УДК 004.05
DOI https://doi.org/10.32782/2663-5941/2023.6/19

Петросян А.Р.
Державний університет «Житомирська політехніка»

ОРГАНІЗАЦІЯ ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ БОРТОВОГО КОМП'ЮТЕРА БЕЗПІЛОТНОГО ПОВІТРЯНОГО СУДНА

Останнім часом безпілотні повітряні судна демонструють значне технологічне зростання, що викликає інтерес не тільки у пересічних громадян, а й у військовій, промисловій та цивільній сферах діяльності.

Проєктування та розробка програмного забезпечення управління орієнтацією безпілотного повітряного судна у просторі потребує тривалого часу, включаючи час на відлагодження, тестування, проведення льотних випробувань. Помилки, допущені при розробці програмного забезпечення, можуть спричинити виникнення аварійних ситуацій або інших непередбачених збоїв у процесі експлуатації, що може привести до руйнування апарату та заподіяння шкоди людям та навколишньому середовищу. Відлагодження та тестування програмного забезпечення безпілотного повітряного судна є складним завданням, тому застосовують спеціальні програмні та апаратні засоби.

Проаналізовано останні дослідження підходів до тестування програмного забезпечення вбудованих систем з виявленням переваг і недоліків. Аналіз показав, що розроблюють як механічні випробувальні стенди, так і використовують імітаційне моделювання (симуляцію) у 3D-середовищі: SIL та HIL. Механічний випробувальний стенд дозволяє перевіряти і калібрувати параметри моделі та виконувати керування в реальному часі багатороторним безпілотним повітряним судном, а також забезпечує рух по тангажу, крену та рисканню. SIL-симуляція дозволяє легко організувати тестування, оскільки не потребує додаткового обладнання. тому її можна використати на ранніх етапах розробки програмного забезпечення ще до того, як воно буде інтегровано в цільове обладнання. HIL-симуляція включає використання цільового обладнання, що максимально наближає роботу системи до реальних умов.

Обґрунтовано важливість та запропоновано удосконалений варіант HIL-сімуляції. Основу даного варіанту складає симулятор датчиків та актуаторів, який забезпечує роботу оригінальної прошивки безпілотного повітряного судна та дозволяє організувати взаємозв'язок бортового комп'ютера та персонального комп'ютера. Для перевірки ідеї, система була реалізована на практиці. Також розроблено утиліту конфігурування симулятора датчиків та актуаторів.

Ключові слова: *БПС, безпілотні повітряні судна, бортовий комп'ютер, тестування програмного забезпечення, HIL-симуляція, 3D-середовище, симулятор датчиків та актуаторів.*

Постановка проблеми. Безпілотні повітряні судна (БПС), які є літаючими роботами, становлять важливу частину наукових досліджень у військовій, промисловій та цивільній сферах діяльності: аерофотозйомка та картографування, оперативне отримання інформації про наслідки надзвичайних ситуацій, моніторинг за об'єктами промисловості та природних комплексів, доставка товарів, у розважальних цілях тощо. Проєктування та розробка програмного забезпечення управління орієнтацією БПС у просторі (автопілота) потребує тривалого часу, включаючи час на відлагодження, тестування, проведення льотних випробувань. Помилки, допущені при розробці програмного забезпечення БПС, можуть спричинити виникнення аварійних ситуацій або інших непередбачених збоїв у процесі експлуатації. У зв'язку з цим постає завдання перевірки надійності програмного забезпечення бортового комп'ютера БПС, що дозволить спростити супровід, модернізацію та оптимізацію програмного забезпечення.

Відлагодження та тестування програмного забезпечення БПС є складним завданням. Проблема полягає у складності формування повного набору сигналів датчиків та неможливості створення на реальному БПС позаштатних ситуацій.

Для розробки, тестування та відлагодження програмного забезпечення БПС застосовують спеціальні програмні та апаратні засоби. Вони мають обмежені сфери застосування та функціональні можливості, тому створення засобів комплексного тестування програмного забезпечення бортового комп'ютера БПС з використанням моделей є актуальним завданням.

Аналіз останніх досліджень і публікацій. Насамперед, зупинимося на розгляді сучасних

підходів, які використовуються для перевірки надійності програмного забезпечення при розробці – тестуванні. За ступенем ізольованості коду існує 4 рівні тестування [1]: модульне тестування; інтеграційне тестування; системне тестування; приймальне тестування.

Модульне тестування – рівень, у якому окремі модулі чи компоненти програми (функції, методи чи класи) тестуються незалежно від решти програми. Цей рівень тестування має перевірити, чи працює кожен компонент коректно. Важливим аспектом модульного тестування є ізоляція компонента, що тестується, від інших компонентів програми для впевненості у тому, що виявлення та виправлення помилок відбувається локально в межах даного компонента.

