Всеукраїнський конкурс на кращу студентську наукову роботу
2019/2020 навчального року

Шифр: «Супутник»

Тема роботи: «Архітектура і реалізація симулятора для відпрацювання програмного забезпечення системи керування космічним апаратом»

Секція: **Комп'ютерна інженерія**

АНОТАЦІЯ

Наукова робота: 26 сторінок, 12 рисунків, 7 джерел.

Досліджено вимоги до архітектури програмного забезпечення (ПЗ), на базі яких побудовано архітектуру моделюючої частини ПЗ симулятора для відпрацювання програмно-алгоритмічного забезпечення (ПАЗ) системи керування (СК) космічним апаратом (КА) для створення програмного продукту, що є актуальним для розробників бортового програмного забезпечення (БПЗ) космічної галузі.

Метою роботи є *побудова архітектури моделюючої частини ПЗ симулятора з графічним інтерфейсом користувача*.

Наукова новизна роботи полягає в тому, що *за допомогою системного підходу до аналізу етапів та режимів відпрацювання БПЗ СК КА та з урахуванням алгоритмів функціонування СК КА запропоновано архітектуру ПЗ дослідницького стенду (індустріального симулятора) та графічний інтерфейс користувача для організації ефективного відпрацювання БПЗ СК КА.*

Практична цінність результатів роботи полягає в тому, що *розроблено ПЗ з графічним інтерфейсом користувача для технологічного комп'ютера (ТК) симулятора, яке дозволяє моделювати рух центру мас КА, кутовий рух КА, зовнішнє середовище та командні прилади СК, а також реалізувати інтерфейс обміну даними між ТК та платою мікроконтролера.*

Робота виконана в рамках НДР «Розробка програмного забезпечення дослідницького стенду для відпрацьовування програмно-алгоритмічного забезпечення системи управління космічного апарату», ДР № 0118U001666.

Основні положення й результати роботи доповідалися й обговорювалися на конференції «Тиждень науки-2019», ЗНТУ.

АРХІТЕКТУРА, ВІДПРАЦЮВАННЯ, ГРАФІЧНИЙ ІНТЕРФЕЙС, КОСМІЧНИЙ АПАРАТ, МОДЕЛЮВАННЯ, СИМУЛЯТОР, СИСТЕМА КЕРУВАННЯ, ТЕХНОЛОГІЧНИЙ КОМП'ЮТЕР

ЗМІСТ

ВСТУП

Керування космічним апаратом (КА) здійснюється бортовим комплексом керування (БКК) та передбачає комплекс взаємопов'язаних дій, спрямованих на досягнення цілей польоту з максимальною повнотою, безпечністю та надійністю. Система керування (СК) бортовою апаратурою є центральним елементом БКК. Поняття та концепція побудови БКК виходять з вимог системного підходу до проектування бортових засобів керування і контролю та реальної практики керування польотом КА різного класу і призначення [1].

Для відпрацювання програмно-алгоритмічного забезпечення КА використовують симулятори, під якими розуміють дослідницькі стенди (ДС), що дають змогу через моделювання вивчати та аналізувати динамічні системи шляхом імітації їх поведінки. Як правило, ДС складається із технологічного комп'ютеру (ТК) та комплекту розробника бортового програмного забезпечення (БПЗ), апаратною складовою якого є налагоджувальна плата.

У **першому розділі** визначається постановка задачі, вирішення якої відбувається у ході досліджень та роботи над проектом.

У **другому розділі** описано організацію розробки БПЗ – стисло розглядаються існуючі стандарти космічної галузі, етапи та життєвий цикл розробки.

У **третьому розділі** відбувається огляд існуючих програмних продуктів симуляторів та проводиться вибір засобів для розробки власного продукту.

У **четвертому розділі** вирішується питання архітектури програмного забезпечення симулятора.

У **п'ятому розділі** приводяться рішення щодо розробки графічного інтерфейсу користувача та взаємодії з ним розробника БПЗ.

1 ПОСТАНОВКА ЗАДАЧІ

Задача полягає у дослідженні режимів відпрацювання бортового програмного забезпечення системи керування космічним апаратом та розробці програмного забезпечення симулятора з графічним інтерфейсом користувача.

Актуальність дослідження полягає, по-перше, у визначенні підходів до розробки архітектури та графічного інтерфейсу ПЗ, які б відповідали вимогам до налагоджувального ПЗ космічної галузі та покращували ефективність розробки; по-друге, у вдосконаленні, спрощенні процесу відпрацювання БПЗ та створенні власного програмного продукту для промислового використання.

Основна робота при відпрацюванні БПЗ системи керування полягає в моделюванні в «замкнутому циклі» у прискореному й реальному часі руху центру мас (ЦМ) КА, кутового руху його навколо ЦМ, зовнішнього середовища, роботи командних приладів системи керування з використанням прототипу БПЗ СК. Алгоритми керування КА виконуються у вигляді прототипу БПЗ на мікроконтролері TMS570LS3137. Між програмними моделями, які працюють на ТК, і БПЗ має бути організовано обмін даними [2].

2 ОРГАНІЗАЦІЯ ПРОЦЕСУ РОЗРОБКИ БПЗ

2.1 Стандарти розробки БПЗ

Для європейських космічних проектів існує цілий ряд стандартів для розробки КА в цілому, це так звані ECSS-стандарти, розроблені й опубліковані Європейським агентством з космічної стандартизації (European Cooperation for Space Standardization). Для розробки ПЗ та симуляторів основними є наступні стандарти:

– ECSS-E-ST-40 Software engineering;

– ECSS-Q-ST-80 Software product assurance.

Крім ECSS-стандартів, також використовують стандарти аеронавігаційного ПЗ (Aeronautical Software Standards), стандарти на загальне ПЗ (Standards for general Software ANSI/IEEE), стандарти ПЗ для спеціальних космічних проектів (наприклад, Columbus Software Development Standard – CSDS, Galileo Software Standard – GSWS).

Окремим блоком вимог до симулятора є чітке дотримання аспектів проведення моделювання руху КА, його зовнішнього середовища та роботи командних приладів СК у реальному часі. Щоб було можливо хронологічно відслідковувати поточну діяльність, всі пакети обміну між ТК і мікроконтролером штампуються часовими підписами. Однак, необхідно розрізняти різні типи інформації про час, аби уникнути неправильних тлумачень. Зокрема, розрізняють:

– час сесії симулятора (Simulator Session Time) – фактичний час, який називається «zulu-time» або «місцевий час», в основному зазначається в універсальному часовому коді (UTC), в позначеннях типу «year:day:hr:min:s:ms». Цей час застосовується в якості метаданих (мітки часу) для службових пакетів (телеметріїї та службової інформації) та пакетів корисного навантаження;

– час виконання для симулятора (Simulation Runtime) – час в мілісекундах або секундах, що підраховується, коли симулятор обчислює, тобто не знаходиться в режимі «призупинення». Цей тип лічильника часу необхідний для чисельних обчислень всередині симулятора для інтеграції змінних стану, що залежать від часу;

– імітований час місії (Simulated Mission Time, SMT) – це час імітованої ситуації місії в космосі, яка в більшості випадків запланована на майбутнє, оскільки на момент моделювання КА все ще знаходиться у стадії збірки та ще не запущений. SMT зазвичай обчислюється в UTC або глобальній системі позиціонування (GPS);

– бортовий час (On-board Time, OBT) – це, як правило, лічильник в БПЗ, який починає рахувати під час завантаження апаратного чи імітованого бортового комп'ютера. Зазвичай OBT може бути встановлений за допомогою телекомунікацій до абсолютного значення – SMT, оскільки БПЗ також потрібна абсолютна інформація про час. Також тут у більшості випадків використовується UTC або GPS Time [3].

