МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК

КВАЛІФІКАЦІЙНА МАГІСТЕРСЬКА РОБОТА

на тему:

«Інформаційна технологія аналізу даних вимірювань з надпровідних ланцюгів»

Завідувач
випускаючої кафедри **Довбиш А. С.**

Керівник роботи **Кузіков Б. О.**

Студента групи ІН.м – 81н **Сиротенко С. Г.**

СУМИ 2020

Сумський державний університет

(назва вузу)

Факультет *ЕлІТ* Кафедра *Комп'ютерних наук*

Спеціальність *«Інформатика»*

Затверджую:

зав.кафедрою _____

_____

"_____" _____20___р.

ЗАВДАННЯ
НА ДИПЛОМНИЙ ПРОЕКТ СТУДЕНТОВІ

Сиротенко Сергію Геннадійовичу

(прізвище, ім'я, по батькові)

1. Тема проекту (роботи) *Інформаційна технологія аналізу даних вимірювань з надпровідних ланцюгів*

затверджую наказом по інституту від "____"_____20___ р. № _____

2. Термін здачі студентом закінченого проекту (роботи)

_____

3. Вхідні данні до проекту (роботи)

_____

_____

_____

_____

4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити)

1) Аналіз проблеми. Постановка задачі дослідження. 2) Надпровідні ланцюги 3) Пристрої Superconducting Quantum Interference Devices. 4) Розробка інформаційного та програмного забезпечення системи.

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

_____
_____
_____
_____
_____

6. Консультанти до проекту (роботи), із значенням розділів проекту, що стосується їх

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання _____

Керівник _____

(підпис)

Завдання прийняв до виконання _____

(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів дипломного проекту (роботи)	Термін виконання проекту (роботи)	Примітка
1.	*Аналіз проблеми. Постановка задачі дослідження*	*14 днів*	
2.	*Надпровідні ланцюги. Пристрої Superconducting Quantum Interference Devices*	*14 днів*	

3.	*Вибір програмного засобу рішення поставленої задачі*	*7 днів*	
4.	*Розробка інформаційного та програмного забезпечення системи*	*120 днів*	
5.	*Тестування та виправлення помилок*	*21 день*	
6.	*Оформлення пояснювальної записки до дипломної роботи*	*7 днів*	

Студент – дипломник _____

(підпис)

Керівник проекту _____

(підпис)

РЕФЕРАТ

Записка: 54 стор., 33 рис., 1 додаток, 14 джерел.

Об'єкт дослідження — надпровідні ланцюги (з'єднання Джозефсона) в умовах наявності ефекту шуму та резонансу.

Мета роботи — розробка програмного забезпечення для зчитування й обробки сигналів з надпровідних ланцюгів в креогеному обладнанні.

Методи дослідження — методи проектування.

Результати — розроблено гнучку систему враховуючи умови модульності. Результати роботи дозволяють зчитувати дані з криогенного обладнання за допомоги обладнання National Instruments, обробляти їх в режимі near-realtime та робити більш глибокий аналіз. Також розроблене програмне забезпечення дозволяє витягувати з експериментальних даних необхідну інформацію для симуляції за допомоги JSim. Також в результаті було отримано опис та схеми архітектурних рішень. Обробка цих даних дозволить скласти додаткові обмеження і допущення, які необхідно враховувати при подальших дослідженнях.

НАДПРОВІДНІ ЛАНЦЮГИ,
З'ЄДНАННЯ ДЖОЗЕФСОНА, МОДУЛЬНІСТЬ,
NATIONAL INSTRUMENTS,
IV-CURVE, JSIM, PYTHON

Зміст

Вступ

Аналіз сучасних літературних джерел відносно розробки та впровадження приладів, в основі яких лежить технологія надпровідності, свідчить про значне зростання уваги до цього питання.

Майже кожна команда дослідників в даній області розробляють програмне забезпечення вивчення поведінки надпровідних ланцюгів окремо, що призводить до зниження рівня кооперації. Така структура роботи призводить до складнощів використання, поліпшення та інтеграції програмних додатків, а іноді і унеможливлює це.

Розробка єдиної основи (бази) програмного забезпечення з використанням модульного підходу є рішенням, що спонукає підвищенню рівня сумісної роботи команд і в одно час ефективності досліджень.

При такому підході кожна команда дослідників може створити власні модулі, які можуть бути унікальними або включати вже створені власні або сторонні розробки. Єдиним обмеженням для таких модулів є необхідність використання узгодженого інтерфейсу, що забезпечує обмін даними.

Планується, що розробники зможуть розширювати створюванні модулі, використовувати розробки інших команд та формувати власне, унікальне програмне забезпечення для конкретних цілей без перевантаження зайвим функціоналом.

Таким чином, завданням даного дослідження є:

- розробка програмного забезпечення, яке буде задовольняти вимогам великої гнучкості і модульності, що може використовуватися в наукових цілях для вивчення надпровідних ланцюгів.
- реалізувати функціональне тестування.
- створення опису та схем архітектурних рішень, що дозволять скласти додаткові обмеження і допущення, які необхідно враховувати при подальших дослідженнях

- проведення оцінки доцільності впровадження даного ПЗ та визначення напрямку подальших досліджень.

Виділимо допущення і обмеження:

- Можливість отримання даних з декількох девайсів обмежена самими девайсами (ця можливість не була передбачена виробниками). Такий функціонал підтримують лише останні моделі девайсів, які на даний час не є широко розповсюдженими.
- Зростання використання оперативної пам’яті не повинно залежати від кількості часу роботи програмного забезпечення.

1. Аналітичний огляд існуючих рішень

Даний розділ присвячений огляду програмних засобів для визначення основних потреб користувачів, складання фундаментального списку функціональних можливостей, виділення сильних і слабких сторін вже розробленого програмного забезпечення. Також був проведений аналіз існуючих пакетів і програм перед фазою розробки необхідних алгоритмів, що дозволяє скоротити час на розробку програмних і графічних інтерфейсів та сфокусуватися на поставленій меті. Аналіз було проведено серед програмного забезпечення, що дозволяє взаємодіяти с обладнанням National Instruments (NI).

1.1. Огляд існуючих рішень

Застосування надпровідної цифрової електроніки (також відомої під назвою Single-Flux-Quantum (SFQ)) варіюється від біомедичного аналізу до досліджень геофізики, космічних телекомунікацій, надчутливих приймачів сигналів та суперкомп'ютерів [1]. Від напівпровідникової електроніки її відрізняють такі особливості:

- потребує умов надпровідності;
- використовує поодинокі кванти магнітного потоку для подання цифрової інформації;

Відсутність електричного опору дозволяє створювати логічні схеми з високою швидкодією, а останні розробки відрізняються і високою енергоефективністю. Надпровідна логіка є варіантом для створення процесорів, з високою частотою перемикання окремих логічних елементів (до сотень ГГц). За оцінкою на червень 2011 року комп'ютер, побудований на традиційній логіці, повинен споживати близько 0,5 гігават енергії, тоді як комп'ютер на основі енергоефективної надпровідної логіки міг би мати в 10-100 разів менше енергоспоживання [2].

Пристрої, які використовують дану технологію, називаються Superconducting Quantum Interference Devices (SQUID) і широко

використовуються у багатьох проектах, що вимагають надзвичайної чутливості до магнітного потоку. Така технологія в вимірювальній техніці демонструє зазвичай більш високу чутливість за рахунок більш високої трансформації потоку від вимірювального об'єму [3]. SQUID на змінному струмі дешевше і простіше у виробництві в малих кількостях. Значна частина експериментів у фундаментальній фізиці і вимірювань в біомагнетізма, включаючи вимір надмалих сигналів, були виконані з використанням SQUID саме на змінному струмі.

В основі SQUID лежить з’єднання Джозефсона (Josephson junction). Через те, що розвиток даної області не дозволяє створити стабільне з’єднання Джозефсона, всі розробки на даний час мають залежну від часу нелінійну поведінку. І тому, цей нелінійний елемент обмежує розробників надійно передбачити поведінку пристроїв [4]. Однією з основних задач в даній області є розроблення стабільного SQUID, що є надпровідним ланцюгом, а саме подвійним з’єднанням Джозефсона.

Для розробників дуже важливим та інформативним є вимірювання залежності опору від сили струму, тобто, отримання IV-кривих (IV-curve). При цьому, однією з перешкод є те, що на практиці вимірювання надпровідних квантових перешкод постійного струму (DC-SQUID) можуть показувати несподівані резонансні поведінки, що знижують їх продуктивність [5].

Таким чином, отримання та моделювання IV-кривих, з урахуванням того, що дані містять ефект шуму та резонансу, є одним з основних завдань на сьогоднішній день.

1.2. Аналоги готових продуктів

1.2.1. Отримання даних

Основним обладнанням, яке використовується для отримання даних є обладнання компанії NI. Популярними є такі сімейства продуктів NI, як пристрої збору даних (data acquisition), приладові інтерфейси (instrument control), системи керування електроприводами і машинного зору (machine vision) [6].

Найрозповсюдженішим програмним забезпеченням, що дозволяє отримати дані з обладнання є LabVIEW (Laboratory Virtual Instrumentation Engineering Workbench) (Рис. 1.1). Це середовище розробки і платформа для виконання програм, створених на графічній мові програмування «G». LabVIEW використовується в системах збору та обробки даних, а також для управління технічними об'єктами і технологічними процесами.

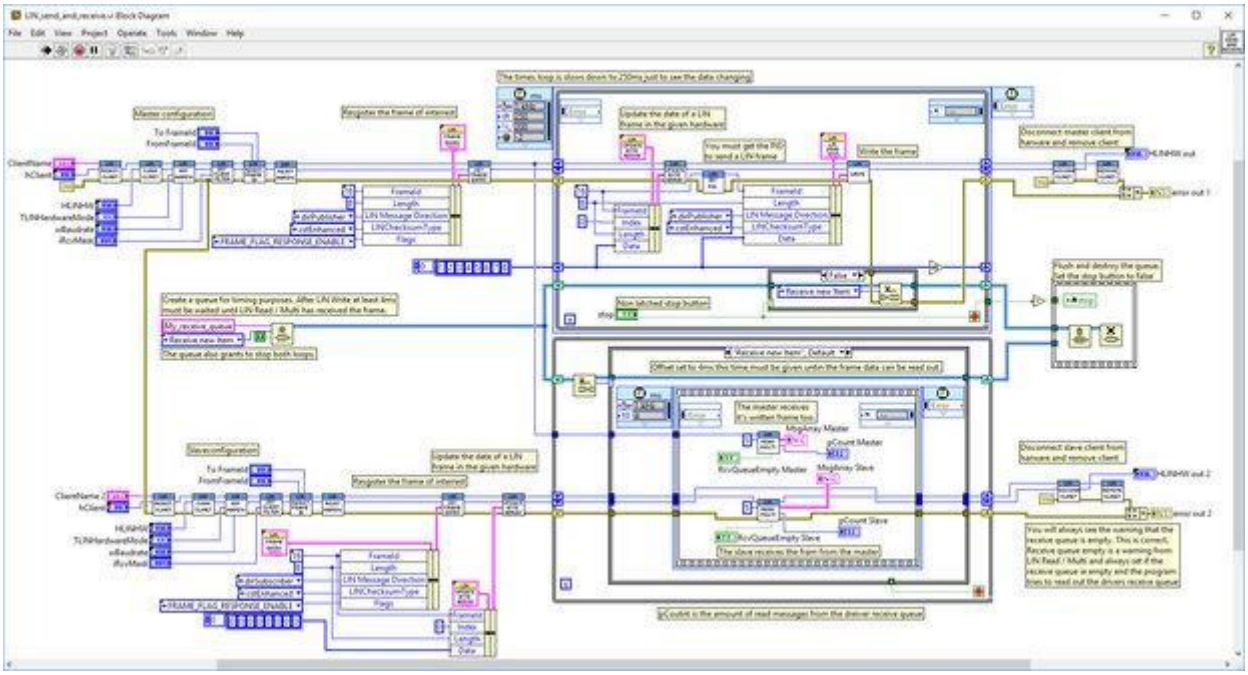


Рисунок 1.1. – Приклад програми LabVIEW

При великих дослідженнях наявність графічної мови програмування та неможливість використання інших мов програмування є основним недоліком. Насправді, людський мозок не здатен охопити та утримувати створені схеми, що інколи набувають розміру декілька десятків екранних просторів.

Також, оперативна пам'ять, що необхідна LabVIEW, може істотно збільшитись при отриманні даних на великому обсязі часу та обробці декількох сигналів одночасно.

1.2.2. Візуалізація даних

В якості візуалізації та аналізу даних дослідники використовують сторонні бібліотеки:

Matplotlib – це найпопулярніша бібліотека для візуалізації даних, що написана на мові Python. Є простою у використанні та не потребує великого порогу входження. Але це стосується лише простих графіків.

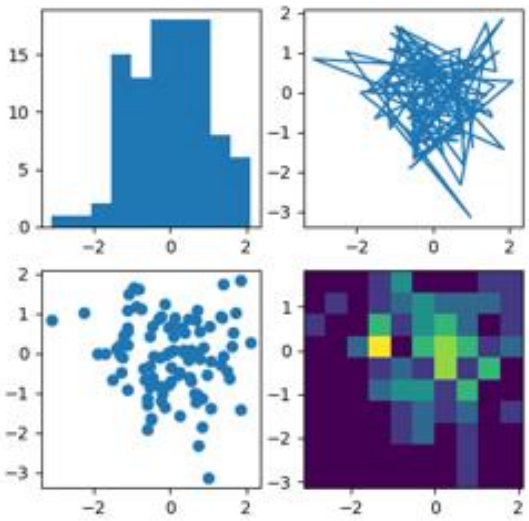


Рисунок 1.2. – Приклад візуалізації даних за допомоги Matplotlib

Налаштування даних, параметрів і графіків для кожного нового проекту займає багато часу. Matplotlib має багато неочевидних особливостей, які незадокументовані. Тому для побудови складних графіків цей інструмент не придатний [7].

Plotly – бібліотека з відкритим вихідним кодом. Основною особливістю цього продукту є те, що він дозволяє створювати необмежену кількість графіків в автономному режимі, а також до 25 діаграм онлайн. Plotly більш потужний, ніж Matplotlib, але має таку ж проблему: налаштування і створення графіків в ньому

займає багато часу, є менш інтуїтивно зрозумілою та потребує значно більший поріг входження [8].

