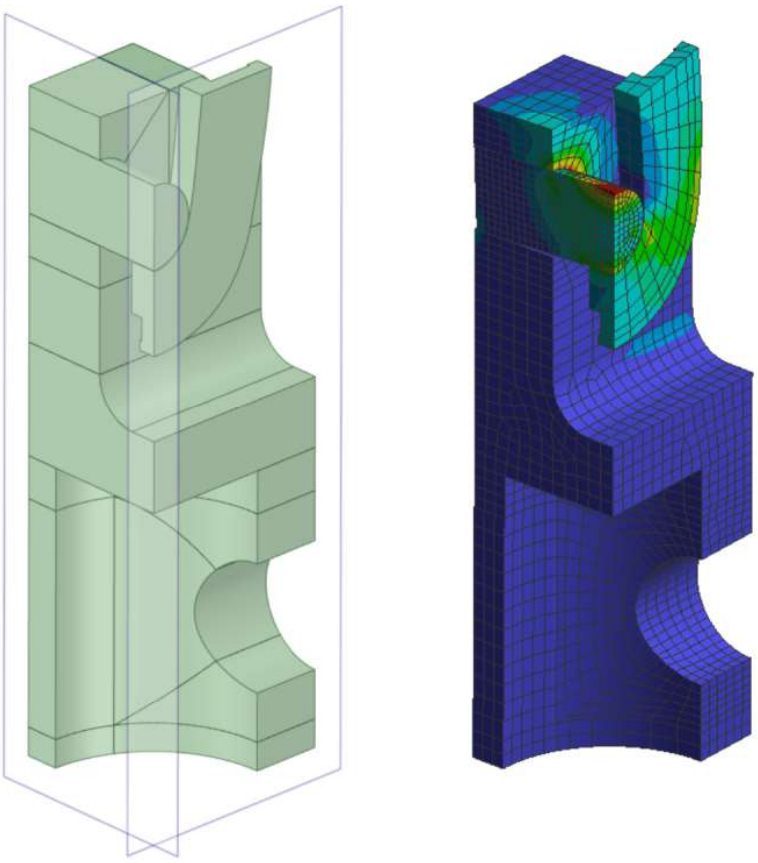
KATEDRA MECHANIKI
I INŻYNIERII OBLICZENIOWEJ
WYDZIAŁ MECHANICZNY TECHNOLOGICZNY POLITECHNIKA ŚLĄSKA

Studencka Konferencja Naukowa

METODY KOMPUTEROWE 2020



Gliwice 2020

Katedra Mechaniki i Inżynierii Obliczeniowej

Wydział Mechaniczny Technologiczny

Politechnika Śląska

Studencka Konferencja Naukowa

„METODY KOMPUTEROWE – 2020”

Gliwice, wrzesień 2020 r.

Katedra Mechaniki i Inżynierii Obliczeniowej
Wydział Mechaniczny Technologiczny
Politechnika Śląska
ul. Konarskiego 18A, 44-100 Gliwice
tel.: 32 237 12 04, fax: 32 237 12 82

Wydanie zeszytów naukowych zostało sfinansowane przez MESco Sp. z o. o. w Bytomiu.
Rysunek na okładce wykonała inż. Barbara Ciszyńska, Autorka artykułu na stronie 13.
ISBN 978-83-951185-1-7
Artykuły opublikowano na podstawie oryginałów dostarczonych przez Autorów.
Druk i oprawę wykonano w Wydawnictwie Politechniki Śląskiej.
Nakład 100 egz. Druk ukończono we wrześniu 2020 r.

Wstęp

Zeszyt naukowy zawiera 42 artykuły prezentowane na czternastej Studenckiej Konferencji Naukowej „Metody Komputerowe”, odbywającej się 24 września 2020 roku na Wydziale Mechanicznym Technologicznym Politechniki Śląskiej w Gliwicach. Konferencję zorganizowali studenci i pracownicy Katedry Mechaniki i Inżynierii Obliczeniowej Politechniki Śląskiej. Publikacje dotyczą zastosowania metod komputerowych w różnych dziedzinach techniki, takich jak:

- wspomaganie komputerowe prac inżynierskich,
- wytrzymałość materiałów,
- biomechanika,
- hydromechanika,
- termodynamika,
- robotyka,
- informatyka,
- optymalizacja,
- badania doświadczalne.