2.2 Етапи розробки ПЗ СК КА

На основі стандартизованих вимог сформовано наступні загальноприйняті етапи розробки ПЗ системи керування КА:

– «Algorithm in the Loop» (перевірка алгоритму) – повне функціональне моделювання супутникового та космічного середовища;

– «Software in the Loop» (перевірка ПЗ) – детальне відпрацювання програмно-алгоритмічного комплексу системи керування;

– «Controller in the Loop» (гібридна перевірка) – тест на сумісність БПЗ із апаратним забезпеченням і тест на сумісність БПЗ із імітованим супутником;

– «Hardware in the Loop» (гібридна перевірка) – розширення тестування до повної сумісності апаратного та програмного забезпечення з усім обладнанням.

У більшості проектів космічних апаратів стенд етапу «Algorithm in the Loop» обмежений алгоритмом перевірки положення та супутникової системи керування орбітою. «Software in the Loop» використовується не лише для бортових програмних тестів в області контролю положення чи орбіти, має також забезпечувати керування та моніторинг усіх функцій модельованого КА.

Розглядаючи бортовий комп'ютер (БК) з класичною комп'ютерною архітектурою, БПЗ складається з:

– операційної системи БК;

– програм керування КА, включаючи всі алгоритми керування;

– драйверів для інтерфейсів вводу та виводу між БК та супутниковим обладнанням;

– функцій, необхідних для телекомунікацій з наземною станцією.

«Hardware in the Loop» розвивається з «Controller in the Loop», інтегруючи все більше супутникових апаратних засобів. Відповідно до інтеграції реального обладнання, події моделі імітованого обладнання видаляються зі стенду. Для керування цим додатковим обладнанням КА або для його зовнішнього стимулювання з часом може знадобитися додаткове обладнання. Наприклад, може виникнути необхідність у бортовому обладнанні, такому як зоряний датчик, датчик магнітного поля Землі, таке інше [3].

2.3 V-модель життєвого циклу розробки

Застосування V-подібної моделі життєвого циклу дозволяє сконцентрувати увагу на перевірці результатів розробки, точно спланувати, які властивості програмного забезпечення будуть досліджуватися, на яких етапах і на підставі якої документації. Саме такий підхід до вибору моделі життєвого циклу розробки характерний для бортових СК КА. Усі перераховані етапи розробки БПЗ проходять V-подібний життєвий цикл (див. рис. 1).

Життєвий цикл розробки, оснований на V-моделі, дозволяє мінімізувати ризики, тобто зробити проект більш «прозорим», підвищити якість результатів, зменшити загальну вартість проекту та покращити рівень комунікації між учасниками проекту.

Більшість перерахованих переваг досягається завдяки можливості тестування системи на ранніх етапах розробки. При цьому розрізняють поняття верифікації та валідації системи. Верифікація відповідає за дотримання всіх

норм якості продукту при його реалізації, а валідація – за відповідність кінцевого продукту очікуванням та вимогам замовника.

Рисунок 1 – V-модель життєвого циклу

3 ВИБІР ТА ОБҐРУНТУВАННЯ ЗАСОБІВ РОЗРОБКИ ПЗ СИМУЛЯТОРА

Перед початком пошуку власних рішень для створення ПЗ, яке б забезпечувало відпрацювання БПЗ у всіх необхідних режимах, та створення власного GUI (Graphic User Interface – графічний інтерфейс користувача) були розглянуті приклади існуючих середовищ моделювання складних динамічних систем, такі як:

– Simulink – імітаційний набір, що надає можливість графічно моделювати динамічні системи. Комерційно доступну бібліотеку зі стандартними елементами для моделювання системи, теорії керування та інженерії можна розширити за допомогою самостійно запрограмованих

функціональних блоків, що можуть бути реалізовані на мовах FORTRAN або C/C++;

– базова версія «Spacecraft Control & Operation System», що є безкоштовною для використання в проектах у державах-членах ESA (European Space Agency), однак для повної функціональності випробувальних стендів потрібні комерційні оновлення.

Однак, існуючі програмні засоби для відпрацювання є, по-перше, іноземного походження, а по-друге, здебільше загально-орієнтовані, як, наприклад, Simulink. Тому створення власного ПЗ має надавати такі переваги, як підтримка декількох мов інтерфейсу, зручність задання параметрів моделювання та мінімізація виникнення помилок при їх формуванні.

Для ПЗ симулятора, за допомогою якого проводять відпрацювання БПЗ у прискореному та реальному часі, було обрано й використано ряд засобів, що разом визначили структуру ПЗ індустріального дослідницького стенду (див. рис. 2). Наведена пакетна структура ПЗ ДС має чотири головних гілки: системне ПЗ (System Software), інструментарій (Tools), додатки (Applications) та сервісне ПЗ (Service Software).

Аналіз існуючих засобів розробки призвів до обрання мови програмування C/C++ та кросплатформного середовища розробки Qt Creator. Вибір мови програмування для відпрацювання БПЗ визначили її переваги: швидкодія, економія пам’яті і зрозумілість у вирішенні питань із взаємодією з апаратним забезпеченням.

Крім того, було використано комплект розробника БПЗ TMS570LS31x Hercules TM Development Kit (HDK), що складається з налагоджувальної плати на базі мікроконтролера TMS570LS3137 та сервісного ПЗ.

Системне ПЗ включає в себе операційну систему (ОС) Windows 7 та ОС Ubuntu 18.04 і вбудовані в них сервісні програми та утиліти.

У ході роботи над проектом було використано середовище QtLinguist, утиліти lupdate і lrelease для створення файлів з перекладами GUI та інструмент binarycreator для створення інсталятора ПЗ.