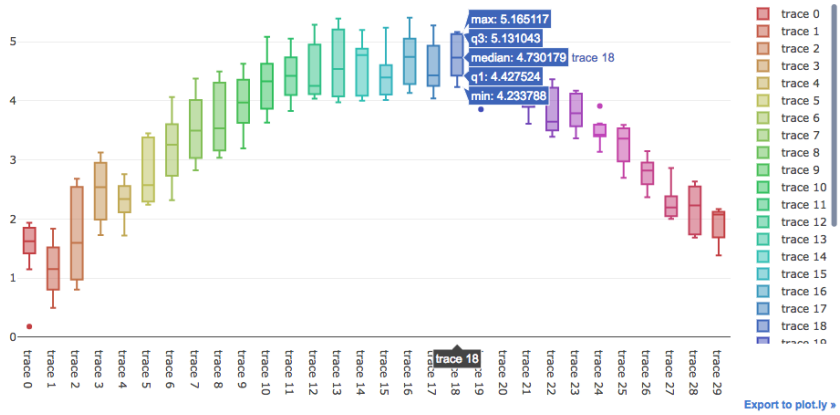


Рисунок 1.3. – Приклад візуалізації даних за допомоги Plotly

Також, серед мінусів можна виділити наступні:

- потрібно ключ API і реєстрація, а не просто установка через pip;
- побудова графіків, які є унікальними для Plotly;
- потребує багато коду;

Серед переваг були відмічені наступні:

- можливість редагування графіки на веб-сайті Plotly, а також в середовищі Python;
- багато функціоналу підтримки інтерактивних графіків, панелей;

Pygal – менш відома бібліотека для візуалізацій, яка використовує граматику графіки для побудови зображень. Це відносно простий пакет через те, що всі об'єкти на графіку дуже примітивні.

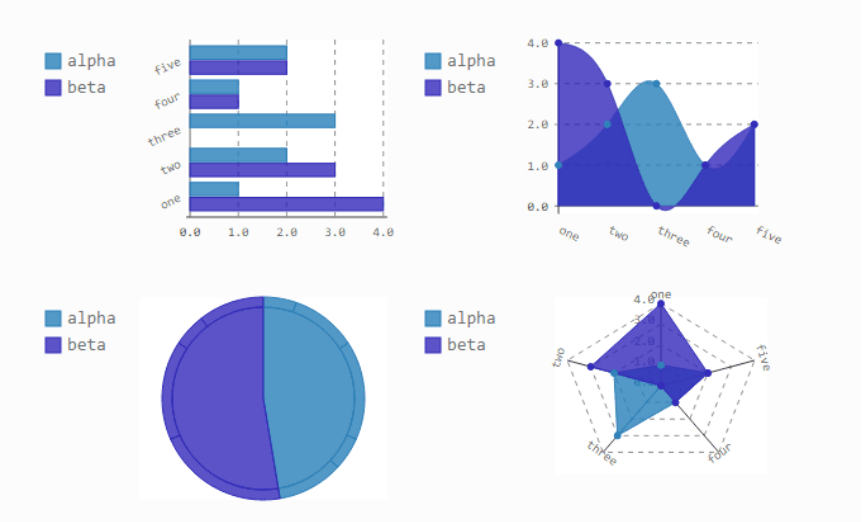


Рисунок 1.4. – Приклад візуалізації даних за допомоги Pygal

Основною проблемою є рендерінг графіків. Необхідно зберегти згенероване зображення в окремий файл, а потім його відкрити, щоб побачити результат [9].

В даній роботі в якості бібліотеки побудови графіків використовується **PyQtGraph**. Цей інструмент має не низький поріг входження, але є інтуїтивно зрозумілим для розробників на мові Python.

Основні риси:

- 2D та 3D-графіки.
- Дані можна переміщувати / масштабувати мишею.
- Відображає більшість типів даних (int або float; будь-яка бітова глибина; RGB, RGBA або яскравість).

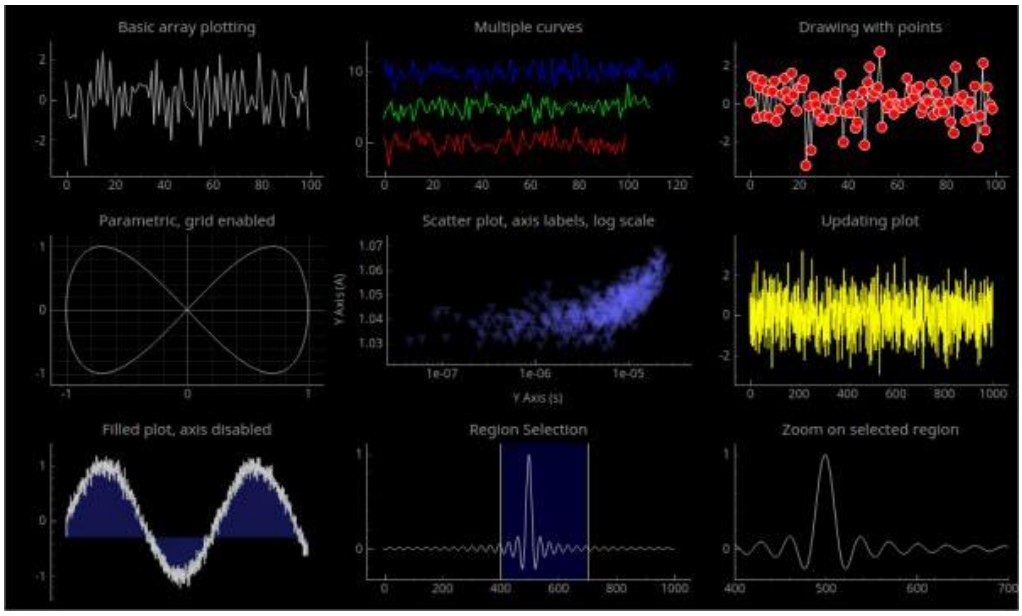


Рисунок 1.5. – Приклад візуалізації даних за допомоги PyQtGraph

Ця потужна бібліотека має багато вбудованих видів графіків, які легко налаштовуються. Більше детально причини вибору саме PyQtGraph розкриті в розділі 1.3.

1.2.3. Симуляція даних

JSim – це імітаційна система на основі Java для розробки моделей, проектування експериментів та оцінки гіпотез за допомогою тестування модельних рішень на основі даних. Основна увага JSim приділяється фізіології та біомедицині, однак його обчислювальний двигун є досить загальним і застосований до широкого кола наукових сфер.

Вона призначена для інтерактивного, ітеративного маніпулювання кодом моделі, обробки декількох наборів даних та наборів параметрів, а також для порівняння різних моделей, що працюють одночасно або окремо. Інтерфейс призначений для написання моделі, визначення вхідних функцій, підбору алгоритмів, що розв'язують диференціальні рівняння.

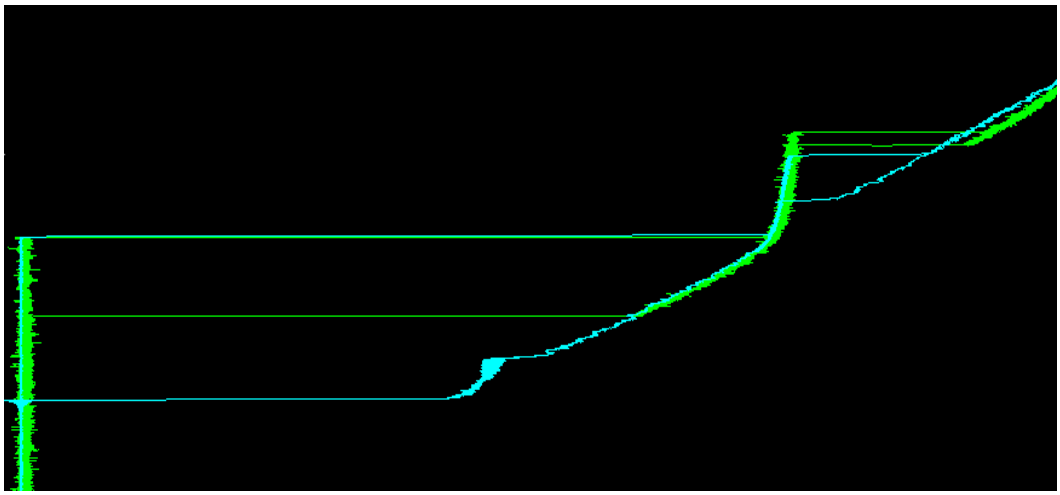


Рисунок 1.6. – Приклад роботи JSim. Зелений графік – актуальні дані вимірювання, блакитний – дані, отримані за допомоги JSim

Для опису моделі використовується власна мова математичного моделювання. Моделі JSim насамперед складаються з оголошення фізичних одиниць, оголошення змінних та оголошення обмежень для цих змінних. Компілятор JSim робить широкі перехресні перевірки, щоб переконатися, що моделі не є Over-defined або Under-defined. Ця перевірка виявляє безліч поширених помилок і тому полегшує та прискорює створення моделей. Це робить JSim дуже зручним інструментом для моделювання та аналізу моделей дослідження [10].

1.3. Вибір методу вирішення задачі

Для подолання описаних вище проблем було вирішено дослідити та розробити програмне рішення на високорівневій мові програмування Python 3.6. Ця мова орієнтована на підвищення продуктивності розробника і читання коду. Синтаксис ядра Python мінімалістичний. У той же час стандартна бібліотека включає великий обсяг корисних функцій. Python підтримує структурний, об'єктно-орієнтоване, функціональне та імперативне програмування. Основні архітектурні риси – динамічна типізація, автоматичне керування пам'яттю, механізм обробки виключень, підтримка багатопоточних обчислень, високорівневі структури даних. Підтримується розбиття програм на модулі, які, в свою чергу, можуть об'єднуватися в пакети.

Через те, що деякі специфічні операції обробки даних (цих операцій не має в існуючих пакетах обробки даних) вимагають великих обчислювальних потужностей, а вбудовані можливості Python не дозволяють виконувати ці операції швидко, було вирішено реалізувати їх у вигляді окремої динамічні бібліотеки на основі мови програмування C (DLL) [11-12].

Для дослідження надпровідних ланцюгів використовується обладнання NI. Для зв’язку з цим обладнанням існують драйвери NI-DAQmx [6], які дозволяють керувати апаратними засобами National Instruments. Для забезпечення зв’язку з драйверами використовується бібліотека NI-DAQmx -Python.

В процесі дослідження було виявлено, що дуже часто виникає необхідність не тільки в отриманні або посиланні даних на обладнання, але й у керуванні цим самим обладнанням. Для цих цілей використовується протокол VISA, що вимагає використання бібліотеки ctypes [13].

В якості редактору коду було обрано інтегроване середовище розробки PyCharm. Це потужне середовище розробки має багатий функціонал навіть у безкоштовній версії Community Edition, яке і використовувалося.

Для створення графічного інтерфейсу користувача використовується Qt, багатоплатформовий фреймворк для розробки програмного забезпечення на мові програмування C++. Його було обрано через те, що існують «прив'язки» до багатьох інших мов програмування, серед яких є і Python (PyQt).

SciPy – відкрита бібліотека високоякісних наукових інструментів для мови програмування Python. SciPy містить модулі для оптимізації, інтегрування, спеціальних функцій, обробки сигналів, обробки зображень, генетичних алгоритмів, розв'язування звичайних диференціальних рівнянь та інших задач, які розв'язуються в науці і при інженерній розробці.

Візуальний аналіз даних вимагає від розробники специфічних маніпуляцій з даними. Можливості LabVIEW в даному випадку не можуть в повній мірі задовольнити потреби користувачів, до того ж, використання стороннього ПЗ не є зручним. Тому було вирішено інтегрувати побудову та управління

візуалізацією даних в розроблювальне програмне забезпечення. В якості бібліотеки для побудови графіків використовується PyQtGraph. Ця бібліотека базується на PyQt4 / PySide та numpy. Вона призначена для використання в математичних, наукових та технічних програмах. Незважаючи на те, що цілком написана на Python, бібліотека дуже швидка завдяки використанню numpy та модулю QtGraphicsView для швидкого відображення. До того ж, ця бібліотека дуже добре співпрацює з фреймворком Qt, який було обрано для побудови графічного інтерфейсу [14].

1.4. Постановка задачі

В даній роботі вирішується завдання розробки об'єктно-орієнтованого гнучкого програмного забезпечення для обробки одно- та багатоканального прийому сигналу та пост-обробки даних. Необхідно реалізувати обробку шуму та обробки даних в режимі near-realtime.

Для цього необхідно вирішити наступні завдання:

1. Реалізувати модуль отримання даних.
    a. Можливість самостійно обирати з якого обладнання та з яких каналів отримати дані.
    b. Аналіз налаштувань користувача та вивід повідомлень про помилки та застереження.
    c. Можливість зберегти дані.
2. Реалізувати модуль обробки даних під час отримання сигналів.
    a. Користувацькі правила обробки даних.
    b. Можливість «вимкнути» даний модуль задля підвищення продуктивності системи.
    c. Можливість зберегти дані.
3. Реалізувати модуль відображення даних.
    a. Можливість обрати стиль побудови графіків.

   b. Можливість «вимкнути» даний модуль задля підвищення продуктивності системи.
   c. Можливість зберегти відображені дані.
4. Модуль пост-обробки.
   a. Можливість використання симуляційної системи JSim.
   b. Можливість обрати користувацькі налаштування для JSim.
   c. Можливість зберегти дані.
5. Реалізувати програмне забезпечення таким чином, щоб зберігалася можливість його використання без графічного інтерфейсу (передбачити можливість інтерфейсу командної строки).
6. Збереження налаштувань як для всієї програми, так і для окремих модулів.
7. Всі дані про помилки, які виникли під час роботи програми, повинні записуватись в окремий файл.

2. Опис технології

2.1. Основні поняття

Single-Flux-Quantum SFQ – це технологія створення електроніки, заснована на квантових ефектах (ефекті Джозефсона) в надпровідних пристроях.

Superconducting Quantum Interference Devices (SQUID) – надчутливі магнітометри, використовувані для вимірювання дуже слабких магнітних полів.

DC-SQUID – робота SQUID на змінному струмі.

Ефект Джозефсона - явище протікання надпровідного струму через тонкий шар діелектрика, що розділяє два надпровідника.