Dziękuję studentom za przygotowanie artykułów i prezentacji na konferencję, Komitetowi Naukowemu za troskę o poziom naukowy prac, Komitetowi Redakcyjnemu za przygotowanie zeszytu naukowego do druku i wersji elektronicznej materiałów konferencyjnych, a Komitetowi Organizacyjnemu za przygotowanie obrad konferencji.

Szczególne podziękowania za wsparcie finansowe organizacji konferencji składam przedstawicielom firmy MESco Sp. z o. o.

Duża liczba zgłoszonych artykułów świadczy o znacznej aktywności naukowej studentów i potrzebie organizacji tego rodzaju konferencji. Życzę studentom owocnych dyskusji w czasie konferencji. Mam nadzieję, że udział w niej będzie inspiracją do dalszych badań naukowych.

Opiekun Naukowy Studenckiego Koła Naukowego
„Metod Komputerowych”

Prof. dr hab. inż. Piotr Fedeliński

Gliwice, wrzesień 2020 r.

APLIKACJA OPERATORA ROBOTA EKSPLORACYJNEGO

inż. MARCIN NAGI
Automatyka i Robotyka, semestr I, 2 stopień
Opiekunowie naukowi: dr hab. inż. Piotr Przystałka, Prof. PŚ
dr inż. Wawrzyniec Panfil

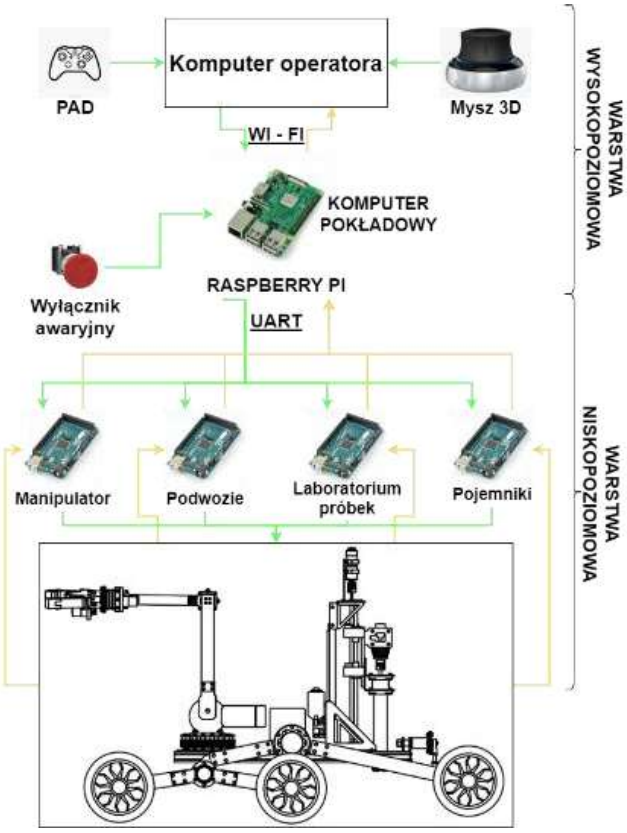
Streszczenie. Projekt polegał na zaprojektowaniu oraz implementacji intuicyjnej aplikacji operatorskiej robota eksploracyjnego - łazika marsjańskiego *Phoenix II*. Projekt został zrealizowany w ramach działań studenckiego koła naukowego *Zastosowania metod sztucznej inteligencji AI – METH*. Zakres prac obejmował: przygotowanie makiet graficznych interfejsu użytkownika, dostosowanie koncepcji oprogramowania do istniejącego systemu komunikacji, wdrożenie oprogramowania i przeprowadzenie testów weryfikacyjnych.