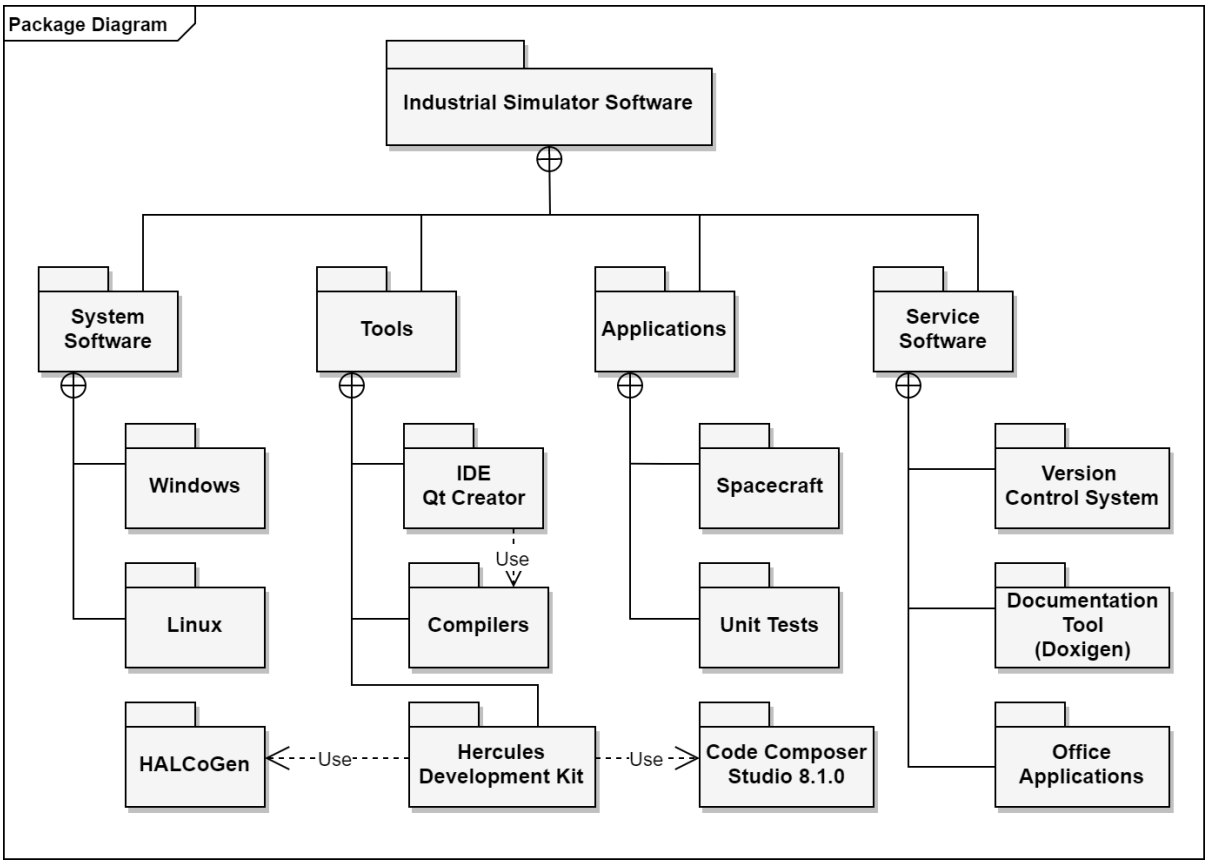

Рисунок 2 – SysML-діаграма пакетної структури індустріального ДС

4 ПОБУДОВА АРХІТЕКТУРИ ПЗ СИМУЛЯТОРА

4.1 Структура моделюючої частини ПЗ симулятора

Модулі моделюючої частини ПЗ симулятора – Spacecraft – згруповані в пакети відповідно до їх функціонального призначення (див. рис. 3). Більшість модулів реалізовані у вигляді класів C++, які містять параметри моделей і функції реалізації моделей.

Стисле пояснення структури моделюючої частини ПЗ:

– вміст пакету CenterMassMotion реалізує модель руху центру мас КА і обчислює Гринвіцький зоряний час;

– вміст пакету AngularMotion моделює кутовий рух КА;

– в пакеті Environment містяться класи, що моделюють гравітаційний потенціал і магнітне поле Землі;

– пакет Sensors містить класи: що моделюють магнітометр, вимірювачі кутових швидкостей; зоряний датчик;

– пакет Actuators містить класи для моделювання керуючих двигунів-маховиків та класи, що моделюють електромагніти;

– вміст пакету Model призначено для узгодження форматів даних, якими обмінюються моделююча програма ТК і алгоритм керування СК призначений для організації процесу моделювання в режимі «Algorithm in the Loop»;

– в пакеті Tools зосереджені додаткові класи, що необхідні для організації математичних обчислень, а також клас XMLParser, призначений для читання xml-файлів з параметрами моделей та клас FileInOut, функції якого записують результати моделювання у відповідні файли на диску для подальшого аналізу і побудови графіків;

– пакет XML містить файли, в яких записані початкові умови моделювання і параметри моделей;

– пакет OBSW містить файл прототипу бортової керуючої програми СК і допоміжний файл для роботи з матрицями, векторами та кватерніонами в бортовій програмі.

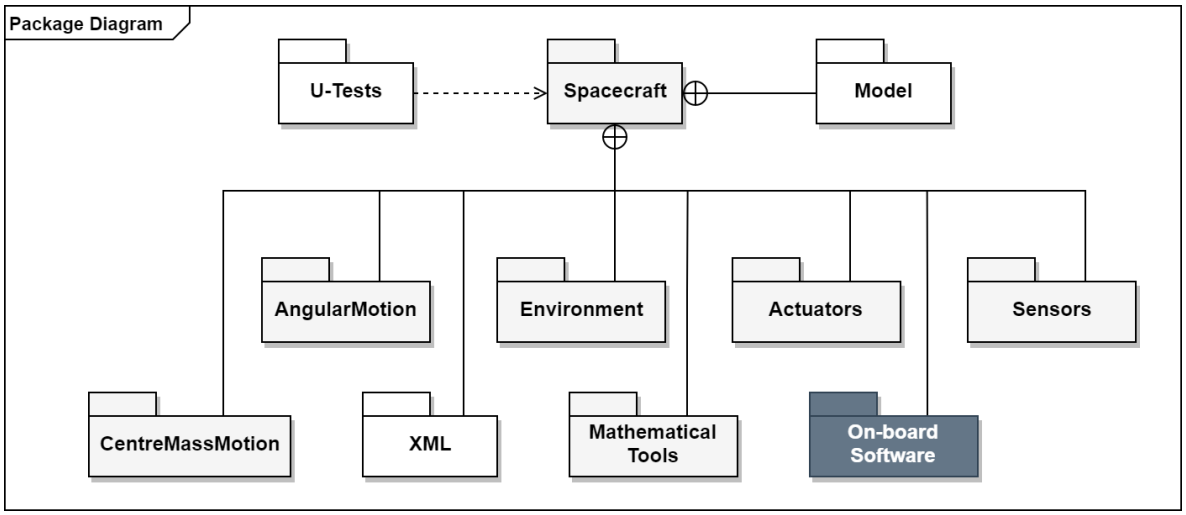

Рисунок 3 – Організація пакетів моделюючого ПЗ

4.2 Режими відпрацювання БПЗ

Відпрацювання БПЗ полягає у моделюванні руху КА і командних приладів СК, яке, в свою чергу, полягає у виконанні обчислень всіх модулів програми та інтегруванні диференціальних рівнянь, що описують рух центру мас КА, кутового руху і динаміку роботи двигунів маховиків.