IV-curve – дані залежності напруги від струму.

Python – високорівнева мова програмування загального призначення.

Мова програмування C – універсальна, процедурна, імперативна мова програмування загальна призначення.

Dynamic Link Library (DLL) – в операційних системах Microsoft Windows динамічна бібліотека, що дозволяє багаторазове використання різними програмними додатками.

Драйвера NI-DAQmx – надають підтримку клієнтам, які використовують пристрої збору даних NI.

Бібліотека ctypes – сумісна з типом даних C і дозволяє викликати функції в DLL.

Протокол VISA – стандартизований інтерфейс введення-виведення в області тестування і вимірювань для управління приладами з персонального комп'ютера.

SciPy – бібліотека для мови програмування Python з відкритим вихідним кодом, призначена для виконання наукових і інженерних розрахунків.

PyQt – набір «прив'язок» графічного фреймворку Qt для мови програмування Python, виконаний у вигляді розширення Python.

JSim – являє собою симуляційну систему на основі Java для побудови та аналізу моделей.

Netlist – текстове представлення інформації про модель для JSim (в даній роботі – електричні з'єднання, що містять компоненти електронного пристрою).

Root Mean Square Error – є вимірювачем відмінностей між значеннями (значення вибірки або сукупності), що передбачені моделлю, і фактичними значеннями.

2.2. Абстрактна модель

Основна ідея проекту представлена графічній формі на Рис. 2.1.

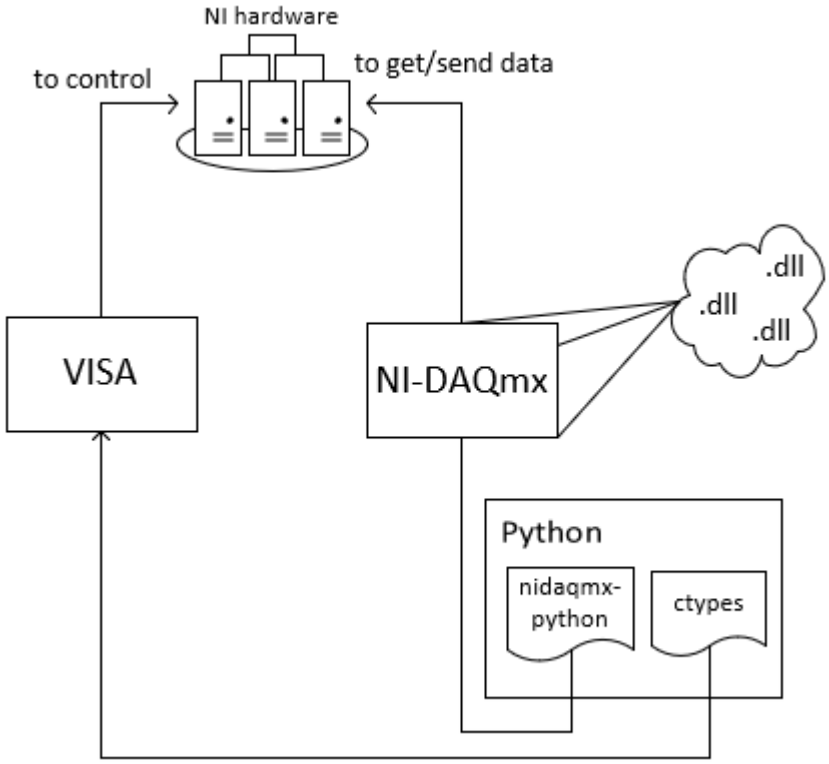


Рисунок 2.1. – Основна ідея роботи

Необхідно контролювати апаратне забезпечення National Instruments та отримувати/відсилати дані. Для цього використовується драйвер NI-DAQmx. Цей драйвер містить DLL для маніпуляції з даними (nicaiu.dll) та контролю приладів (visa32.dll) (стандарт VISA). Взаємодія з драйвером NI-DAQmx за допомогою Python здійснюється за допомоги пакету nidaqmx-python. Пакет

реалізований у вигляді об'єктно-орієнтованої обгортки навколо API NI-DAQmx, що створена з використанням мови C, за допомогою Python ctypes.

Рівняння (2.1) та (2.2) використовуються для пошуку 2-ї похідної для вилучення критичного струму.

$$I'' = I_{i+1} - 2I_i + I_{i-1} \quad (2.1)$$

$$V'' = V_{i+1} - 2V_i + V_{i-1} \quad (2.2)$$

Рівняння (2.3) використовується для вилучення нормального опору, де I_1 і V_1 - значення в початковій точці обраного діапазону даних, а I_1 і V_2 в кінцевій точці вимірюваних даних (насправді, V_2 – максимальне значення напруги).

$$R_n = (V_2 - V_1) / (I_2 - I_1) \quad (2.3)$$

Вибір архітектури ПО також є дуже важливим завданням, тому що планується постійне розширення та додавання нових функцій та модулів не тільки безпосередньо розробником, але й користувачами.

З огляду на те, що планується обмін створеними модулями між дослідниками і необхідність забезпечення обміну даними між модулями, необхідність узгодженості інтерфейсу, було розроблено архітектуру цільового ПЗ, що зображено на схемах 2.2-2.20.