OPERATOR'S CONTROL SOFTWARE FOR THE EXPLORATION ROBOT

Abstract. The purpose of the project was to design and implement an intuitive operator's control software for the exploration robot – Martian rover Phoenix II. The project was carried out as a part of activities of the interfaculty student research group "Applications of methods of artificial intelligence AI – METH". The scope of work included: preparation of graphic user interface mockups, adaptation of the software concept to the existing communication system, implementation of the software and carrying out verification tests.

1. Wprowadzenie

Łazik marsjański *Phoenix II* składa się z 4 głównych modułów: podwozia, manipulatora 5DOF wraz z chwytakiem, urządzenia do pobierania próbek gleby wraz z pojemnikami i systemu bezpieczeństwa. Schemat sterowania robota eksploracyjnego (Rys. 1) zawiera główną koncepcję komunikacji między modułami robota. Polecenia wydawane przez operatora zostają wysyłane do komputera pokładowego, a z niego trafiają do mikrokontrolera dla każdego z modułu, aby ostatecznie aktuatory modułu wykonały zadaną czynność. Moduły manipulatora, podwozia, laboratorium próbek oraz pojemników sterowany są za pomocą Arduino MEGA, natomiast system bezpieczeństwa to osobny program uruchomiony na Raspberry Pi, gdzie wykorzystany został moduł wejść - wyjść, w celu wyzwolenia hard stopu jak i reakcji na wciśnięcie czerwonego przycisku wyłącznika awaryjnego. Kolorem zielonym zaznaczono przepływ poleceń wydawanych przez operatora, natomiast kolor pomarańczowy ukazuje przepływ informacji zwrotnej.



Rys. 1. System sterowania robota eksploracyjnego
Fig. 1. Control scheme of exploration robot

2. Założenia

Aplikacja operatora powinna pozwalać na intuicyjne oraz przystępne sterowanie robotem. W celu zabezpieczenia robota przed niespodziewanym zaprzestaniem działania któregoś z modułów, zostały utworzone pojedyncze aplikacje do sterowania każdą częścią z osobna. Sterowanie robotem odbywa się z jednego komputera operatora. Wysyłanie oraz odbieranie informacji zostało zaimplementowane dzięki wykorzystaniu środowiska *Robot Operating System.* Wstępne szkice graficznego interfejsu użytkownika utworzone zostały przy pomocy programu *Balsamiq Mockups 3.* Do zaprojektowania docelowych okienek posłużył *QtDesigner* w wersji 5, zawierający podstawowe widżety. Oprogramowanie zostało napisane w języku *Python 3* z wykorzystaniem biblioteki *PyQt5* (nakładka na bibliotekę *Qt*). Zatem postanowiono, aby aplikacja operatora składała się z pomniejszych podaplikacji dla każdego z modułów robota. Taka decyzja została podjęta z uwagi na możliwość nieprzewidzianego zaprzestania działania okienka i utraty kontroli nad całym robotem. Rozproszenie aplikacji w przypadku niespodziewanego wyłączenia jednego okienka dla danego modułu zapewnia dalszą kontrolę nad pozostałą częścią robota. Każde okienko w zależności od wymagań charakteryzuje się innymi funkcjonalnościami.