ПЗ передбачає три режими проведення відпрацювання БПЗ:

– режим моделювання в прискореному часі без підключення до мікроконтролера (програма отримує вихідні дані, проводить обчислення на процесорі ТК і користувач через короткий проміжок часу отримує числові та графічні результати);

– режим моделювання в прискореному часі з підключенням до мікроконтролера (програма отримує вихідні дані, обчислення відбуваються на процесорі мікроконтролера, після закінченнях яких на екран ТК виводяться числові та графічні результати);

– режим моделювання в реальному часі з підключенням до мікроконтролера (програма отримує вихідні дані, обчислення відбуваються на процесорі мікроконтролера із заданою тривалістю, при цьому на екран ТК виводяться поточні числові та графічні результати, користувач має можливість впливати на хід моделювання – призупиняти, відновлювати чи примусово скасовувати).

4.3 Архітектурні рішення

Для виконання поставленої задачі було необхідно знайти таке архітектурне рішення для моделюючої частини ПЗ, яке б давало можливість

удосконалити процес відпрацювання БПЗ, задовольняючи всі наявні потреби, та залишити шляхи для його модернізації або видозміни у майбутньому.

Концепція відкритої архітектури забезпечує проведення робіт згідно загальним принципам з використанням обмеженого набору уніфікованих модулів та стандартних інтерфейсів, які у разі потреби можуть бути розширені. Дотримання ідеї модульного ПЗ та відповідного компонентного програмування дозволяє суттєво спростити та прискорити процес створення реального БПЗ.

Після аналізу режимів відпрацювання та відповідних вимог до ПЗ симулятора було прийнято рішення взяти за основу архітектурний шаблон MVC (Model-View-Controller), тобто розподіляти код програми на три відповідні блоки: «Model» (Модель), «View» (Вид) та «Controller» (Контролер) (див. рис. 4). Вибір шаблону пояснюється тим, що MVC дає можливість реалізувати наступні концепції:

– одна реалізація моделі даних – багато видів її представлення;

– гнучкість у зміні будь-якого існуючого чи створенні нового модуля ПЗ;

– розподіл бізнес-логіки та графічного інтерфейсу (можливість виконання роботи різними розробниками);

– зручність застосування при розпаралелюванні програми на декілька потоків.

При використанні шаблону MVC організація зв’язків між функціональними блоками у різних проектах може дещо відрізнятися. Наприклад, при наявності лише одного варіанту виводу результатів блок «Model» міг би відразу відправляти дані блоку «View». Але у даному випадку програма має декілька формулярів відображення обчислених даних, крім того, у фреймворку Qt не доцільна організація доступу до елементів графічного інтерфейсу із сторонніх класів, тому оновлення користувацького інтерфейсу вікна (формуляра) має відбуватися лише з класу того формуляра, який має відповідний покажчик (ui). Виходячи з цього, блок «Controller» виступає як посередник між блоками «Model» та «View».

Сама організація передачі даних між функціональними блоками відбувається завдяки сигнально-слотовим зв’язкам, які є зручною особливістю фреймворку Qt.

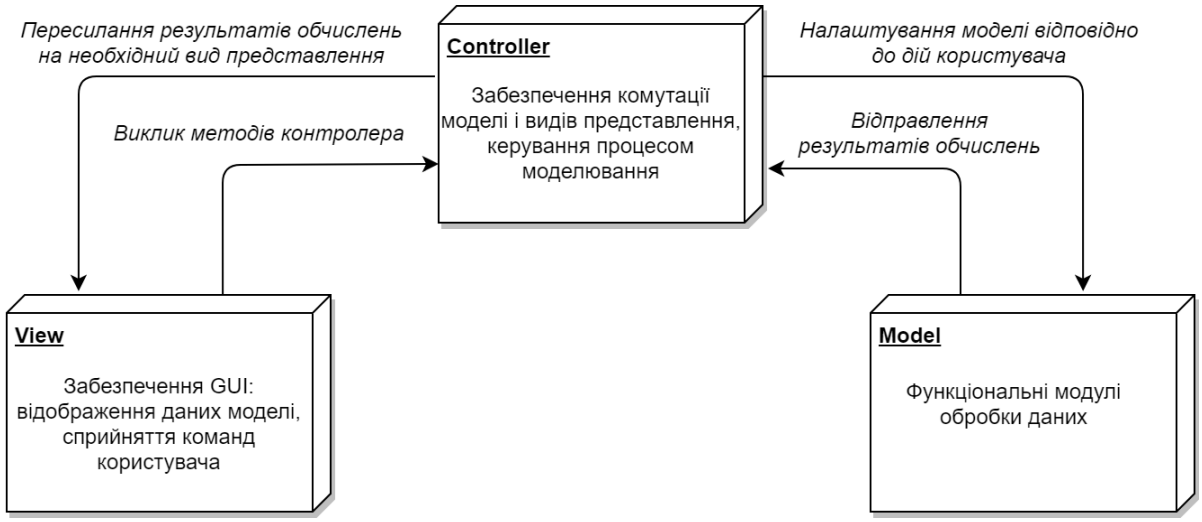

Рисунок 4 – Архітектура моделюючої частини ПЗ симулятора на основі шаблону MVC

Відповідно до допрацьованої архітектури було доповнено організацію пакетів моделюючої частини ПЗ пакетами View та Controller (див. рис. 5). Пакет View містить класи різних видів представлення, а пакет Controller – різні типи контролерів із своєю логікою для проведення моделювання.

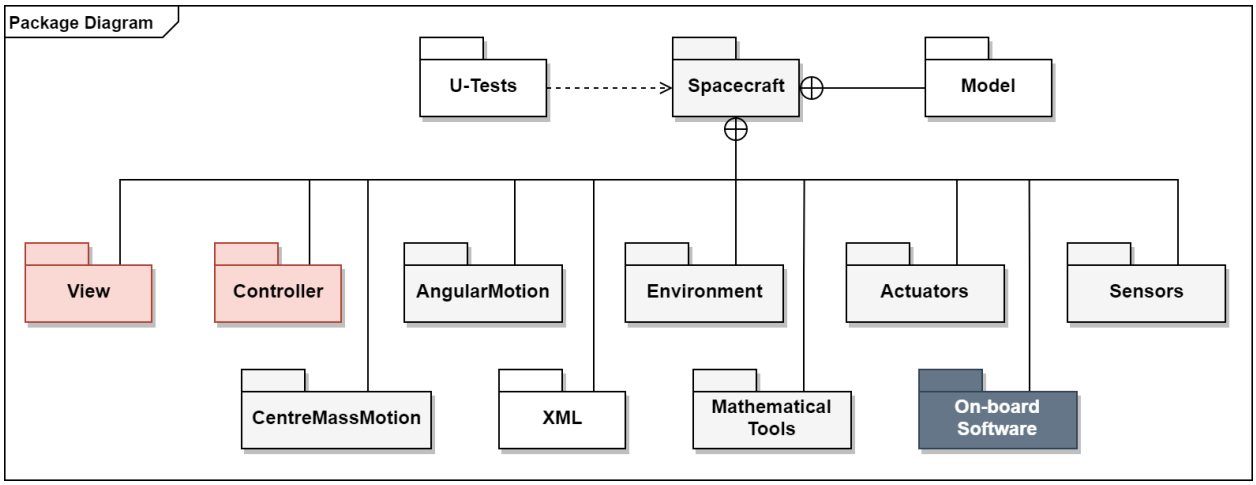

Рисунок 5 – Організація пакетів допрацьованого моделюючого ПЗ

Ініціалізація, хід виконання моделювання та можливий контроль над ним відбувається у блоці Контролера. Так як у програмі можливе моделювання у реальному та прискореному часі, то потрібне створення двох відповідних класів контролерів, які б мали наслідування від абстрактного класу контролера зі спільними методами (див. рис. 6). Наведені лістинги 1 та 2 демонструють різницю організацій циклу процесу обчислень в різних класах контролерів.