Модуль отримання даних

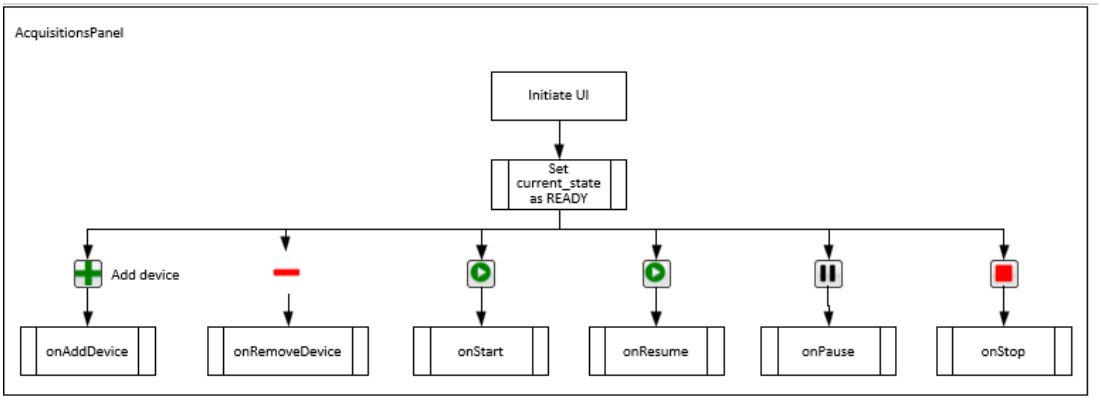


Рисунок 2.2. – Функціональні можливості модулю зчитування даних

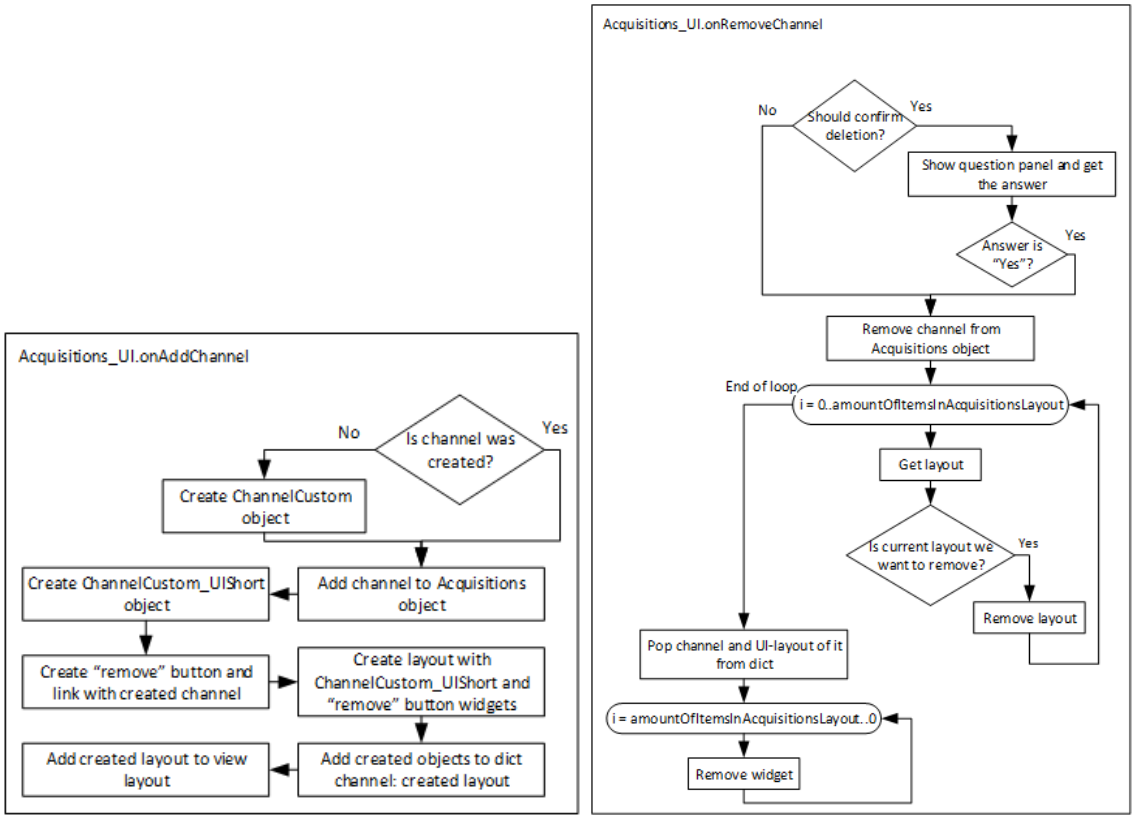


Рисунок 2.3. – Додавання та видалення каналів

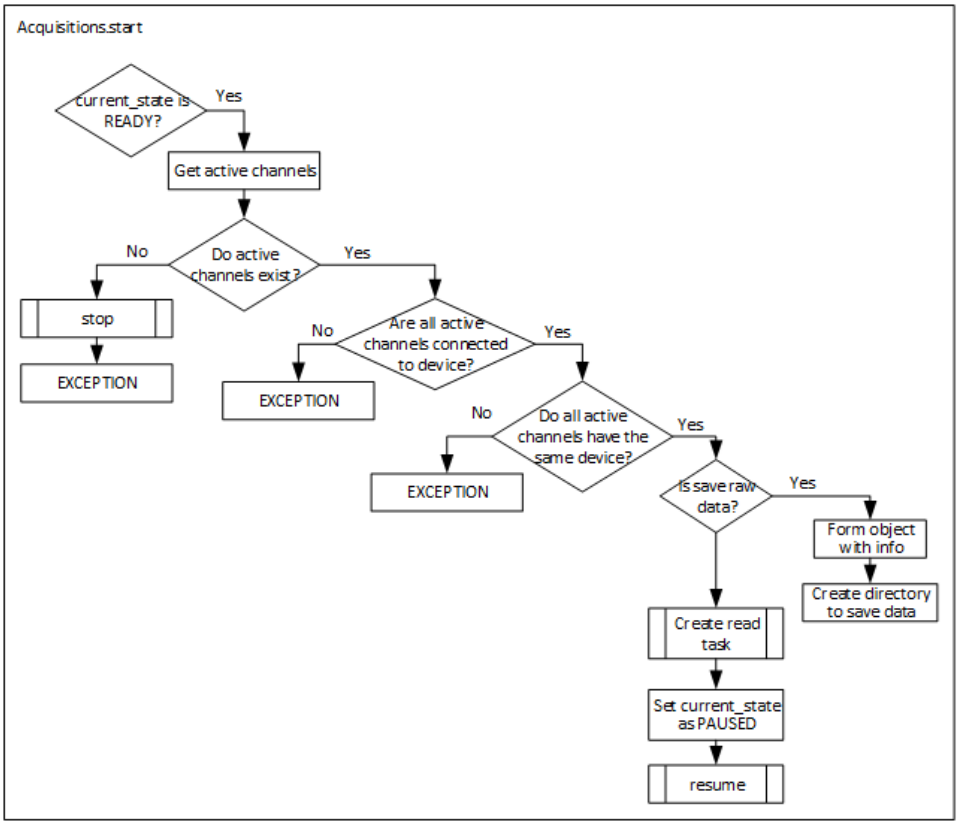


Рисунок 2.4. – Розпочати зчитування даних

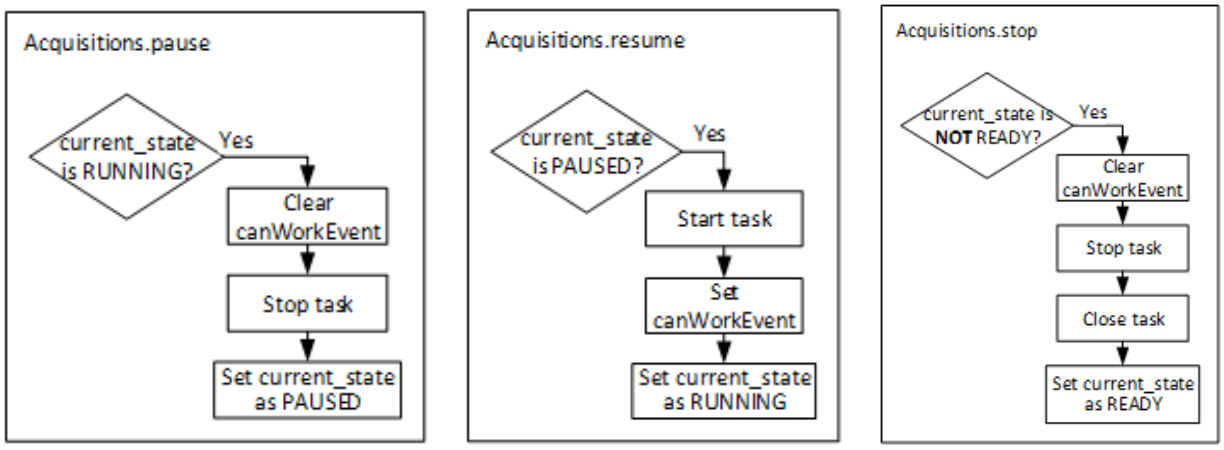


Рисунок 2.5. – Призупинити, продовжити, зупинити зчитування даних

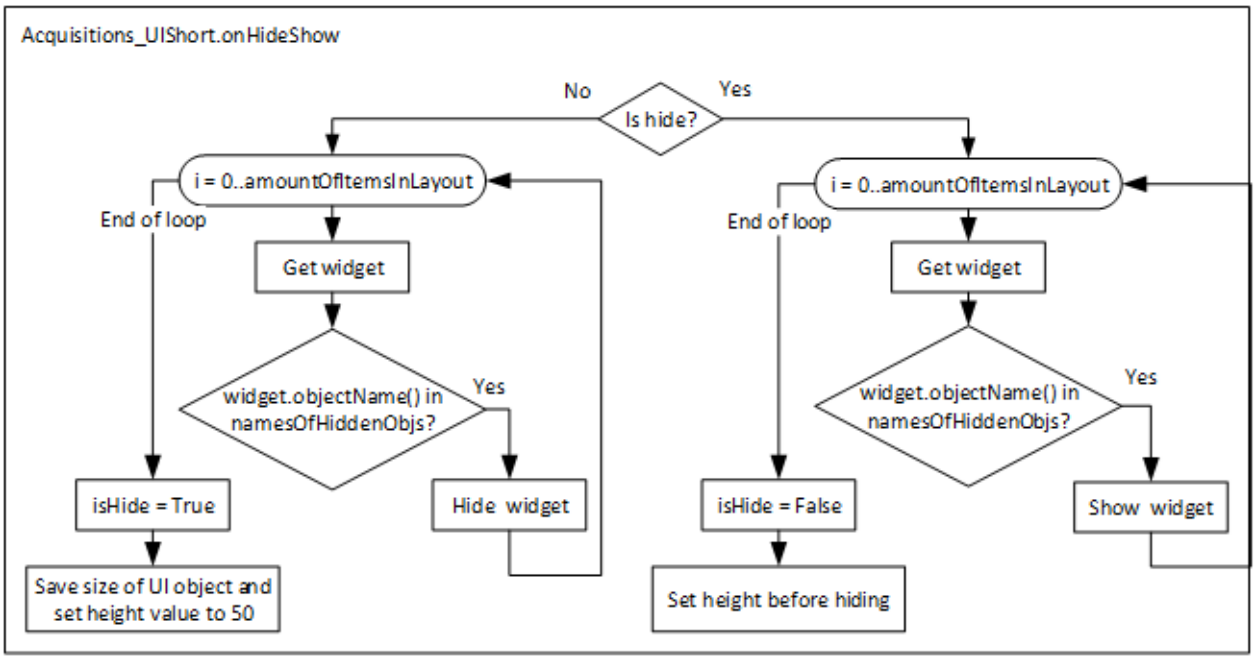


Рисунок 2.6. – Приховати/показати модуль зчитування даних

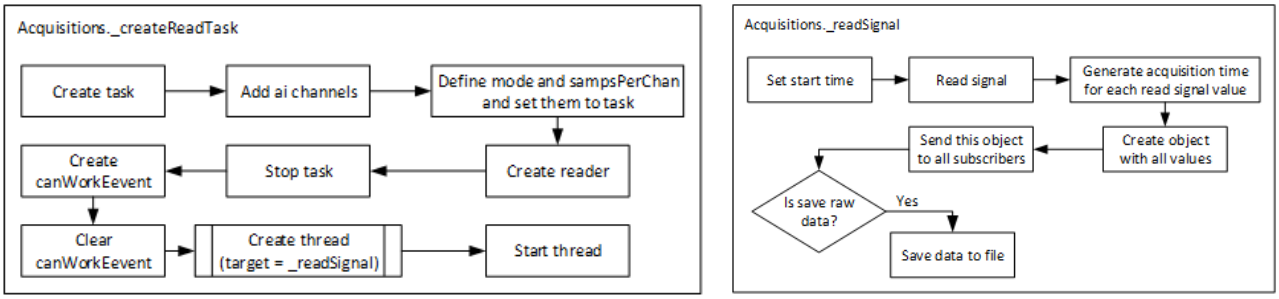


Рисунок 2.7. – Створення та робота задачі зчитування даних

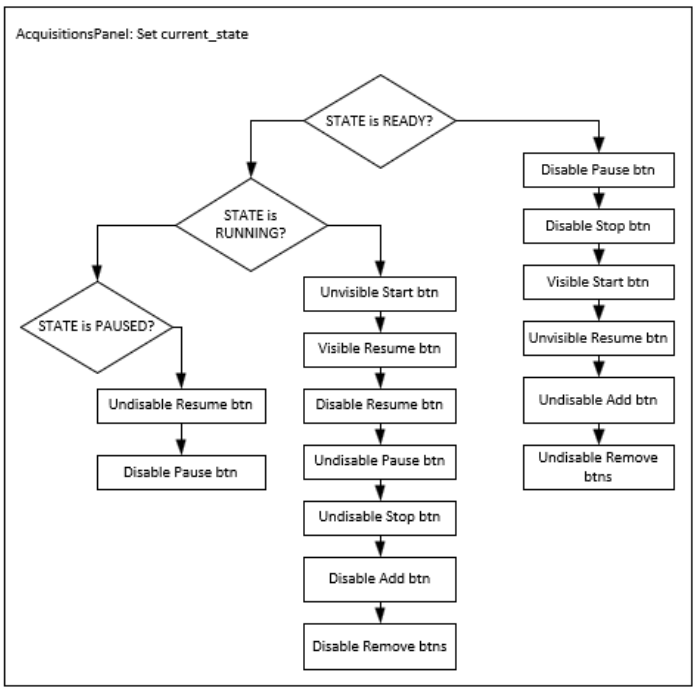


Рисунок 2.8. – Дії при зміні стану модулю зчитування даних

Налаштування та робота каналу зчитування даних

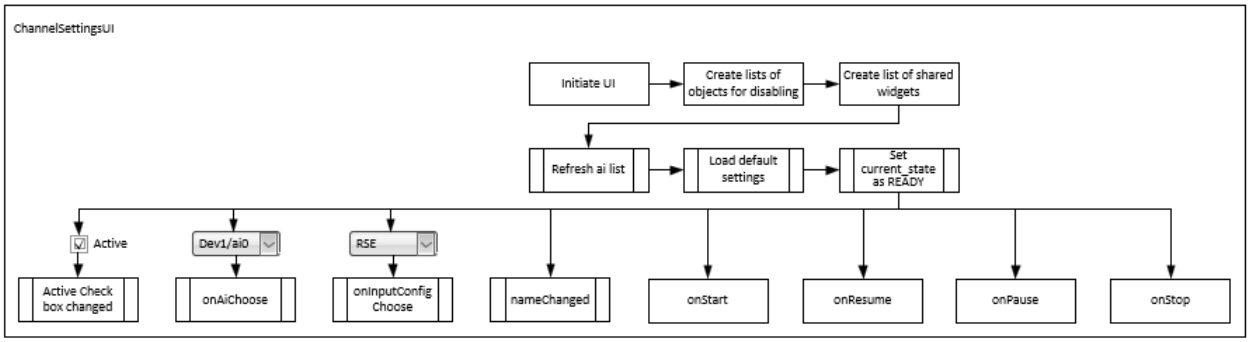
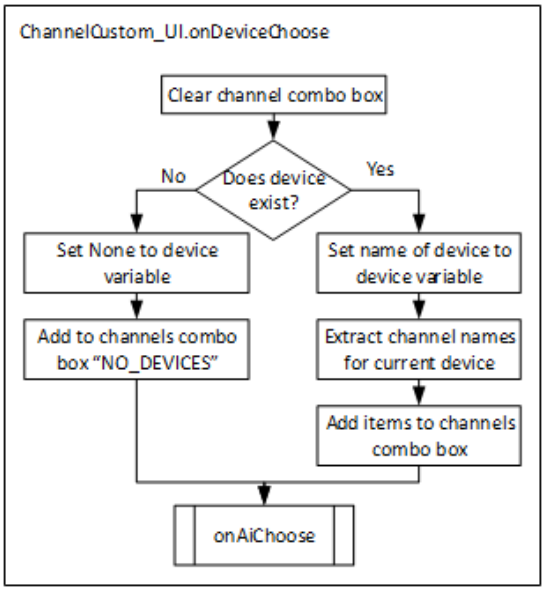
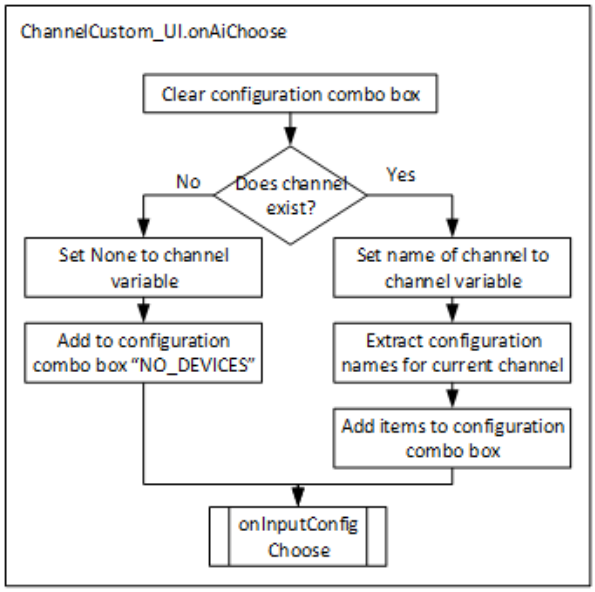