Інтеграційне тестування – рівень, який спрямований на перевірку взаємодії між різними модулями чи компонентами програми. Під час інтеграційного тестування перевіряється, як компоненти взаємодіють один з одним і як вони взаємодіють із зовнішніми системами, якщо такі є. Цей рівень тестування зазвичай слідує за модульним тестуванням, щоб забезпечити правильне функціонування всієї системи.

Системне тестування – рівень, який проводиться на завершальній стадії розробки, коли всі компоненти програми вже інтегровані разом та готові для тестування у реальних умовах перед випуском програми в експлуатацію. Цей рівень забезпечує перевірку всієї системи як єдиного цілого для впевненості у тому, що вона відповідає специфікаціям та вимогам користувачів.

Приймальне тестування – рівень, який спрямований на перевірку відповідності програми вимогам замовника чи кінцевого користувача. Воно зазвичай виконується замовником чи його представниками.

Завдяки тестуванню забезпечується:

- якість коду;
- відповідність заявленим вимогам;
- оптимізація обчислювальних ресурсів;
- безпека та цілісність даних;
- економія часу (пошук помилок може займати багато часу);
- тощо.

Застосування тестування під час розробки, наприклад, прикладного програмного забезпечення для комп'ютера (software) у багатьох випадках достатньо, щоб забезпечити необхідний рівень надійності програми, однак для програмного забезпечення вбудованих систем (firmware) – ні.

Для ефективного та дієвого тестування вбудованого програмного забезпечення використовується велика кількість методів тестування, підходів, інструментів, тому в роботі [2] проведено огляд та виконано систематизацію літературних джерел у цьому напрямку.

Вбудовані системи часто повинні працювати у взаємодії із навколишнім середовищем, використовуючи датчики для отримання вхідних даних (температури, тиску тощо) [3] і БПС не є винятком. У роботі [4] основною ідеєю є розробка тестової платформи для механічного моделювання польоту та кращої стабілізації багатороторних БПС за допомогою різних алгоритмів керування зі зворотним зв'язком, включаючи ПІД-регулятори [5, 6]. Даний стенд дозволяє перевіряти і калібрувати параметри моделі та виконувати керування в реальному часі багатороторним БПС. Більш функціональний стенд для навчальних цілей наведено у роботі [7]. Як і в попередній роботі, тестовий стенд являє собою гіроскопічну конструкцію з трьома ступенями свободи, який забезпечує рух по тангажу, крену і рисканню квадрокоптера. Однак, на відміну від попереднього тестового стенда, дозволяє здійснювати поступальний рух, хоча і з обмеженнями (рис. 1).

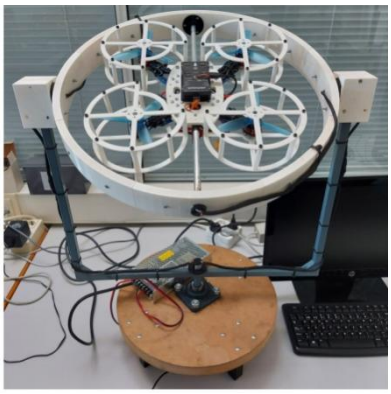

Рис. 1. Експериментальний прототип із 3 ступенями свободи для первірки алгоритмів керування у квадрокоптері

Одним із дієвих методів тестування вбудованих систем є імітаційне моделювання. Імітаційне моделювання (симуляція) – метод дослідження, який використовує моделі, що описують процеси реального світу. Використовуючи такі моделі, можна провести випробування реальної системи, не наражаючи її на небезпеку. До імітаційного моделювання найчастіше вдаються, коли неможливо провести експеримент на реальному об'єкті або необхідно симулювати поведінку реальної системи в часі.

Тест прошивки при симуляції БПС можна виконати у трьох місцях: на хост-комп'ютері, використ-

товуючи компіляцію коду під вихідну платформу; в емуляторі на хост-комп'ютері, наприклад, QEMU; на реальному польотному контролері, використовуючи крос-компіляцію коду для цільового обладнання. Перші два варіанти використовуються при виконанні симуляції SIL (Software in the Loop) [8, 9]. Перевага SIL в тому, що його легко організувати, оскільки не потрібно додаткового обладнання. SIL дозволяє розробникам виконати симуляцію прошивки на ранніх етапах розробки ще до того, як вона буде інтегрована в цільове обладнання. Третій варіант використовується при виконанні симуляції HIL (Hardware in the Loop) [8, 10]. HIL-симуляція включає використання цільового обладнання, що максимально наближає роботу системи до реальних умов.