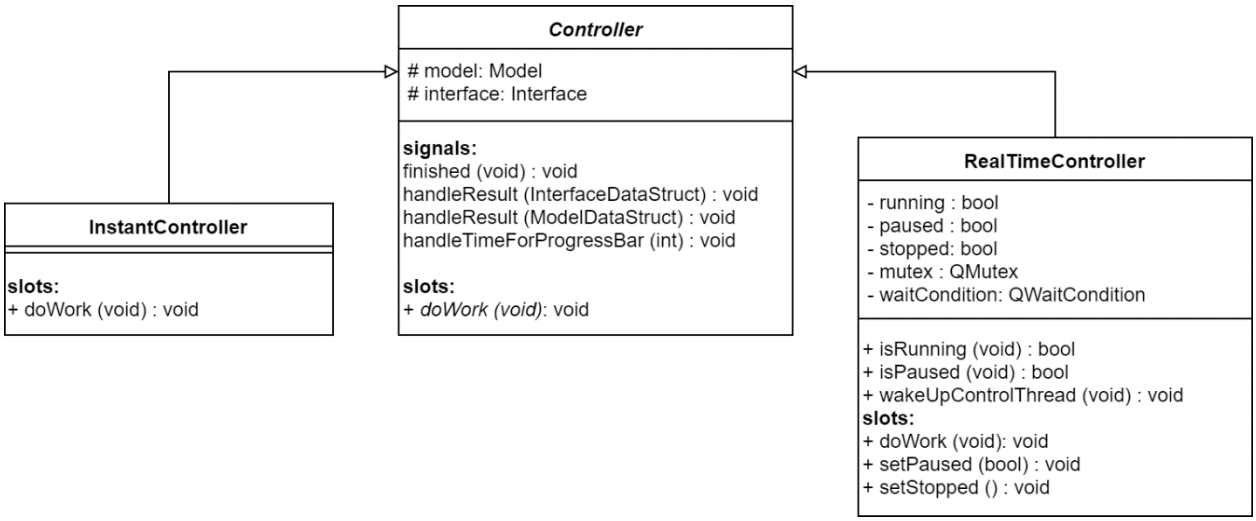

Рисунок 6 – UML-діаграма класів блоку Controller

Лістинг 1 – реалізація роботи контролера режиму прискореного моделювання

```
void InstantController::doWork()
{
    RungeKuttaMethod rk(model.n,model.h);
        for(;model.t<=model.T;)
        {
            // ...
            rk.integralRK(model.y,f,&model);
            model.t+=model.h;
            model.modeling();
            interface.interface(model);
        }
        // ...
        qInfo(logInfo()) << "Instant Simulation is done";
}
```

Лістинг 2 – реалізація роботи контролера режиму моделювання реального часу

```
void RealTimeController::doWork()
{
    running = true;
    qInfo(logInfo()) << "Simulation is started in real-time";

    RungeKuttaMethod rk(model.n,model.h);
        for(;model.t<=model.T;)
        {
            // ...
            rk.integralRK(model.y,f,&model);
            model.t+=model.h;
            model.modeling();
            interface.interface(model);

            if(paused){ // реалізація призупинення обчислень
                qInfo(logInfo()) << "Simulation is paused";
                QMutexLocker locker(&mutex);
                while(paused){
                    waitCondition.wait(&mutex);
                }
                qInfo(logInfo()) << "Simulation is resumed";
            }

            if(stopped) { // реалізація зупинки обчислень
                running = false;
                qInfo(logInfo()) << "Simulation is stopped";
                emit finished();
                return;
            }
        }
        // ...
        running = false;
        qInfo(logInfo()) << "Simulation is finished";
        emit finished();
}
```

4.4 Реалізація багатопочності

Для можливості впливу на хід моделювання руху КА і командних приладів СК та для забезпечення при цьому своєчасного оновлення інтерфейсу користувача необхідно організувати паралельність виконання цих дій.

Розпаралелювання задач має бути досягнуто шляхом розподілу обчислень та операцій вводу-виводу на кілька потоків (див. рис. 7). Головним потоком у програмах, створених за допомогою Qt, завжди є GUI-thread, тому його не слід навантажувати довготривалими операціями, які б могли позбавити відгуку інтерфейсу на дії користувача. Отже, головний потік має відповідати за:

– запуск нових потоків;
– оновлення графічного інтерфейсу;
– реакцію на дії користувача.

Дочірній потік GUI-thread – Controller-thread зобов’язаний:

– створювати об'єкти моделювання;
– контролювати процес моделювання;
– пересилати результатів обчислень на потрібний "екран".

Внутрішній цикл обчислень з опитуванням щодо надходження команд "пауза" або "стоп" також відбувається у Controller-thread.

Рисунок 7 – Організація потоків виконання програми.

Для зв’язку об’єктів, у тому числі із різних потоків, та передачі повідомлень між ними використовується механізм сигналів та слотів, що дозволяє забезпечити об’єктно-орієнтовану комунікацію. Кожний

успадкований від класу QObject об’єкт здатен отримувати та надсилати сигнали та/або слоти. Сигнали визначаються у класі заголовків, як звичайні функції, але без реалізації. Слоти майже нічим не відрізняються від звичайних функцій, але вони приєднуються до сигналів, тобто їх виконання відбувається тоді, коли був відправлений певний сигнал (проте як один слот можна приєднати до багатьох сигналів, так і один сигнал – до багатьох слотів).

З’єднання об’єктів відбувається за допомогою статичного метода connect(), котрий визначений у класі QObject:

QObject::connect (constQObject* sender,constchar* signal, constQObject* receiver, constchar* slot, Qt::ConnectionTypetype = Qt::AutoConnection);

Параметр type керує режимом обробки, має три можливих значення: Qt::DirectConnection – сигнал обробляється одразу викликом відповідного слоту, Qt::QueuedConnection – сигнал перетворюється у подію та стає в спільну чергу для обробки, Qt::AutoConnection – автоматичний режим, згідно якому якщо об’єкт, котрий відправив сигнал, знаходиться в одному потоці з приймаючим його об’єктом, то встановлюється режим Qt::DirectConnection, у протилежному випадку – режим Qt::QueuedConnection [4].

