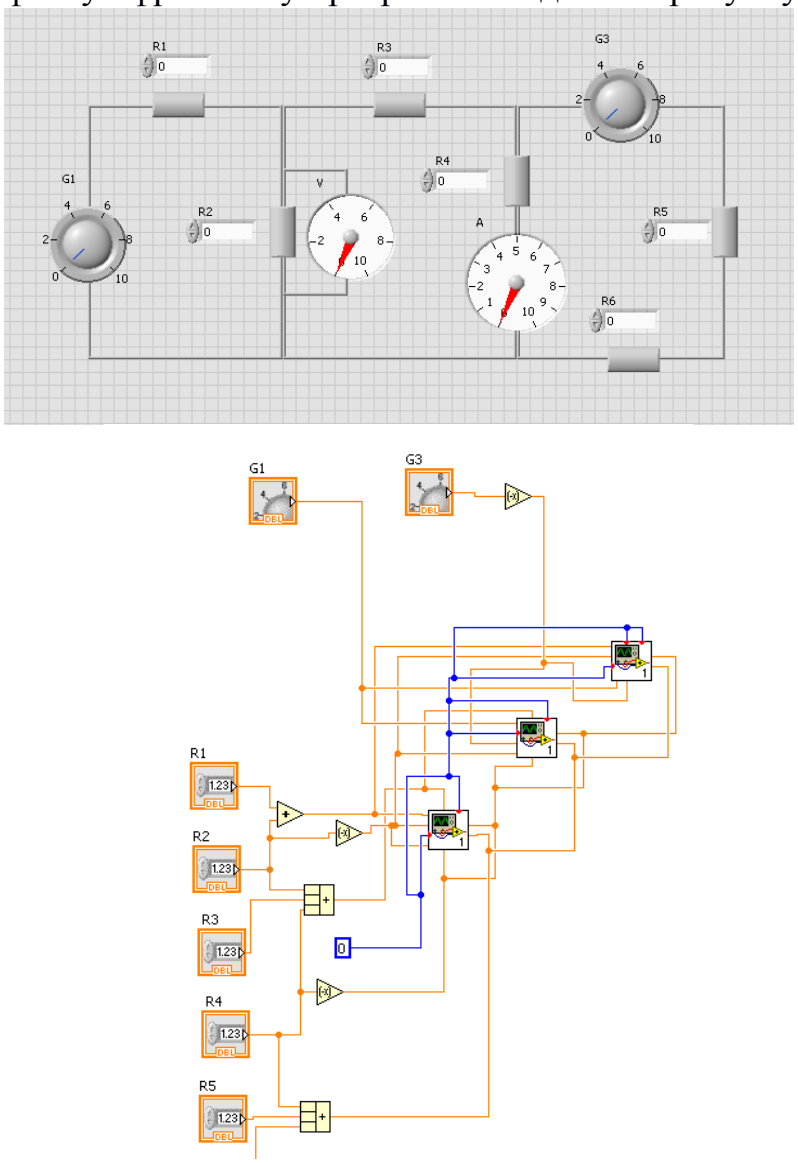


Рис.Д2. Інтерфейс та фрагмент програми віртуального приладу.