Постановка завдання. Процес тестування програмного забезпечення БПС може призвести до руйнування апарату внаслідок нештатних ситуацій. Для забезпечення безпечного тестування БПС використовуються випробувальні стенди. Метою статті є організація системи тестування програмного забезпечення БПС із використанням віртуального середовища. Для досягнення поставленої мети в роботі необхідно вирішити наступні завдання: провести порівняльний аналіз існуючих підходів тестування з виявленням переваг і недоліків; організувати взаємозв'язок між бортовим комп'ютером та віртуальним середовищем на персональному комп'ютері; зробити висновки щодо отриманих результатів.

Виклад основного матеріалу. Для забезпечення надійності програмного забезпечення вбудованих систем, як було зазначено вище, використовуються різні методи тестування та інструменти. Основну складність викликає тестування апаратних вузлів вбудованої системи, тому розробляють додаткові апаратні та/або програмні інструменти. На рис.1 показано тестовий стенд, який забезпечує перевірку функціонування алгоритмів у мікроконтролері, роботу датчиків, двигунів тощо. Однак, очевидно, що він обмежує переміщення БПС у просторі. Інший підхід, який дозволяє перевірити повноцінний політ БПС – це використання HIL-симуляції. Політ здійснюється в 3D-середовищі на комп'ютері (наприклад, jMavSim або інший). На рис. 2 показано схему такої системи HIL-симуляції програмного забезпечення автопілота PX4 [11].

Вочевидь, що такий підхід дозволяє протестувати лише роботу мікроконтролера, інші апаратні вузли є віртуальними. Взаємодія цих віртуальних вузлів із бортовим комп'ютером відбувається з використанням протоколу MAVLink [12]. Для забезпечення тестування прошивки мікроконтролера потрібно використати mock-об'єкти. Призначення mock-об'єктів полягає в заміні об'єктів, що тестуються у програмному коді, на відлагоджувальні еквівалентні об'єкти. Створення mock-об'єктів може бути пов'язане з деякими труднощами, наприклад, використання переривань від конкретного інтерфейсу зв'язку, по якому працює реальний датчик. Недолік такого підходу в тому, що прошивка мікроконтролера під час тестування не буде відповідати кінцевій її реалізації.

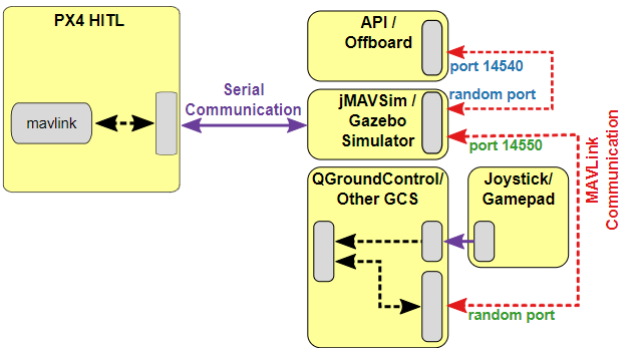

Рис. 2. Структурна схема HIL-симуляції у PX4

Для усунення зазначених недоліків пропонується реалізувати такий апаратний симулятор, який використовує реальну прошивку для мікроконтролера. У цьому випадку бортовий комп'ютер звертатиметься до реальних апаратних вузлів. З іншого боку необхідно використати 3D-середовище як при HIL-симуляції для організації безпечних польотів при тестуванні алгоритмів БПС. Очевидно, що необхіден вузол, який пов'язуватиме бортовий комп'ютер із 3D-середовищем для передачі навігаційної інформації та інформації управління БПС. З вище сказаного випливає, що потрібен симулятор датчиків і актуаторів, у якого програмна модель відповідає апаратним вузлам. Також він повинен з одного боку обмінюватися інформацією з бортовим комп'ютером з використанням реальних інтерфейсів, а з іншого боку обмінюватися інформацією з 3D-середовищем на комп'ютері. Структурна схема такого симулятора показана на рис. 3.