У Qt можлива реалізація багатопоточності декількома способами – QThread, QRunnable/QThreadPool, QtConcurrent::run(), QtConcurrent (Map, Filter, Reduce) і QWorkerScript (QML). Кожен із способів має свої переваги та недоліки, які досить детально описані в офіційній документації фреймворку Qt.

Найбільше задовольняє вимогам з контролю над процесом виконання моделювання в одному окремому потоці спосіб наслідування класу-виконавця (контролера) від класу QObject та після створення його об’єкту та об’єкту класу QThread, переміщення контролера до нового створеного потоку.

Приклад застосування такого способу:

```
    RealTimeController realTimeController = new
RealTimeController();
    ControllerThread controllerThread = new QThread(this);
    realTimeController->moveToThread(controllerThread);
```

```
    connect(controllerThread, SIGNAL(started()),
realTimeController, SLOT(doWork()));
    controllerThread->start(); // запуск потоку на виконання
```

5 ОРГАНІЗАЦІЯ ВЗАЄМОДІЇ РОЗРОБНИКА БПЗ ІЗ СИМУЛЯТОРОМ

5.1 Моделювання сценаріїв роботи

Виходячи з описаних вище трьох режимів відпрацювання БПЗ, було детально розглянуто відповідні сценарії взаємодії користувача з програмою в даних режимах. Для простоти розуміння сценаріїв було побудовано діаграми послідовностей дій, а також складено розгорнуті описи прецедентів функціоналу ПЗ [5].

Для прикладу наведено один з можливих сценаріїв дій при роботі з GUI моделюючої частини ПЗ симулятора. Відповідна діаграма відображає послідовність повідомлень між об’єктами, тобто послідовність викликів об’єктами методів, які завершуються виконанням деяких задач (див. рис. 8). Зображену на діаграмі послідовність можна описати наступним чином.

Крок 1. Після запуску програми було відкрито вікно головного меню (MainWindow), де обрано режим моделювання в реальному часі з підключенням до мікроконтролера, який відкриває відповідне вікно формуляра (RealTimeForm). Відкриття вікна RealTimeForm зумовлює приховання вікна MainWindow.

Крок 2. Підготовка до проведення моделювання – завантаження xml-файлу з параметрами моделювання та його перевірка, перевірка зв’язку ТК з платою мікроконтролера.

Крок 3. При успішному виконанні попереднього кроку запуск процесу моделювання – створення об’єктів контролеру (відповідне створення контролером необхідних об’єктів моделювання) та потоку.

Крок 4. Цикл процесу моделювання, у ході якого відбувається пересилання результатів контролером до формуляра RealTimeForm.

Крок 5. Встановлення процесу моделювання на паузу.

Крок 6. Відновлення процесу моделювання.

Крок 7. Примусове завершення процесу моделювання, після чого звільняється пам’ять, що була виділена під об’єкти контролеру та його потоку, руйнуються об’єкти моделювання.

Крок 8. При поверненні до головного меню відбувається виклик деструктора вікна формуляра RealTimeForm.

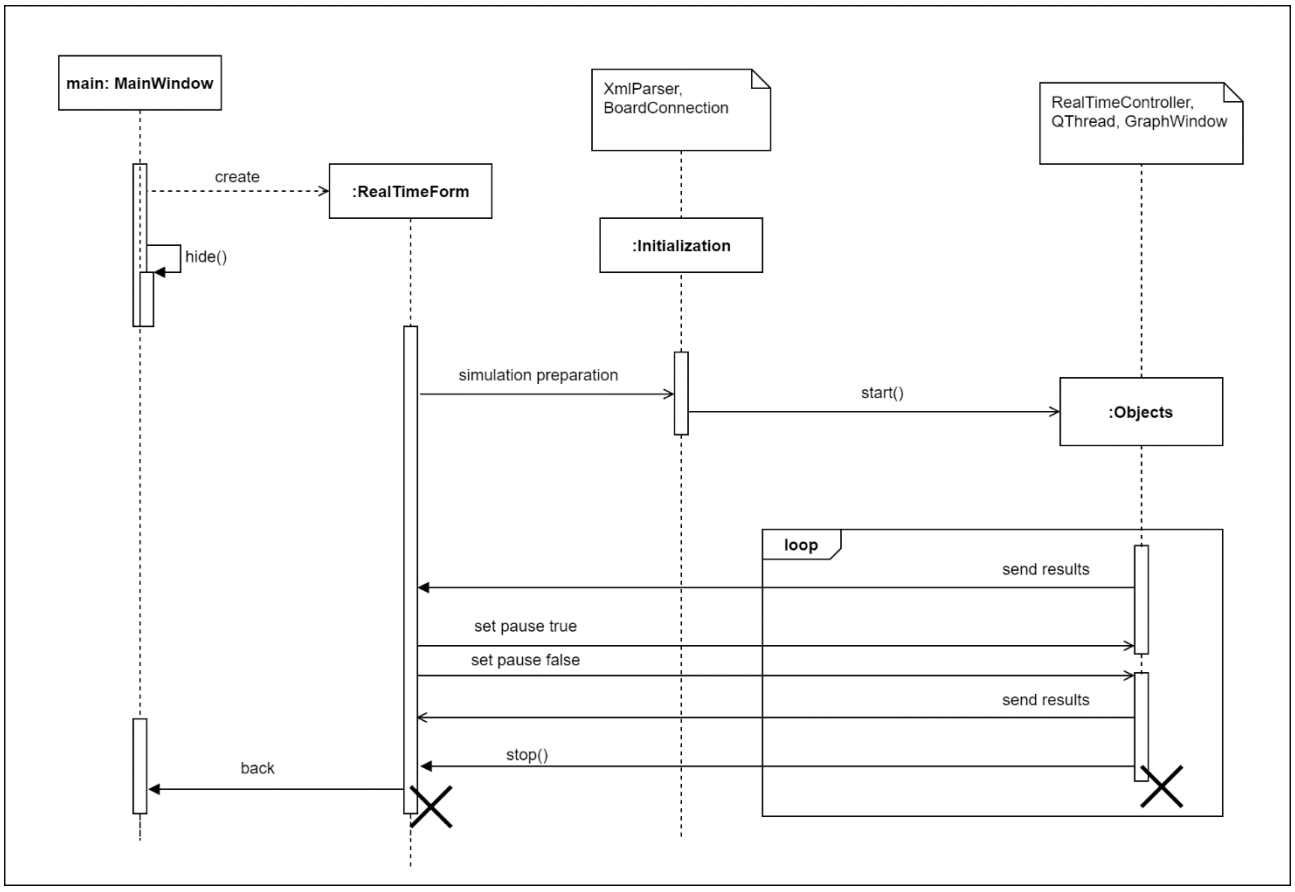

Рисунок 8 – UML-діаграма сценарію послідовностей дій при роботі з GUI для моделювання в реальному часі

5.2 Реалізація графічного інтерфейсу користувача

GUI було реалізовано за допомогою використання IDE Qt Creator, візуального редактора Qt Designer та написання qss-файлів (Qt Style Sheets – таблиці стилей Qt) із правилами стилю (див рис. 9-12).