Рисунок 2.9. – Функціональні можливості налаштування каналу





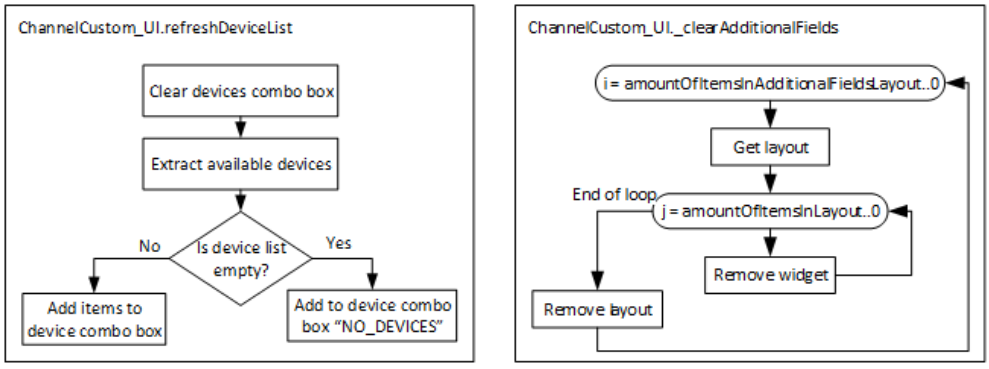


Рисунок 2.10. – Зміна пристрою та каналу для зчитування

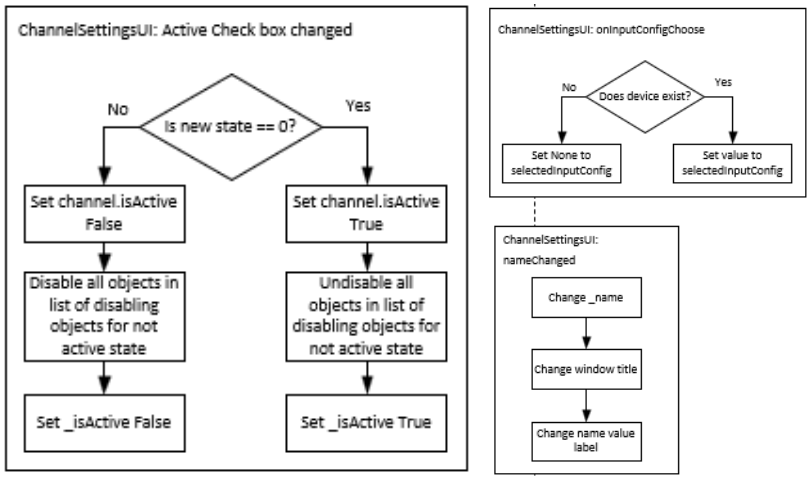


Рисунок 2.11. – Зміна режиму зчитування та активності каналу

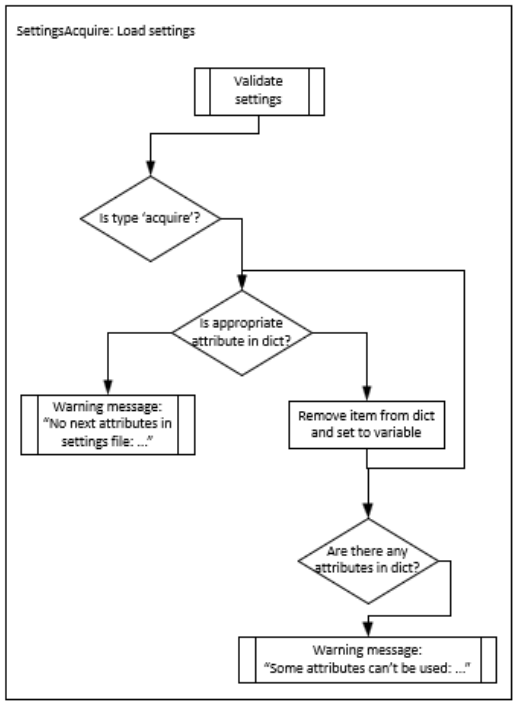


Рисунок 2.12. – Застосування зовнишнього файлу конфігурації

Модуль візуалізації даних

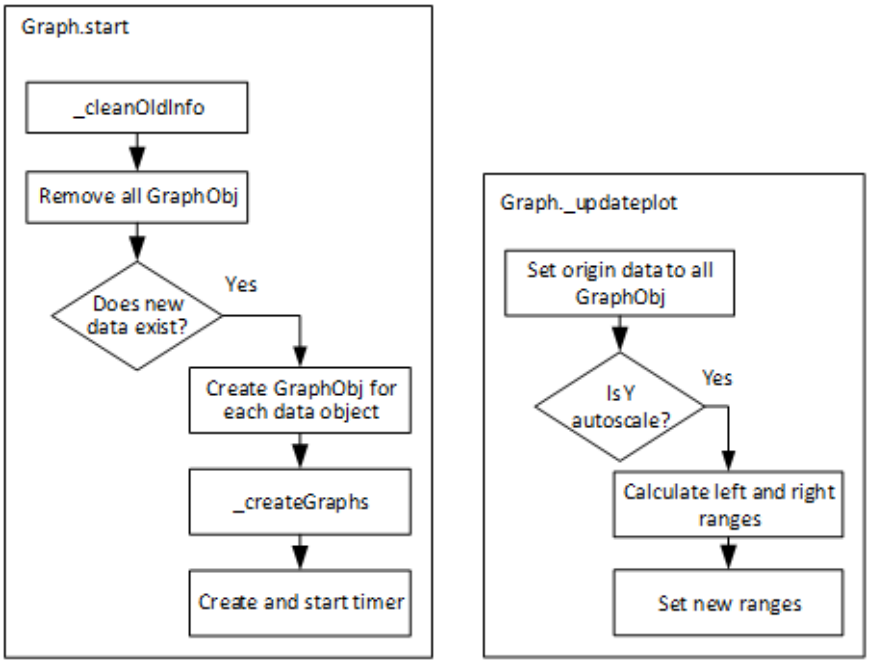


Рисунок 2.13. – Початок відображення даних та оновлення інформації

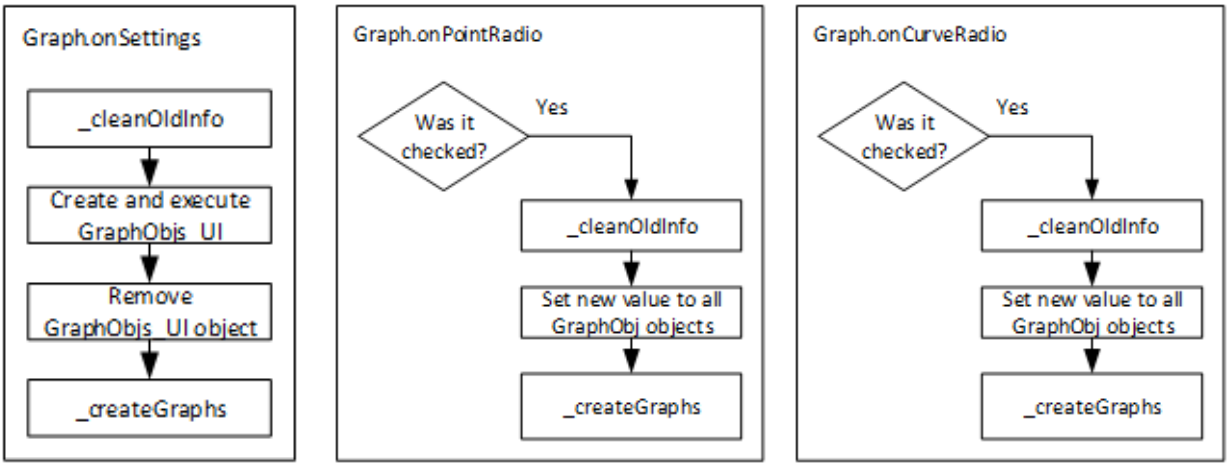


Рисунок 2.14. – Налаштування модуля

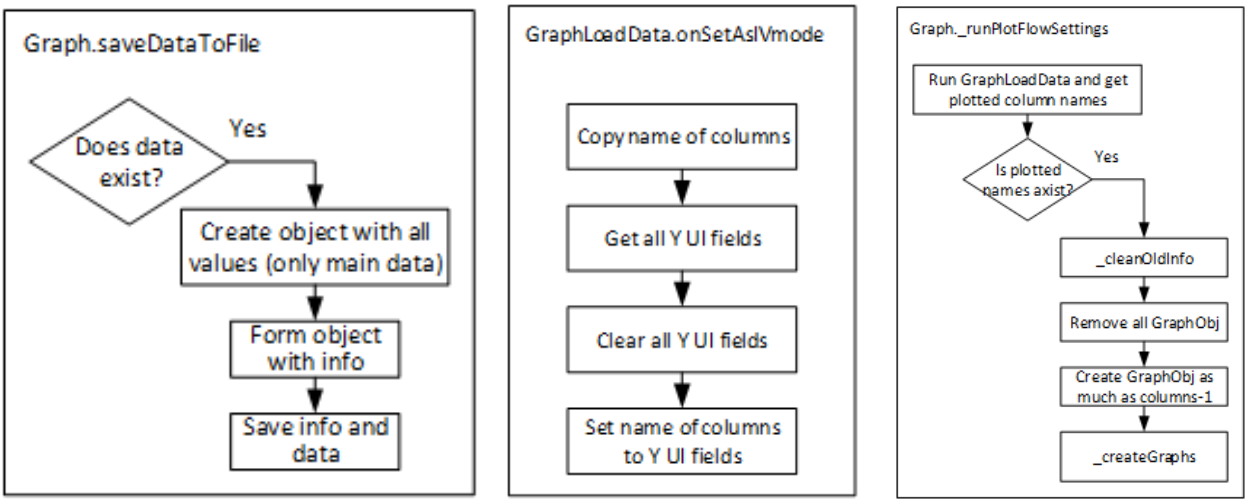


Рисунок 2.15. – Іморт та експорт даних

Модуль near-realtime процесу

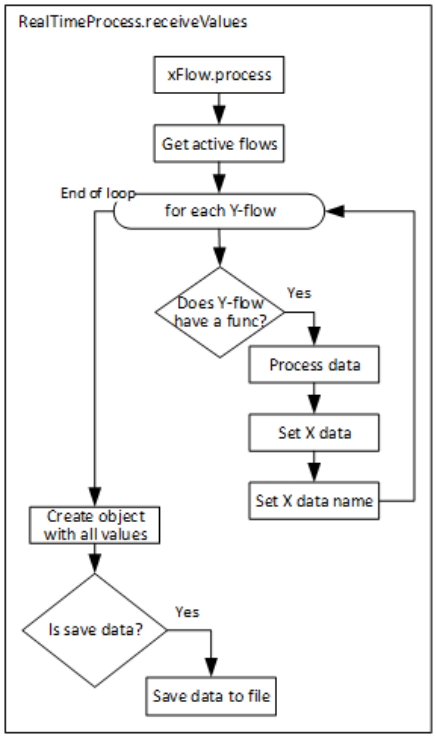


Рисунок 2.16. – Обробка користувацької функції

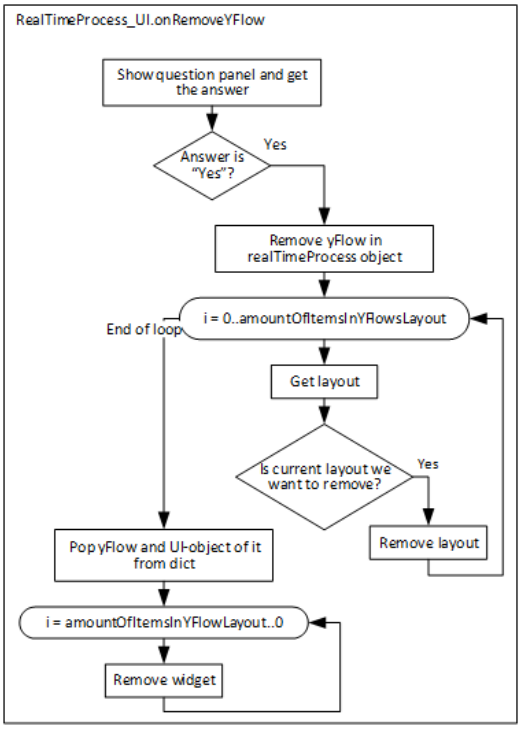


Рисунок 2.17. – Видалення користувацького поля Y

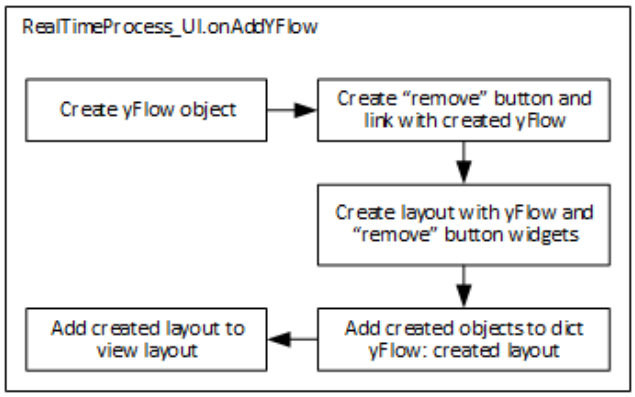


Рисунок 2.18. – Додавання користувацького поля Y

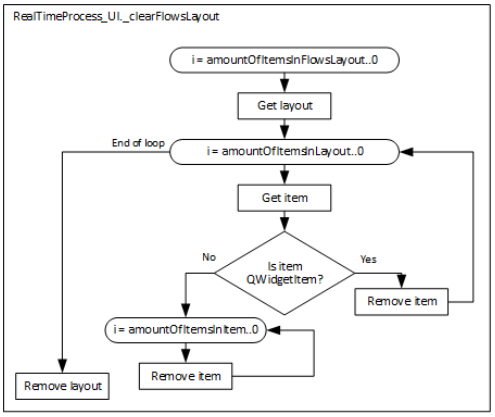


Рисунок 2.19. – Очищення даних

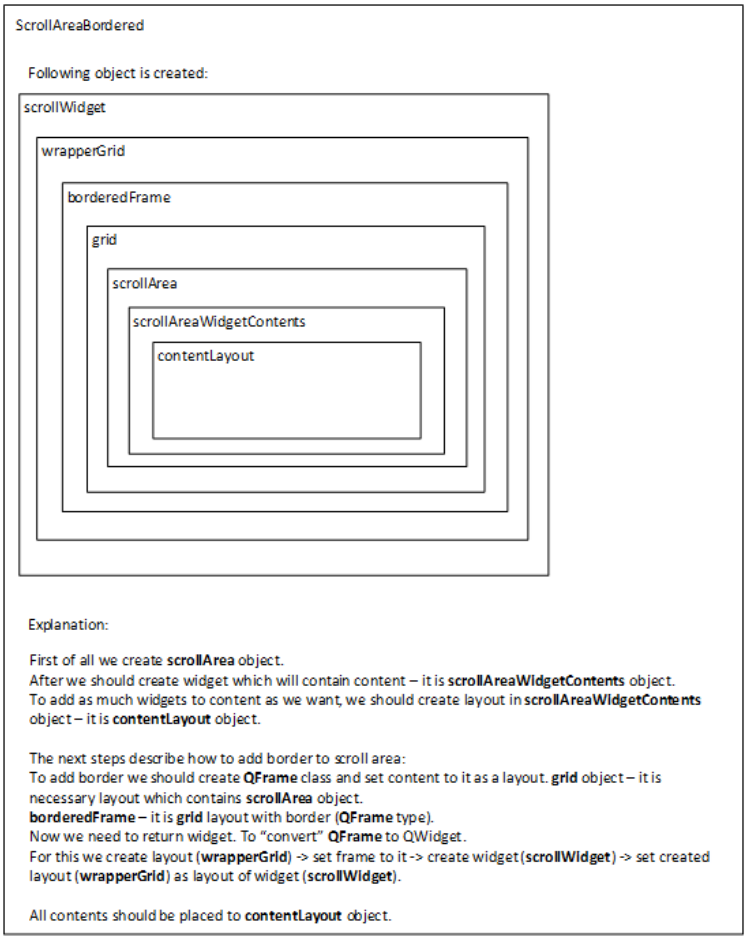


Рисунок 2.20. – Приклад організації UI-елементів

Описана вище архітектура гарантує простоту в розширенні, тобто додавання нових функцій та модулів не тільки безпосередньо розробником, але й користувачами. Угоджені зовнішні інтерфейси забезпечують обмін даними між модулями.

3. Практична реалізація

Відповідно до загальних вимог, дослідження можна розділити на наступні підзадачі:

- Отримання даних з обладнання.
- Генерація даних.
- Обробка даних On-The-Fly.
- Швидкий аналіз даних.
- Пост-обробка.

Очевидно, що перше, що нам потрібно зробити – це отримати дані з обладнання.

Для цього користувач повинен додати мінімум один канал зв’язку з обладнанням та налаштувати його.

Канали мають наступні властивості:

- **State** – характеризує поточний стан каналу:
  “State” – очікує на запуск;
  “Running” – запущений;
- **Active** (checkbox) – означає, чи буде приймати участь даний канал в отримані/посиланні даних.
- **Name** – ім’я каналу (задає користувач)
- **Channel**:
  Device – список, в якому необхідно обрати підключений пристрій NI.
  Device channel – список, в якому необхідно обрати канал пристрою, який буде використовуватися для отримання/посилання даних.
  Channel input configuration – режим отримання даних:
    “DIFFERENTIAL” – вимірює різницю потенціалів між AI+ та AI-.
    “RSE” – вимірює різницю потенціалів між AI та AI GND.
- **Max** value (double) – максимальне значення в потоку даних.
- **Min** value (double) – мінімальне значення в потоку даних.

- **Special settings** – спеціальні, специфічні значення для цього каналу. Можуть бути додані лише через файл конфігурації, в якому необхідно зазначити ім’я та значення. Користувач може задати декілька значень для спеціальної змінної і обрати необхідне з випадаючого списку.

Користувач може зберегти налаштування каналу або застосувати існуюче.

Найчастіше змінювані характеристики винесені у швидке меню. Доступ до всіх інших характеристик можна отримати з вікна «Налаштування».