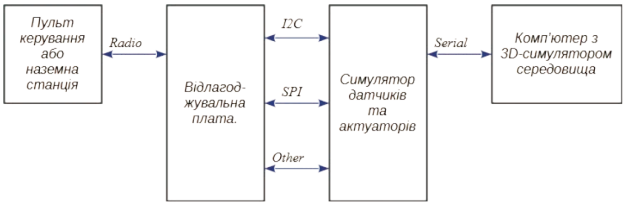

Рис. 3. Структурна схема апаратної симуляції

Для симуляції БПС необхідна відлагоджувальна плата. Найбільш популярні мікроконтролери для бортового комп'ютера STM32F405RGT6 та STM32F722RET6. Нажаль STMicroelectronics не випускає відлагоджувальну плату на базі мікроконтролера STM32F405RGT6, тому можливі 2 варіанти виходу з цього положення: скористатися відлагоджувальною платою стороннього виробника, наприклад, Core405R від Waveshare; використати відлагоджувальну плату NUCLEO-64 від STMicroelectronics, замінивши мікроконтролер на STM32F405RGT6 (це можливо, тому що у STMicroelectronics мікроконтролери з однаковим корпусом мають однакове розташування виводів). Відлагоджувальну плату NUCLEO-F722ZE можна використати для симуляції бортового комп'ютера на базі мікроконтролера STM32F722RET6. Симулятор датчиків та актуаторів (СДА) може бути виконаний на будь-якій відлагоджувальній платі, що має по три інтерфейси I2C та SPI (кількість інтерфейсів обумовлена їх кількістю у вище згаданих мікроконтролерах). В нашому випадку використовувалась NUCLEO-F767ZI. Налаштування СДА для кожного датчика виконується аналогічно його прототипу, а регістри даних заповнюються інформацією, яка поступає від 3D-середовища.

Для перевірки відповідності реалізації програмної моделі СДА датчика та оригінального датчика було застосовано наступну структурну схему (рис. 4).

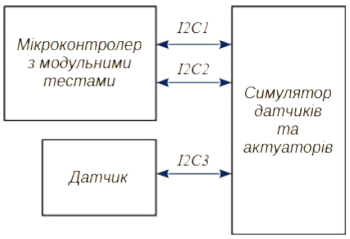

Рис. 4. Структурна схема перевірки ідентичності датчика та симулятора

Через інтерфейс I2C1 надходить інформація від СДА датчика, а через інтерфейс I2C2 надходить інформація від оригінального датчика. Сам датчик підключається до I2C3. При запиті на отримання інформації з регістру даних оригінального датчика вона також записуються в регістр даних СДА датчика. Якщо налаштування СДА датчика та оригінального датчика збігаються, то результати запитів будуть повністю збігатися. На даний час реалізовано лише один датчик СДА – MPU-6500.

Для конфігурування СДА розроблено просту утиліту мовою C++ з використанням фреймворку QT (рис. 5). Для кожного каналу налаштовується модель датчика, інтерфейс та варіативна частина адреси.

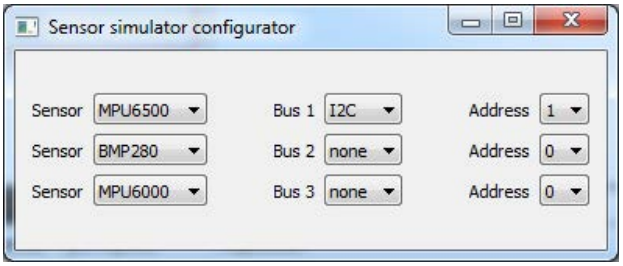

Рис. 5. Зовнішній вигляд конфігуратора симулятора датчиків та актуаторів

Висновки. У статті проведено розгляд організації тестування програмного забезпечення бортового комп'ютера безпілотного повітряного судна. Для забезпечення надійності програмного забезпечення вбудованих систем використовуються різні методи тестування та інструменти. Основну складність викликає тестування апаратних вузлів вбудованої системи, тому розробляють додаткові апаратні та/або програмні інструменти.

Для виконання поставленої задачі було проведено порівняльний аналіз існуючих підходів тестування з виявленням переваг і недоліків. Аналіз показав, що розроблюють як механічні випробувальні стенди, так і використовують імітаційне моделювання (симуляцію) у 3D-середовищі: SIL та HIL. Механічний випробувальний стенд має велику вагу та обмежує переміщення БПС у просторі. Існуючі симулятори SIL та HIL не потребують додаткового обладнання, однак вони не враховують реальну роботу апаратних вузлів.

Запропоновано удосконалений варіант HIL-сімуляції. Основу даного варіанту складає СДА, який забезпечує роботу оригінальної прошивки БПС та дозволяє організувати взаємозв'язок бортового комп'ютера та персонального комп'ютера. СДА було виконано на базі відлагоджувальної плати NUCLEO-F767ZI. Розроблено для конфігурування СДА просту утиліту мовою C++ з використанням фреймворку QT.