QSS в значній мірі є подібним до каскадних таблиць стилів (CSS), внаслідок чого має схожий синтаксис. Зокрема, як і в CSS, в QSS можна змінювати форму, кольори, прозорість елемента, а також візуальну реакцію на події (такі, як натискання кнопки). Таким чином, можливо з легкістю створювати різні теми оформлення та редагувати їх.

Для розробки GUI програми був обраний метод ітераційного прототипування [6]. Qt Designer використовувався лише для створення прототипів графічних інтерфейсів користувача. Qt Designer та графічний редактор у Qt Creator мають ряд подібних інструментів, тому розробка вже функціонального інтерфейсу відбувалась у Qt Creator. В «Панелі віджетів» доступні для використання елементи інтерфейсу – віджети, такі як, наприклад, «випадаючий список» (ComboBox), «поле вводу» (LineEdit), «кнопка» (PushButton) та багато інших. Кожен віджет має свій власний набір властивостей, що визначається відповідним йому класом бібліотеки Qt [7].

Графічні результати виконаних обчислень надаються в окремому вікні з п’ятьма вкладками, кожна з яких містить графік певного процесу. Побудова графіків у режимі реального часу була реалізована завдяки використанню бібліотеки QCustomPlot. Графіки можливо експортувати в різні формати, такі як векторні PDF-файли чи растрові зображення, такі як PNG, JPG та BMP.

Таким чином, у вікні формуляра режиму моделювання реального часу користувач може керувати процесом моделювання, натискаючи кнопки «Start» (Старт), «Pause» (Пауза), «Stop» (Стоп); бачити поточні числові результати у прямокутних віконцях; слідкувати за побудовою графічних результатів у окремому вікні, відкриття якого відбувається при натисканні кнопки «Show

graphs» (Показати графіки); зберігати графічні результати у потрібному форматі при натисканні кнопки «Save graphs» (Зберегти графіки).

Вікно для редагування користувачем xml-файлів з параметрами моделювання з оболонки програми містить відповідні поля вводу та керуючі кнопки для відміни або застосування змін.

Крім того, рядок стану внизу вікна сповіщає про поточний стан моделювання, наявність або відсутність помилок при завантаженні файлу з параметрами для моделювання, коректності підключення плати мікроконтролера, збереженні змін у xml-файлах.