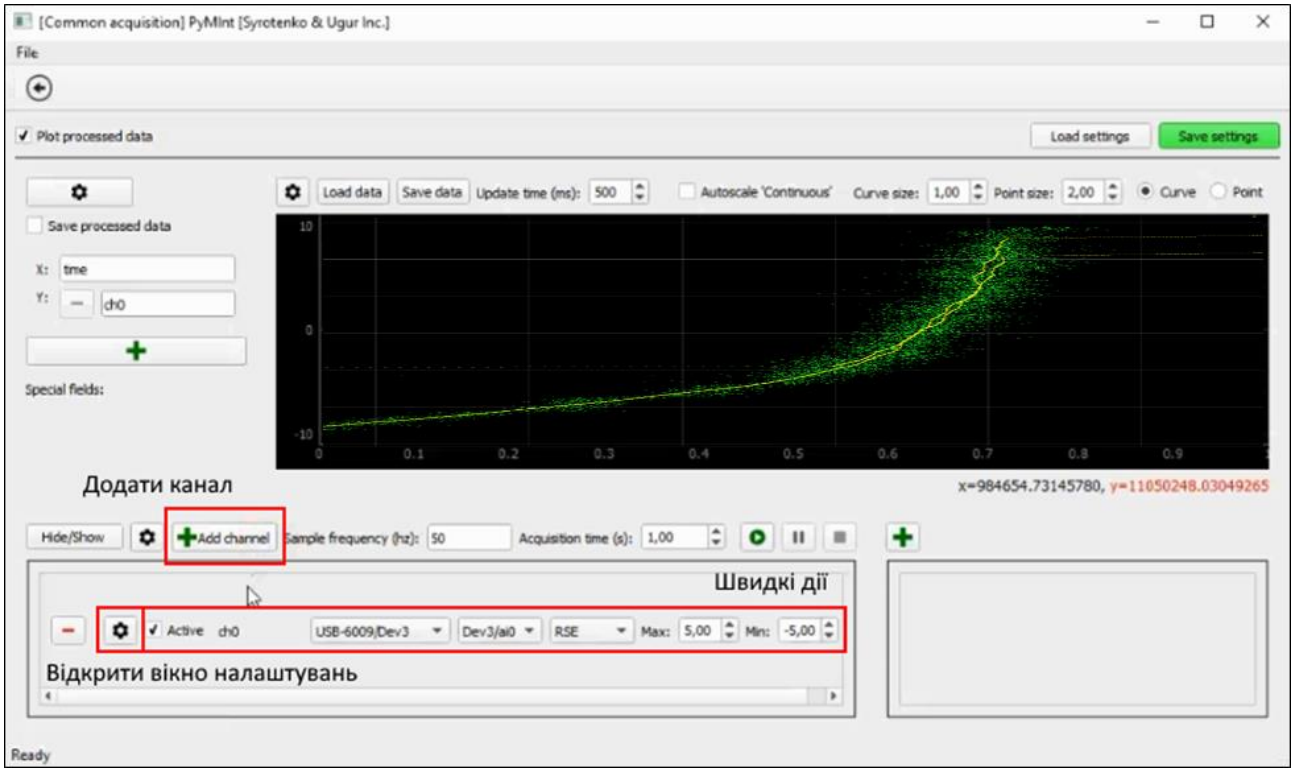


Рисунок 3.1. – Модуль зчитування даних

Рисунок 3.2. – Налаштування каналу

Після того, як користувач додав та налаштував всі необхідні канали, необхідно задати загальні налаштувати для отримання даних. Серед них:

- **Sample frequency (hz)** – частота отримання даних
- **Acquisition time (s)** – час отримання даних. Якщо задати значення «0», дані будуть отримуватися поки користувач не зупинить.
- **Save data** (checkbox) – зберігати отриманні дані у файл.
- **Start** (button) – розпочати отримання даних
- **Pause** (button) - призупинити
- **Stop** (button) – зупинити
- **Choose path** – обрати шлях для збереження даних

Користувач може додати, виділити або налаштувати вже існуючі канали. Також користувач може зберегти налаштування або застосувати існуюче. При цьому зберігаються налаштування всіх каналів, які приймають участь в отриманні даних. Користувач може приховати/показати панель отримання даних для збільшення екранного простору.

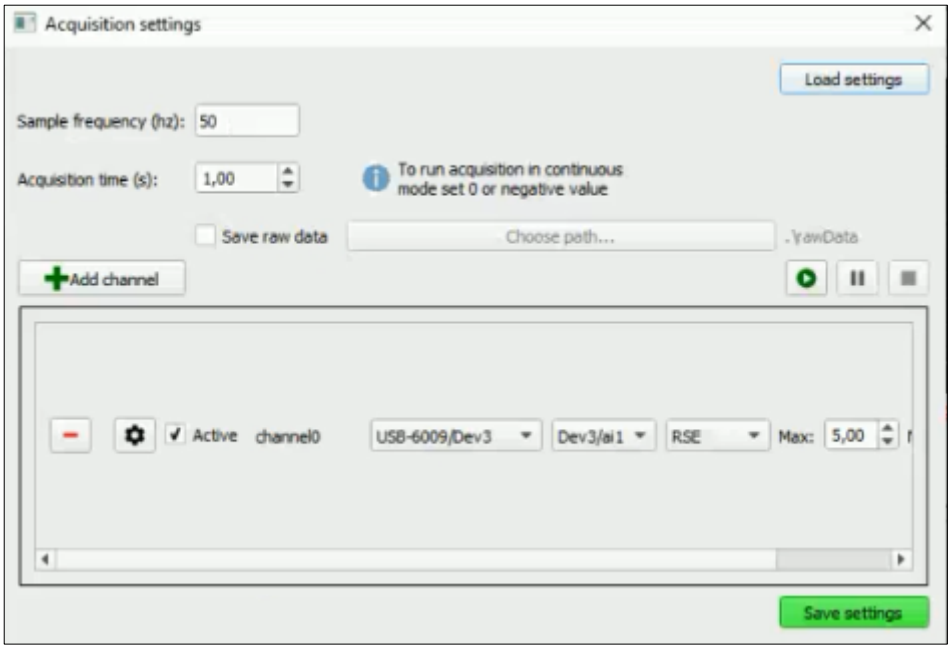


Рисунок 3.3. – Налаштування модулю зчитування даних

Під час отримання даних цільове ПО повинно мати можливість їх обробки near-realtime (в даному дослідженні використовується назва «On-The-Fly»). Це

необхідно для майже миттєвого аналізу даних, що дозволяє відмовитися від створювання нескінченної кількості додаткових модулів.

Однією з основних вимог є те, що користувач повинен сам задавати правила трансформування даних. Для цього існують два обов'язкових текстових поля X та Y для формул. Також користувач має можливість додавати необхідну йому кількість полів Y.

Правила заповнення цих полів наступне:

- користувач повинен використовувати назву каналів з панелі отримання даних та / або значення «time», яке завжди існує;
- користувач може використовувати спеціальні змінні. Наприклад, звичайний канал не має спеціальних змінних, але спеціальні підготовлені налаштування каналів для IV-вимірювань мають спеціальні змінні "Gain" та "R". Ми можемо додати інші «спеціальні змінні», які забажаємо. Щоб використовувати спеціальні змінні, користувач повинен задати ім'я каналу + «.» + ім'я змінної. Наприклад: «I.R6».

Рисунок 3.4. – Модуль near-realtime обробки даних

Це є потужним механізмом, який дозволяє забути про те, що користувач працює лише з масивами даних та числами. Він дозволяє абстрагуватися та наблизитися до фізичного значення та предметної області. Користувач може зберегти налаштування або застосувати існуюче.

Модуль, який відповідає за побудову графіку, має наступні властивості:

- **Update time (ms)** – час, через який графік оновлюється (відображає нові дані, якщо такі є)
- **Autoscale ‘Continuous’** – показує дані лише за останні 20 секунд (є корисним при великому обсязі даних)
- **Curve size** – товщина кривої
- **Point size** – товщина точок
- **Curve/Point** – відображати дані як криву або як точки
- **Color** – колір відображення даних
- **Average size** – ступінь застосування Moving Average (кількість точок)
- **Hide** (checkbox) – дозволяє приховати данні

Окрім цього, даний модуль дозволяє виконувати додаткові операції швидкого аналізу даних:

- змінна середня (moving average);
- верифікація інверсійної симетрії кривих;
- імпорт та збереження даних.

При імпорті даних, користувач може обрати як і які данні необхідно відображати. Також користувач може змінити залежність вісей (відображати не Y та Z від X, а X та Z від Y).

Перш ніж перейти до останньої та головної частини роботи, а саме пост-обробку та симуляцію даних, звернемо увагу на відправку даних до пристрою. Для генерації даних, користувач повинен додати канал на панелі генерації даних. Також користувач може відкрити канал генерації даних окремим вікном (File -> Generation). Канал генерації даних має наступні властивості:

- **State** – характеризує поточний стан каналу:

  “State” – очікує на запуск;

  “Running” – запущений;
- **Name** – користувацьке ім’я каналу
- **Choose device**

  Device – список, в якому необхідно обрати підключений пристрій NI.

Channel – список, в якому необхідно обрати канал пристрою, який буде використовуватися для отримання/посилання даних.

- **Signal** – обрати існуючий тип сигналу
- **Max** – максимальне значення даних
- **Min** – мінімальне значення даних
- **Frequency –** частота
- **Sample freq** – кількість значень в одному періоді
- **Ampl** – амплітуда
- **Data file** – обрати файл з даними для генерації
- **Start** (button) – розпочати генерацію даних
- **Stop** (button) - зупинити

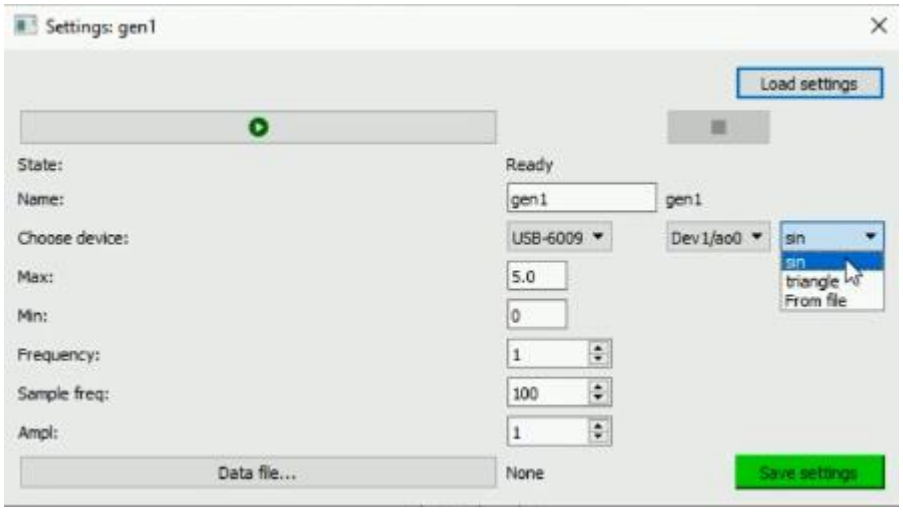


Рисунок 3.5. – Налаштування каналу для генерації даних

Користувач може зберегти налаштування або застосувати існуюче.

Також користувач може відключити побудову графіку під час отримання даних – це дозволяє значно зменшити обсяги необхідної оперативної пам’яті при великих обсягах даних.

Пост-обробка дозволяє знайти критичний струм і нормальний опір. Це також дозволяє моделювати IV-криву за допомогою симуляційної системи JSim. Користувач може вибрати файл Netlist та налаштувати додаткові параметри симуляції використовуючи створений графічний інтерфейс.

Для моделювання IV-curve перш за все необхідно визначити деякі параметри. Розроблене програмне забезпечення автоматично витягує ці параметри з експериментальних даних. Це необхідно для автоматизації процесу побудови моделі на основі отриманих даних. Якщо головне вікно прогами містить зчитані дані, то вони автоматично експортуються у вікно симуляції. Якщо таких даних не має або користувач бажає використовувати інші дані, можна скористатися функцією імпорту даних.

Параметри, що витягуються автоматично:

- критичний струм (**I_c**)
- нормальний опір (**R_n**)
- розрядна напруга (**V_g**)

Для знаходження критичного струму та нормального опору необхідно локалізувати точку перегину. Для підвищення ефективності розрахунків, було вирішено обирати конкретні області вимірюваних даних. Для вилучення параметрів вибираються дані зі збільшенням струму (значення по осі Y більше за нуль).

Крім того, користувач може налаштувати інші параметри побудови моделі:

- **I_{max}** – максимальне значення струму
- **I_{min}** – мінімальне значення струму
- **Temperature** – температура, при якій виконується експеримент
- **DataRunnAvgWindow** – ступінь застосування Moving Average (кількість точок) для експериментальних даних

- **Thermal noise** (checkbox) – врахувати тепловий шум
- **tD** – параметр моделювання (кількість точок в періоді)
- **SimRunnAvgWindow** – ступінь застосування Moving Average (кількість точок) для результату моделювання
- **Path to Netlist** – обрати шлях до файлу, що описує модель для JSim

Зауважимо, що для успішної симуляції необхідно ще знати $\mathbf{R_{n\text{-}negative}}$ та $\mathbf{V_{g\text{-}negative}}$. Ці параметри обраховуються з від'ємної частини IV-curve.

Модуль пост-обробки також дозволяє обчислити середньоквадратичну помилку (Root Mean Square Error), що дає представлення того, наскільки близька імітація IV-кривої до реальних вимірювань.

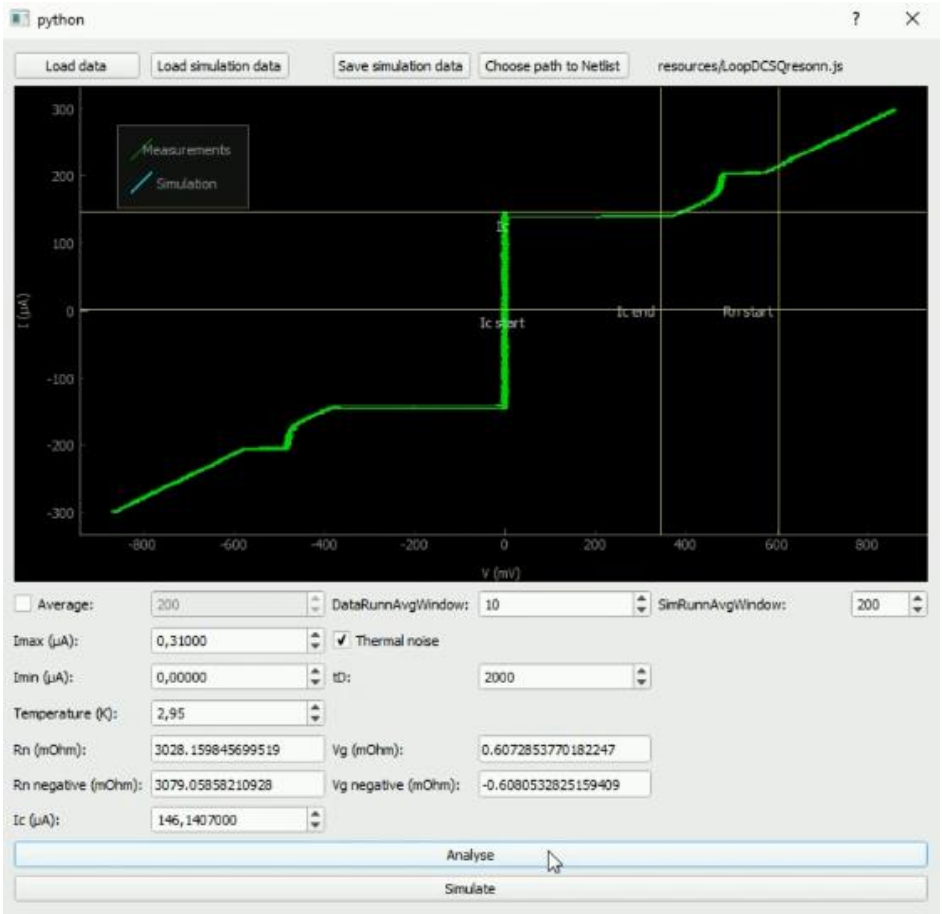


Рисунок 3.6. – Витягування значень з експериментальних даних

Рисунок 3.7. – Результат роботи JSim

Висновки

В результаті дослідження було створено гнучку систему враховуючи умови модульності. Результати роботи дозволяють зчитувати дані з криогенного обладнання за допомоги обладнання National Instruments, обробляти їх в режимі near-realtime та робити більш глибокий аналіз. Також розроблене програмне забезпечення дозволяє витягувати з експериментальних даних необхідну інформацію для симуляції за допомоги JSim.