Список літератури:

1. Aniche M. Effective Software Testing: A Developer's Guide. Manning Publications Co., 2022. 328 p.
2. Garousi V., Felderer M., Karapicak CM., Yilmaz U. Testing embedded software: A survey of the literature. *Information and Software Technology*. 2018. Vol. 104. P. 14–45. DOI: 10.1016/j.infsof.2018.06.016.

3. Banerjee A., Chattopadhyay S., Roychoudhury A. On testing embedded software. *Advances in Computers*. Elsevier, 2016. Vol. 101. P. 121–153. DOI: 10.1016/bs.adcom.2015.11.005.

4. Hancer M., Bitirgen R., Bayezit I. Designing 3-DOF Hardware-In-The-Loop Test Platform Controlling Multirotor Vehicles. *IFAC-PapersOnLine*. 2018. Vol. 51, № 4. P. 119–124.

5. Petrosian R.V., Pilkevych I.A., Petrosian A.R. Algorithm for optimizing a PID controller model based on a digital filter using a genetic algorithm. *Proceedings of the 3rd Edge Computing Workshop*. 2023. Vol. 3374. P. 97–111. URL: https://ceur-ws.org/Vol-3374/paper07.pdf.

6. Research on improved sparrow search algorithm for PID controller parameter optimization / Zhang M. et al. *Bulletin of the Polish Academy of Sciences Technical Sciences*. P. e147344.

7. Quadcopters Testing Platform for Educational Environments / U. Veyna et al. *Sensors*. 2021. Vol. 21, № 12. P. 4134. DOI: 10.3390/s21124134.

8. Coopmans C., Podhradský M., Hoffer N. Software-and hardware-in-the-loop verification of flight dynamics model and flight control simulation of a fixed-wing unmanned aerial vehicle. *2015 Workshop on Research, Education and Development of Unmanned Aerial Systems (RED-UAS), November 23–25, 2015*. Mexico, 2015. P. 115–122.

9. Nguyen K.D., Ha C., Jang J.T. Development of a new hybrid drone and Software-in-the-Loop simulation using PX4 code. *Intelligent Computing Theories and Application: 14th International Conference, ICIC 2018, Wuhan, China, August 15–18, 2018, Proceedings, Part I.* Springer, 2018. P. 84–93. DOI: 10.1007/978-3-319-95930-6_9.

10. Nguyen K.D., Ha C. Development of Hardware-in-the-Loop simulation based on Gazebo and Pixhawk for unmanned aerial vehicles. *International Journal of Aeronautical and Space Sciences*. 2018. JASS 19. P. 238–249. DOI: 10.1007/s42405-018-0012-8.

11. Hardware in the Loop Simulation (HITL). *PX4 Autopilot User Guide.* URL: https://docs.px4.io/main/en/simulation/hitl.html (date of access: 07.11.2023).

12. Protocol Overview. *MAVLink Developer Guide*. URL: https://mavlink.io/en/about/overview.html (date of access: 07.11.2023).

Petrosian A.R. ORGANIZATION OF TESTING THE SOFTWARE OF THE ON-BOARD COMPUTER OF AN UNMANNED AERIAL VEHICLE

Recently, unmanned aerial vehicles have demonstrated significant technological growth, which is of interest not only to ordinary citizens but also to military, industrial, and civilian sectors.

Designing and developing software for controlling the orientation of an unmanned aerial vehicle in space takes a long time, including time for debugging, testing, and flight tests. Errors made during software development can cause emergencies or other unforeseen failures during operation, which can lead to the destruction of the vehicle and harm to people and the environment. Debugging and testing the software of an unmanned aerial vehicle is a complex task, so special software and hardware tools are used.

The latest studies of approaches to software testing of embedded systems are analyzed with the identification of advantages and disadvantages. The analysis showed that they are developing both mechanical test benches and using simulation modeling (simulation) in a 3D environment: SIL and HIL. The mechanical test bench allows you to test and calibrate model parameters and perform real-time control of a multi-rotor unmanned aerial vehicle, as well as provide pitch, roll and yaw motion. SIL simulation makes it easy to organize testing because it does not require additional equipment. therefore, it can be used in the early stages of software development before it is integrated into the target hardware. HIL simulation includes the use of target equipment, which brings the system's operation as close as possible to real conditions.

The importance is substantiated and an improved version of HIL simulation is proposed. The basis of this option is a simulator of sensors and actuators, which ensures the operation of the original firmware of the unmanned aerial vehicle and allows you to organize the relationship between the on-board computer and the personal computer. To test the idea, the system was implemented in practice. A configuration utility for the simulator of sensors and actuators has also been developed.

Key words: *UAV, unmanned aerial vehicle, on-board computer, software testing, HIL simulation, 3D environment, sensor and actuator simulator.*