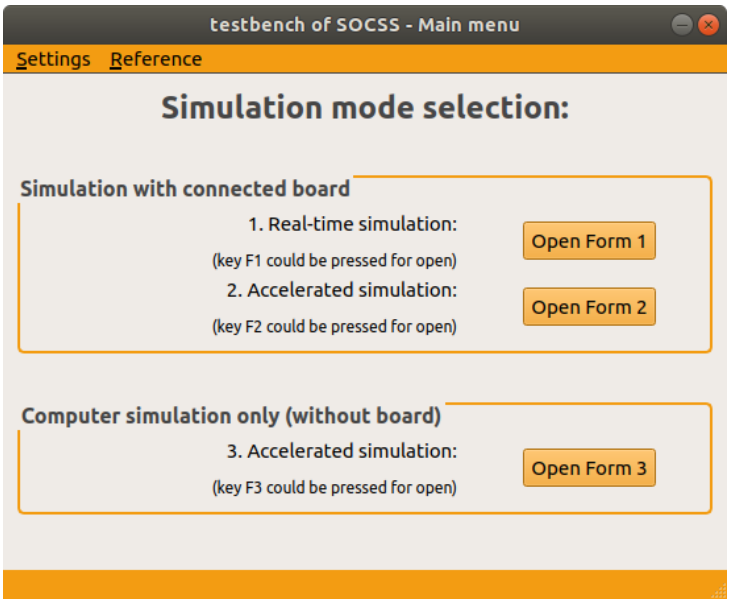

Рисунок 9 – Вікно головного меню програми

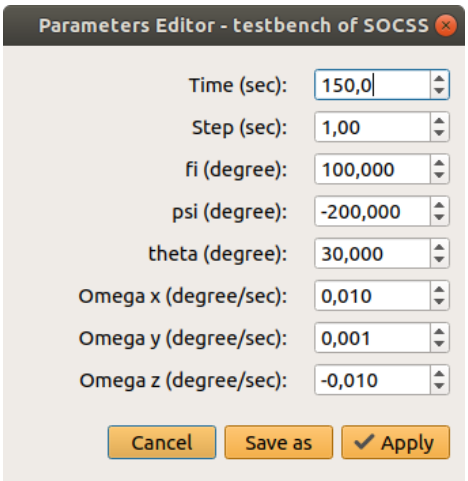

Рисунок 10 – Вікно редагування параметрів моделювання

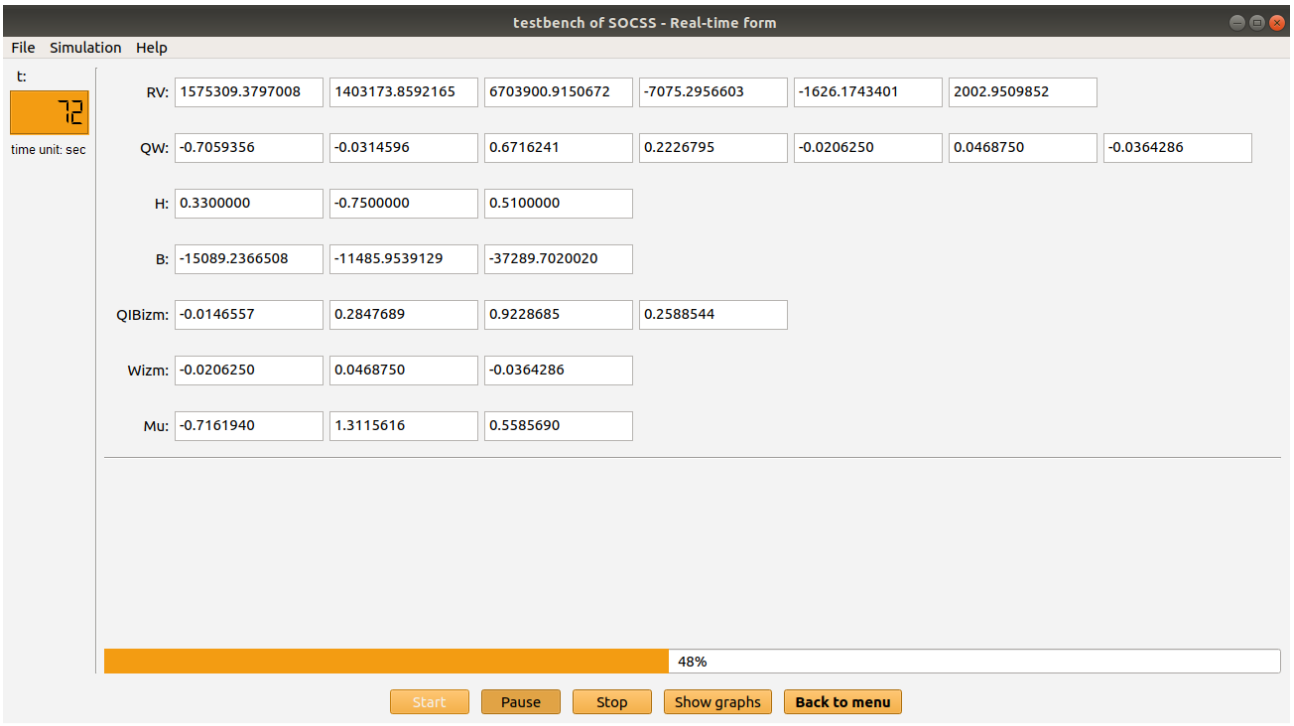

Рисунок 11 – Вікно формуляра моделювання в реальному часі

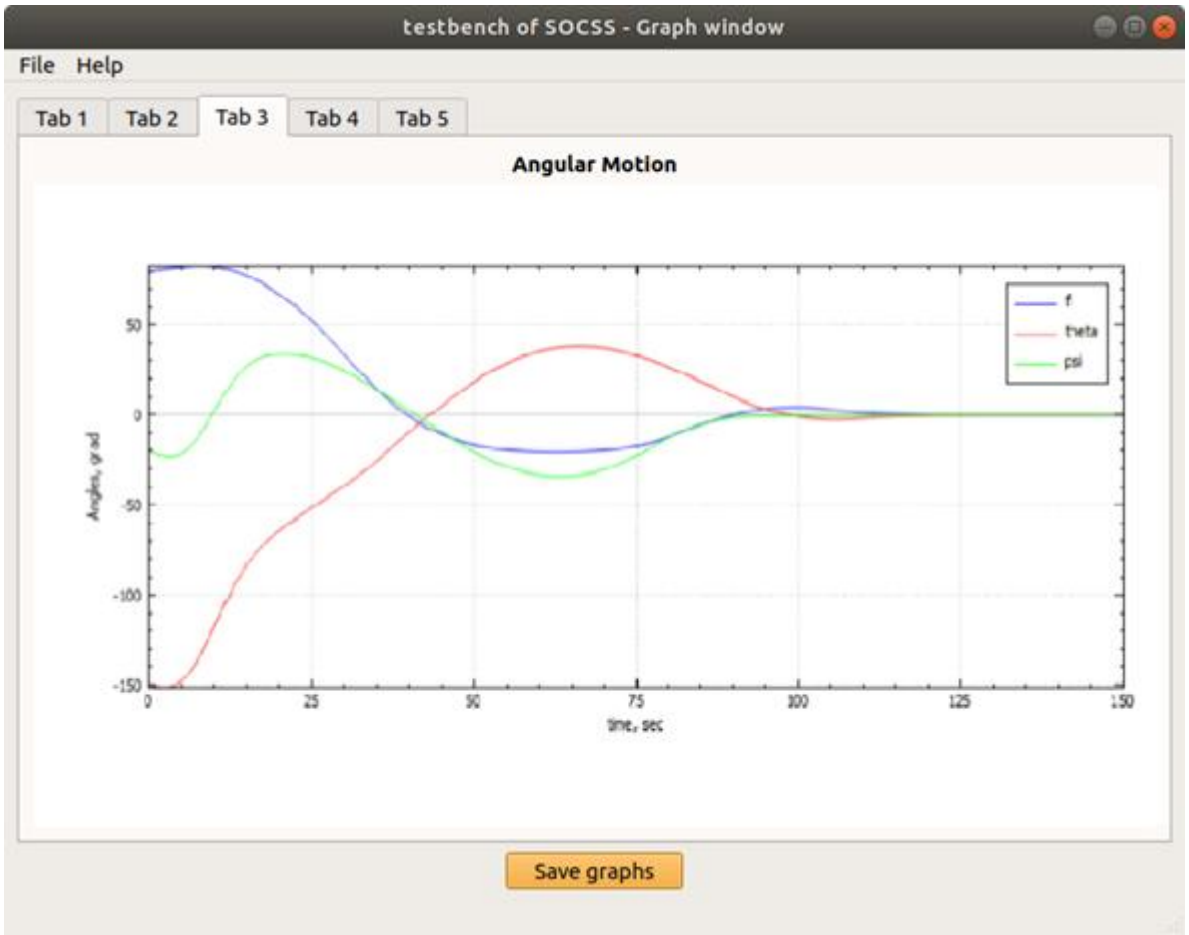

Рисунок 12 – Вікно графіків

ВИСНОВКИ

У процесі роботи було досліджено задачі відпрацювання БПЗ СК КА, розглянуто стандарти і підходи до розробки БПЗ, визначено загальноприйняті етапи розробки БПЗ, на основі яких сформовано вимоги до архітектури ПЗ симулятора для відпрацювання програмного-алгоритмічного забезпечення СК КА.

Побудовано архітектуру моделюючої частини ПЗ симулятора, на основі якої було створено програмний продукт, маючий практичну цінність для розробників БПЗ.

Для розробки сучасного GUI було досліджено різні підходи до його розробки, серед яких обрано метод ітераційного прототипування, що є найбільш зручним і задовольняючим потреби користувача.

Планується доповнення ПЗ симулятора використанням тривимірної графіки для відображення графічних результатів моделювання, зокрема, візуалізації руху КА.

Крім того, дослідження за цим напрямком можуть бути продовжені для впровадження розширеного комплексу симуляторів для відпрацювання інших службових систем КА, наприклад таких, як система енергозабезпечення, система збирання та обробки телеметрії.

ПЕРЕЛІК ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1 Мацяшек Л. А. Практическая программная инженерия на основе учебного примера / Л. А. Мацяшек, Б. Л. Лионг – Пер. с англ. – 3-е изд. – М.: БИНОМ. Лаборатория знаний, 2015. – 959 с.: ил.

2 Звіт про НДР «Розробка програмного забезпечення дослідницького стенду для відпрацьовування програмно-алгоритмічного забезпечення системи управління космічного апарату» за договором від 15.05.2018 р. №4818 (остаточний), наук. керівник договору к.т.н. доц. Кудерметов Р.К. – Запоріжжя: ЗНТУ, 2018. ДР № 0118U001666.

3 Simulating Spacecraft Systems / J. Eickhoff – Germany: Springer, 2009. – 358 p.

4 Шлее М. Qt 5.10. Профессиональное программирование на C++ / М. Шлее – СПб.: БХВ-Петербург, 2018. – 1072 с.

5 About Face: The Essentials of Interaction Design, 4th Edition / A. Cooper, R. Reimann, D. Cronin, C. Noessel. – USA: Wiley, 2014. – 720 p.

6 Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development, 3rd Edition / C. Larman – Addison Wesley Professional, 2004. – 736p.

7 Кармелюк К.О., Юнусов О.І., Польська О.В. Розробка GUI програмного забезпечення дослідницького стенду [Електронний ресурс] / К.О. Кармелюк, О.І. Юнусов, О.В. Польська // Тиждень науки-2019. Факультет комп’ютерних наук і технологій. Тези доповідей науково-практичної конференції, 15-19 квітня 2019, Запоріжжя: ЗНТУ, 2019. ISBN 978-617-529-224-2.