Створене програмне забезпечення легко змінювати та інтегрувати власні створені модулі. Розроблена архітектура дозволяє скомпонувати власну, унікальну програму, яка містить тільки необхідний функціонал та не споживає зайвих ресурсів. Користувачі мають можливість обмінюватися створеними модулями завдяки узгодженому інтерфейсу, що сприяє прискоренню наукових досліджень в області вивчення поведінки надпровідних пристроїв.

Також в результаті було отримано опис та схеми архітектурних рішень. Обробка цих даних дозволить скласти додаткові обмеження і допущення, які необхідно враховувати при подальших дослідженнях.

Результати дослідження дозволять мінімізувати наступні показники:

- Несподівану поведінку IV-Curve
- Невідповідність симульованих даних реальним вимірюванням
- Кількість часу на розробку та впровадження модулів різної складності.
- Кількість часу на підтримку та зміну вже створених модулів:
    - Власних
    - Інших дослідників
- Кількість необхідних комп’ютерних ресурсів:
    - Процесор
    - Оперативна пам’ять

Серед недоліків розробленого програмного забезпечення можна виділити не широкий інтерфейс обміну даних між модулями. Тобто, необхідно розширити можливості співпраці різних модулів.

Подальші дослідження полягають в тому, щоб поліпшити інтерфейс користувача, додати можливість власного налаштування графічних елементів (їх переміщення та масштабування).

Також необхідно поліпшити функціонал витягування необхідних даних для симуляції – убрати необхідність явної вказівки на області перегину, інтегрувати редагування Netlist файлів в програмне забезпечення та збільшити точність використання Root Mean Square Error (вимір відмінностей між значеннями, що передбачені моделлю, і фактичними значеннями).

Результати дослідження були впроваджені в лабораторію дослідження надпровідних ланцюгів університету, на базі якого воно розроблялося (Université Savoie Mont Blanc). Розроблене програмне забезпечення активно використовується для дослідів і ведеться активна розробка з ціллю поліпшення вже існуючого функціоналу та впровадження нового.

Автор приймав участь у наступних конференціях:

Виступ на конференції «ColdFlux» на тему «Software development for post-processing of measured data from superconducting circuits», Pascal Febvre , Serhii Syrotenko, Ugur Yilmaz and Sasan Razmkhah, 2019.

На дану тематику автором було написано тези:

Публікація тез на конференції «Nanomaterials: Applications & Properties» на тему «Experimental study and time-domain analysis of microwave resonances of a transformer-coupled dc SQUID at different temperatures», Pascal Febvre , Serhii Syrotenko, Ugur Yilmaz and Sasan Razmkhah, 2019.

Література

1. A. Chwala, "Superconductor Science&Technology" / A. Chwala, R. Stolz, J. Ramos, V. Schultze, H-G. Meyer, D. Kretzschmar, vol. 12, no. 11, – 1999. – 1036p.
2. Courtland R. "Superconductor Logic Goes Low-Power". IEEE spectrum, – 2011.
3. V. Foglietti, "Transactions on Magnetics". IEEE vol. 27, no. 2, – March 1991. – 2959-2962pp.
4. S. M. Anton, "Magnetic Flux Noise in dc SQUIDs: Temperature and Geometry Dependence" / S. M. Anton, J. S. Birenbaum, S. R. O'Kelley, V. Bolkhovsky, D. A. Braje, G. Fitch, M. Neeley, G. C. Hilton, H.-M. Cho, K. D. Irwin, F. C. Wellstood, W. D. Oliver, A. Shnirman, John Clarke1. Physical Review Letters – 2013.
5. J. Knuutila, "Effects on DC SQUID characteristics of damping of input coil resonances" / J. Knuutila, A. Ahonen, C. Tesche, J. Low Temp. Phys., vol. 68, no. 3–4, – Aug. 1987. – 269–284pp.
6. National Instruments, "Python Resources for NI Hardware and Software," 2019. [Online]. Available: http://www.ni.com/white-paper/53059/en/
7. Sandro Tosi, "Matplotlib for Python Developers", 2009. – 308p.
8. Carson Sievert, "Interactive web-based data visualization with R, plotly, and shiny", 2019. – 208p.
9. Chad Adams, "Learning Python Data Visualization", 2014. – 200p.
10. NSR Physiome Project, "JSim Modeling System Documentation", [Online]. Available: https://www.physiome.org/
11. National Instruments, "Advanced Scripting in Perl, Python and Tcl," 2015. [Online]. Available: http://www.ni.com/white-paper/8911/en/
12. National Instruments, "NI-DAQmx Python Documentation," 2017. [Online]. Available: https://nidaqmx-python.readthedocs.io/en/latest/
13. Python Software Foundation, "ctypes — A foreign function library for Python", 2019. [Online]. Available: https://docs.python.org/3/library/ctypes.html
14. Дронов В.А., "Python 3 и PyQt 5. Разработка приложений" / Дронов В.А., Прохоренок Н.А., – 2018. – 832стр.

Додаток А

Acquisitions.py:

```
import ast
import atexit
import copy
import os
import re


from PyQt5.QtCore import QObjectCleanupHandler, Qt
from PyQt5.QtGui import QIcon, QPixmap, QDoubleValidator
from PyQt5.QtWidgets import QWidget, QPushButton, QGridLayout, QDialog, QLabel, QLineEdit,
QHBoxLayout, QMessageBox, \
    QCheckBox, QFileDialog, QDoubleSpinBox, QSpinBox


from Acquisitions import Acquisitions
from ChannelCustom import ChannelCustom
from ChannelCustom_UIShort import ChannelCustom_UIShort
from ScrollAreaBordered import ScrollAreaBordered
from library.constants import STATES
from library.lib import saveSettingsToFile, getSettingsFromFile, checkAttrPresence,
getNumberOfChannel
from library.libUI import normalizeNumberEdit, checkRemainingSettings


__author__ = "Syrotenko Serhii <syrotenko.sergey.g@gmail.com>, Yilmaz Ugur"



class Acquisitions_UI(QDialog):
    """
    UI representation of `PyNiDAQ.Acquisitions` class. \n
    This class allows to change acquiring configuration through UI.
    """


    def __init__(self, acquisitions=None, settings=None, pathToSettingsFile=None):
        """
        Parameters
        ----------
        acquisitions: Acquisitions.py
            `PyNiDAQ.Acquisitions` object
        settings: dict
            See README_dev.md for object structure
        pathToSettingsFile: str
```

```
            See README_dev.md for file structure
    @currentState.setter
    def currentState(self, value):
        self._currentState = value


        super().__init__()


        if not acquisitions:
            self.acquisitions = Acquisitions()
        else:
            self.acquisitions = acquisitions


        self.getSettings = self.acquisitions.getSettings
        self.acquisitions.subscribeToWarning(self.notifiedWarning)


        self.subsWarning = list()
        self.acquisitionsDict = dict()


        self._initMainWIndow()
        self._initUI()


        self.objectsToDisableMain = [self.addChannelBtn, self.freqEdit, self.acqTimeEdit,
self.isSaveRawDataCheckBox,
                                     self.rawDataDirPathBtn, self.rawDataDirPathValue,
self.isSaveInSeparateFilesCheckBox,
                                     self.maximumSizeOfFileLabel, self.maximumSizeOfFileEdit,
                                     self.timePrecisionLabel, self.timePrecisionEdit,
                                     self.otherValPrecisionLabel, self.otherValPrecisionEdit]
        if not settings:
            if not pathToSettingsFile:

self.loadSettings(getSettingsFromFile('resources/settings/defaultSettings_Acquisition.ini'))
            else:
                self.loadSettings(getSettingsFromFile(pathToSettingsFile))
        else:
            self.loadSettings(settings)
        self.currentState = STATES.READY


        self.destroyed.connect(lambda: self.onDestroy())
        atexit.register(self.onDestroy)


    def onDestroy(self):
        print('Acquisitions_UI was destroyed')
        atexit.unregister(self.onDestroy)
```

```
        self.acquisitions.stop()
        del self.acquisitions

    def showEvent(self, event):
        self._placeWidgets()

    @property
    def channels(self):
        return self.acquisitions.channels

    @channels.setter
    def channels(self, values):
        self.acquisitions.channels = values

    @property
    def currentState(self):
        return self.acquisitions.currentState

    @currentState.setter
    def currentState(self, value):
        self.acquisitions.currentState = value
        self._changeAppState(self.acquisitions.currentState)

    def _changeAppState(self, state):
        if state == STATES.READY:
            self.pauseBtn.setDisabled(True)
            self.stopBtn.setDisabled(True)

            self.startBtn.setVisible(True)
            self.resumeBtn.setVisible(False)

            for object_ in self.objectsToDisableMain:
                object_.setDisabled(False)

            for acquireObject in self.acquisitionsDict.values():
                acquireObject.itemAt(0).widget().setDisabled(False)

            # WARNING: method 'onIsSaveRawData' is not called if use 'setChecked'
            self.onIsSaveRawData(self.acquisitions.isSaveRawData)

        elif state == STATES.RUNNING:
            self.startBtn.setVisible(False)
```

```
            self.resumeBtn.setVisible(True)
            self.resumeBtn.setDisabled(True)
            self.pauseBtn.setDisabled(False)
            self.stopBtn.setDisabled(False)

            for object_ in self.objectsToDisableMain:
                object_.setDisabled(True)

            for acquireObject in self.acquisitionsDict.values():
                acquireObject.itemAt(0).widget().setDisabled(True)

        elif state == STATES.PAUSED:
            self.resumeBtn.setDisabled(False)
            self.pauseBtn.setDisabled(True)

    def _initMainWIndow(self):
        self.setWindowTitle('Acquisition settings')
        self.setWindowFlag(Qt.WindowContextHelpButtonHint, False)
        self.setWindowModality(Qt.ApplicationModal)
        self.resize(730, 400)

    def _initUI(self):
        self.addChannelBtn = QPushButton('Add channel')
        self.addChannelBtn.setObjectName('addChannelBtn')
        self.addChannelBtn.setIcon(QIcon(QPixmap('resources/images/iconPlus.png')))
        self.addChannelBtn.clicked.connect(self.onAddChannel)

        self.startBtn = QPushButton()
        self.startBtn.setObjectName('start')
        self.startBtn.setIcon(QIcon(QPixmap('resources/images/iconStart.png')))

        self.resumeBtn = QPushButton()
        self.resumeBtn.setObjectName('resume')
        self.resumeBtn.setIcon(QIcon(QPixmap('resources/images/iconStart.png')))
        self.resumeBtn.clicked.connect(self.onResume)

        self.pauseBtn = QPushButton()
        self.pauseBtn.setObjectName('pause')
        self.pauseBtn.setIcon(QIcon(QPixmap('resources/images/iconPause.png')))
        self.pauseBtn.clicked.connect(self.onPause)

        self.stopBtn = QPushButton()
```

```
        self.stopBtn.setObjectName('stop')
        self.stopBtn.setIcon(QIcon(QPixmap('resources/images/iconStop.png')))
        self.stopBtn.clicked.connect(self.onStop)

        self.acquisitionsView = self._initAcquisitionsView()
        self.acquisitionsView.setObjectName('acquisitionsView')
        self.acquisitionsView.setContentsMargins(-8, -8, -8, -8)

        self.freqLabel = QLabel('Sample frequency (Hz):')
        self.freqEdit = QLineEdit(str(self.acquisitions.freq))
        self.freqEdit.setValidator(QDoubleValidator())
        self.freqEdit.setFixedWidth(70)
        self.freqEdit.textChanged.connect(self.onFreqChange)

        self.acqTimeLabel = QLabel('Acquisition time (s):')
        self.acqTimeEdit = QDoubleSpinBox()
        self.acqTimeEdit.setValue(self.acquisitions.acqTime)
        self.acqTimeEdit.setSingleStep(0.1)
        self.acqTimeEdit.setMaximum(100000)
        self.acqTimeEdit.setFixedWidth(70)
        self.acqTimeEdit.valueChanged.connect(self.onAcqTimeChange)

        infoContinuousModeIcon = QLabel()

infoContinuousModeIcon.setPixmap(QPixmap('resources/images/iconInfo.png').scaledToWidth(22))
        infoContinuousModeIcon.setFixedWidth(20)
        infoContinuousModeLabel = QLabel('To run acquisition in continuous\nmode set 0 or
negative value')
        infoContinuousModeLayout = QHBoxLayout()
        infoContinuousModeLayout.addWidget(infoContinuousModeIcon)
        infoContinuousModeLayout.addWidget(infoContinuousModeLabel)
        self.infoContinuousMode = QWidget()
        self.infoContinuousMode.setLayout(infoContinuousModeLayout)

        self.maximumSizeOfFileLabel = QLabel('Maximum size of file (Mb):')
        self.maximumSizeOfFileEdit = QSpinBox()
        self.maximumSizeOfFileEdit.setValue(self.acquisitions.maximumSizeOfFile / 1e6)
        self.maximumSizeOfFileEdit.valueChanged.connect(self.onMaximumSizeOfFileChange)

        self.isSaveInSeparateFilesCheckBox = QCheckBox('Save in separate files')
        self.isSaveInSeparateFilesCheckBox.stateChanged.connect(self.onIsSaveInSeparateFiles)
        self.isSaveInSeparateFilesCheckBox.setChecked(self.acquisitions.isSaveInSeparateFiles)
        # WARNING: method 'onIsSaveInSeparateFiles' is not called if use 'setChecked'
```

```
        self.onIsSaveInSeparateFiles(self.acquisitions.isSaveInSeparateFiles)

        self.timePrecisionLabel = QLabel('Time precision:')
        self.timePrecisionEdit = QSpinBox()
        self.timePrecisionEdit.setFixedWidth(70)
        self.timePrecisionEdit.valueChanged.connect(self.onTimePrecision)
        self.timePrecisionEdit.setValue(self.acquisitions.timePrecision)

        self.otherValPrecisionLabel = QLabel('Other values precision:')
        self.otherValPrecisionEdit = QSpinBox()
        self.otherValPrecisionEdit.setFixedWidth(70)
        self.otherValPrecisionEdit.valueChanged.connect(self.onOtherValPrecision)
        self.otherValPrecisionEdit.setValue(self.acquisitions.otherValPrecision)

        self.rawDataDirPathBtn = QPushButton('Choose path...')
        self.rawDataDirPathBtn.clicked.connect(self.onRawDataDirPath)
        self.rawDataDirPathValue = QLabel(self.acquisitions.dataDirPath)

        self.isSaveRawDataCheckBox = QCheckBox('Save raw data')
        self.isSaveRawDataCheckBox.stateChanged.connect(self.onIsSaveRawData)
        self.isSaveRawDataCheckBox.setChecked(self.acquisitions.isSaveRawData)
        # WARNING: method 'onIsSaveRawData' is not called if use 'setChecked'
        self.onIsSaveRawData(self.acquisitions.isSaveRawData)

        self.loadSettingsFileBtn = QPushButton('Load settings')
        self.loadSettingsFileBtn.setFixedWidth(100)
        self.loadSettingsFileBtn.clicked.connect(self.onLoadSettingsFile)

        self.saveSettingsBtn = QPushButton('Save settings')
        self.saveSettingsBtn.setFixedWidth(100)
        self.saveSettingsBtn.clicked.connect(self.onSaveSettingsToFile)
        self.saveSettingsBtn.setStyleSheet("background-color: rgb(0, 240, 0);")

    def _placeWidgets(self):
        QObjectCleanupHandler().add(self.layout())
        grid = QGridLayout(self)

        grid.addWidget(self.loadSettingsFileBtn, 0, 0, 1, 0, Qt.AlignRight)

        grid.addWidget(self.freqLabel, 1, 0)
        grid.addWidget(self.freqEdit, 1, 1)
```

```
        grid.addWidget(self.acqTimeLabel, 2, 0)
        grid.addWidget(self.acqTimeEdit, 2, 1)
        grid.addWidget(self.infoContinuousMode, 2, 2, 1, 4)

        grid.addWidget(self.isSaveRawDataCheckBox, 3, 1)
        grid.addWidget(self.rawDataDirPathBtn, 3, 2)
        grid.addWidget(self.rawDataDirPathValue, 3, 3, 1, 3)
        grid.addWidget(self.isSaveInSeparateFilesCheckBox, 4, 1)
        grid.addWidget(self.maximumSizeOfFileLabel, 4, 2)
        grid.addWidget(self.maximumSizeOfFileEdit, 4, 3)
        grid.addWidget(self.isSaveInSeparateFilesCheckBox, 4, 1)
        grid.addWidget(self.timePrecisionLabel, 5, 1)
        grid.addWidget(self.timePrecisionEdit, 5, 2)
        grid.addWidget(self.otherValPrecisionLabel, 6, 1)
        grid.addWidget(self.otherValPrecisionEdit, 6, 2)

        grid.addWidget(self.addChannelBtn, 7, 0)
        grid.setColumnStretch(2, 1)
        grid.addWidget(self.startBtn, 7, 3)
        grid.addWidget(self.resumeBtn, 7, 3)
        grid.addWidget(self.pauseBtn, 7, 4)
        grid.addWidget(self.stopBtn, 7, 5)
        grid.addWidget(self.acquisitionsView, 8, 0, 1, 0)

        grid.addWidget(self.saveSettingsBtn, 9, 0, 1, 0, Qt.AlignRight)

        self.setLayout(grid)

    def _initAcquisitionsView(self):
        """Create scroll area for channels"""

        scrollAreaBordered = ScrollAreaBordered()
        self.acquisitionsLayout = scrollAreaBordered.getContentLayout()
        self.acquisitionsLayout.setSpacing(0)
        return scrollAreaBordered.getWidget()

    def onAddChannel(self, channel=None, settings=None, pathToSettingsFile=None):
        """
        Scheme:

        .. image:: docImages/Acquisitions_UI.onAddChannel.png
        """
```

```
        if not channel:
            channel = ChannelCustom()
        channel_UI = ChannelCustom_UIShort(channel=channel, settings=settings,
pathToSettingsFile=pathToSettingsFile)

        # Incrementing name of new channel
        namesOfChannel = [channel.name for channel in self.channels]
        if channel_UI.name in namesOfChannel:
            if re.search(r'^ch\d*$', channel_UI.name).group(0):
                channel_UI.name = 'ch' + str(getNumberOfChannel(
                    max(self.channels, key=lambda channel:
getNumberOfChannel(channel.name)).name) + 1)
            else:
                channel_UI.name = channel_UI.name + '1'

        self.acquisitions.addChannel(channel)

        removeChannelBtn = QPushButton()
        removeChannelBtn.setIcon(QIcon(QPixmap('resources/images/iconRemove.png')))
        removeChannelBtn.clicked.connect(lambda: self.onRemoveChannel(channel))

        hLayout = QHBoxLayout()
        hLayout.addWidget(removeChannelBtn)
        hLayout.addWidget(channel_UI)

        self.acquisitionsDict[channel] = hLayout
        self.acquisitionsLayout.addLayout(hLayout)

        return channel_UI

    def onRemoveChannel(self, channel, noAsk=False):
        """
        Scheme:

        .. image:: docImages/Acquisitions_UI.onRemoveChannel.png
        """
        if noAsk:
            answer = QMessageBox.Yes
        else:
            answer = QMessageBox.question(self, 'Remove channel', "Are you sure you want to
remove channel?",
                                          QMessageBox.Yes | QMessageBox.No, QMessageBox.No)
```

```
        if answer == QMessageBox.Yes:
            self.acquisitions.removeChannel(channel)

            for i in range(self.acquisitionsLayout.count()):
                hLayout = self.acquisitionsLayout.itemAt(i)
                if hLayout == self.acquisitionsDict[channel]:
                    hLayout.deleteLater()

            channelUIElems = self.acquisitionsDict.pop(channel)
            for i in reversed(range(channelUIElems.count())):
                channelUIElems.itemAt(i).widget().deleteLater()

    def onStart(self):
        self.acquisitions.start()
        for _, acqureLayout in self.acquisitionsDict.items():
            acqureLayout.itemAt(1).widget().currentState = STATES.RUNNING

        self.currentState = STATES.RUNNING

    def onResume(self):
        try:
            self.acquisitions.resume()
            for _, acqureLayout in self.acquisitionsDict.items():
                acqureLayout.itemAt(1).widget().currentState = STATES.RUNNING

            self.currentState = STATES.RUNNING
        except Exception as e:
            print('ERROR Acquisitions_UI_new onResume:', e)
            self.stopBtn.click()

    def onPause(self):
        try:
            self.acquisitions.pause()
            for _, acqureLayout in self.acquisitionsDict.items():
                acqureLayout.itemAt(1).widget().currentState = STATES.PAUSED

            self.currentState = STATES.PAUSED
        except Exception as e:
            print('ERROR Acquisitions_UI_new onPause:', e)
            self.stopBtn.click()

    def onStop(self):
```

```
    try:
        self.acquisitions.stop()
    except Exception as e:
        print('ERROR Acquisitions_UI_new onStop:', e)
    finally:
        for _, acqureLayout in self.acquisitionsDict.items():
            acqureLayout.itemAt(1).widget().currentState = STATES.READY
        self.currentState = STATES.READY

def onFreqChange(self, value):
    nValue = normalizeNumberEdit(value)
    if nValue is not None:
        self.acquisitions.freq = int(nValue)

def onAcqTimeChange(self, value):
    self.acquisitions.acqTime = value

def onMaximumSizeOfFileChange(self, value):
    self.acquisitions.maximumSizeOfFile = value * 1e6

def onTimePrecision(self, value):
    self.acquisitions.timePrecision = value

def onOtherValPrecision(self, value):
    self.acquisitions.otherValPrecision = value

def onIsSaveRawData(self, value):
    if value == 0 or value is False:
        self.acquisitions.isSaveRawData = False
        self.rawDataDirPathBtn.setDisabled(True)
        self.rawDataDirPathValue.setDisabled(True)
        self.isSaveInSeparateFilesCheckBox.setDisabled(True)
        self.timePrecisionEdit.setDisabled(True)
        self.otherValPrecisionEdit.setDisabled(True)
        self.maximumSizeOfFileEdit.setDisabled(True)

    else:
        self.acquisitions.isSaveRawData = True
        self.rawDataDirPathBtn.setDisabled(False)
        self.rawDataDirPathValue.setDisabled(False)
        self.isSaveInSeparateFilesCheckBox.setDisabled(False)
        self.timePrecisionEdit.setDisabled(False)
```

```
            self.otherValPrecisionEdit.setDisabled(False)
            self.onIsSaveInSeparateFiles(self.isSaveInSeparateFilesCheckBox.checkState())

    def onIsSaveInSeparateFiles(self, value):
        if value == 0 or value is False:
            self.acquisitions.isSaveInSeparateFiles = False
            self.maximumSizeOfFileEdit.setDisabled(True)
        else:
            self.acquisitions.isSaveInSeparateFiles = True
            self.maximumSizeOfFileEdit.setDisabled(False)

    def onRawDataDirPath(self):
        dname = QFileDialog.getSaveFileName(self, 'Choose path',

'{name}'.format(name=self.acquisitions.rawDataDirName))[0]

        if dname:
            if os.path.isdir(dname):
                self.notifiedWarning('Acquisition', 'Failed to create folder for data',
                                     detailedInfo='Folder already exists', show=True,
isError=True)
            else:
                self.acquisitions.dataDirPath = dname
                self.rawDataDirPathValue.setText(dname)

    def onLoadSettingsFile(self):
        fname = QFileDialog.getOpenFileName(self, 'Open file', '.\\', 'Settings files
(*.ini)')[0]
        if fname:
            self.loadSettings(getSettingsFromFile(fname))

    def onSaveSettingsToFile(self):
        fname = QFileDialog.getSaveFileName(self, 'Save File', '.\\settings_Acquisitions.ini',
                                            'Settings files (*.ini)')[0]
        if fname:
            saveSettingsToFile(self.acquisitions.getSettings(), fname)

    def connectStartButton(self, func):
        self.startBtn.clicked.connect(func)

    def connectPauseButton(self, func):
        self.pauseBtn.clicked.connect(func)
```

```
def connectResumeButton(self, func):
    self.resumeBtn.clicked.connect(func)

def connectStopButton(self, func):
    self.stopBtn.clicked.connect(func)

def disconnectStartButton(self, func):
    try:
        self.startBtn.clicked.disconnect(func)
    except:
        pass

def disconnectPauseButton(self, func):
    try:
        self.pauseBtn.clicked.disconnect(func)
    except:
        pass

def disconnectResumeButton(self, func):
    try:
        self.resumeBtn.clicked.disconnect(func)
    except:
        pass

def disconnectStopButton(self, func):
    try:
        self.stopBtn.clicked.disconnect(func)
    except:
        pass

def subscribeToWarning(self, subscriber):
    self.subsWarning.append(subscriber)

def subscribeToValues(self, subscriber):
    self.acquisitions.subscribeToValues(subscriber)

def notifiedWarning(self, *args, **kwargs):
    self.notifyWarning(*args, **kwargs)

def notifyWarning(self, *args, **kwargs):
    for sub in self.subsWarning:
        sub(*args, **kwargs)
```

```
    def loadSettings(self, settings):
        settingsCopy = copy.deepcopy(settings)
        missingAttrs = list()

        if checkAttrPresence('meta', settingsCopy, missingAttrs):
            meta = settingsCopy.pop('meta')
            if checkAttrPresence('type', meta, missingAttrs):
                if meta.pop('type') != 'acquisition':
                    raise Exception('Type of settings is not "acquisition"')
            else:
                self.notifiedWarning('Load settings', 'Unknown type of settings.', isError=False,
show=True)
        else:
            raise Exception('No meta section in settings!')

        if checkAttrPresence('data', settingsCopy, missingAttrs):
            data = settingsCopy.pop('data')
            if checkAttrPresence('frequency', data, missingAttrs):
self.freqEdit.setText(data.pop('frequency'))
            if checkAttrPresence('acqtime', data, missingAttrs):
self.acqTimeEdit.setValue(float(data.pop('acqtime')))
            if checkAttrPresence('issavedata', data, missingAttrs):
                issavedata = data.pop('issavedata').lower() not in ['false', '0', 'f', 'n', 'no']
                self.isSaveRawDataCheckBox.setChecked(issavedata)
                # WARNING: method 'onIsSaveRawData' is not called if use 'setChecked'
                self.onIsSaveRawData(issavedata)
            if checkAttrPresence('pathsavedata', data, missingAttrs):
                pathsavedata = data.pop('pathsavedata')
                self.acquisitions.rawDataDirName = pathsavedata
                self.rawDataDirPathValue.setText(pathsavedata)
            if checkAttrPresence('timeprecision', data, missingAttrs):
self.timePrecisionEdit.setValue(int(data.pop('timeprecision')))
            if checkAttrPresence('othervalprecision', data, missingAttrs):
self.otherValPrecisionEdit.setValue(int(data.pop('othervalprecision')))
            if checkAttrPresence('channels', data, missingAttrs):
                while len(self.channels):
                    self.onRemoveChannel(self.channels[0], noAsk=True)

                channelsStr = data.pop('channels').split(';')
                for channelStr in channelsStr:
                    self.onAddChannel(settings=ast.literal_eval(channelStr))
        else:
            raise Exception('No data section in settings!')
```