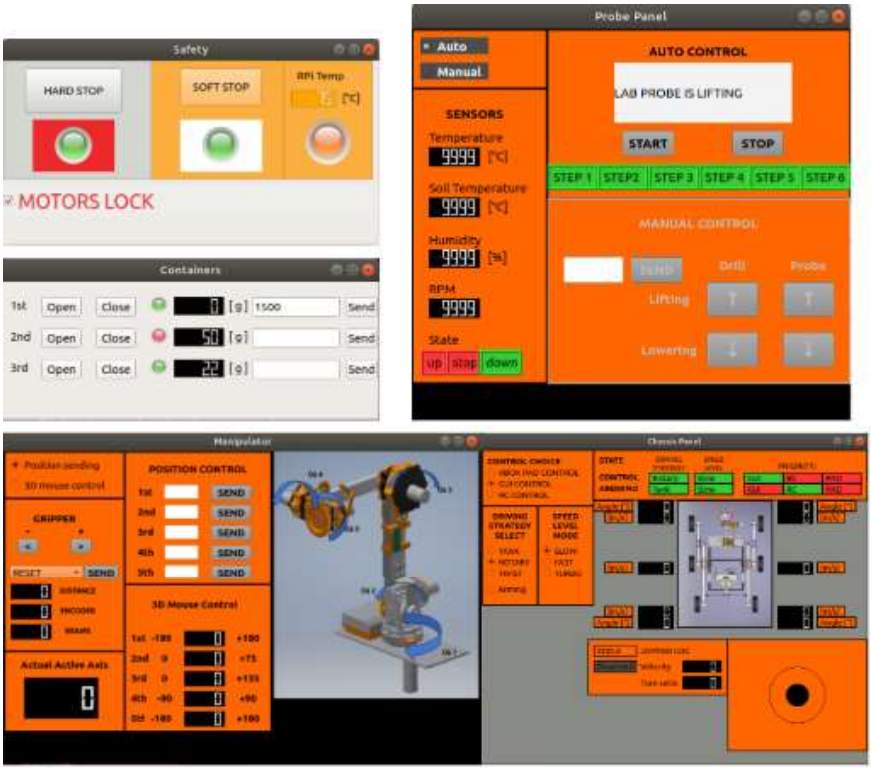
3. Opracowanie projektu aplikacji

Podczas rozwoju oprogramowania dla różnych zastosowań, w celu rozwiązania często występujących problemów z zakresu architektury oprogramowania, stosuje się wzorce architektoniczne struktury aplikacji. Utworzone oprogramowanie panelu operatorskiego

powstało w oparciu o wzorzec *Model-View-Controller* (MVC). Zastosowanie środowiska ROS do opracowania aplikacji należało do najbardziej istotnych etapów powstania oprogramowania, ponieważ oferuje ono sprawdzone, wydajne i łatwe do wdrożenia mechanizmy komunikacji w aplikacjach rozproszonych [1]. Wykorzystując wzorzec MVC dodano kolejny czynnik wpływający na obieg danych w aplikacji [2]. Dla prawidłowego działania i łączenia się środowiska ROS z właściwymi tematami wiadomości każde z okienek stanowi oddzielny węzeł. Węzeł ten jest zarówno nadawcą jak i odbiorcą [3]. Jest to możliwe dzięki wykorzystaniu największej zalety tego środowiska, mianowicie rozproszenia sieci. Nie jest wymagane, aby wszystkie dane musiały przejść przez *roscore*, tylko komunikują się na zasadzie modelu komunikacyjnego peer-to-peer. Nakładka *PyQt5* dzięki oferowanym funkcjom sygnałów i slotów bardzo prosto pozwala połączyć ze sobą nadawcę i odbiorcę z kontrolerem.

4. Implementacja oprogramowania

W ramach projektu opracowano następujące aplikacje (Rys. 2): aplikacja podwozia, manipulatora, laboratorium próbek wraz z pojemnikami oraz aplikacja systemu bezpieczeństwa. Okna zawierają widżety wizualizujące poszczególne stany robota, wykonywane czynności, a także odczyty z czujników, takich jak potencjometry czy czujniki Halla itp. Ważnym zaprojektowanym widżetem był wirtualny joystick używany do sterowania podwoziem, który działał w ten sam sposób jak kontroler XBOX. Zmieniające się kolory podświetlenia etykiet i wskaźników lamp pozwoliły na płynną interpretację zdarzeń przez operatora.



Rys. 2. Okienka utworzonej aplikacji operatora
Fig. 2. Windows of the operator's control panel application

5. Weryfikacja

Ważnym etapem w rozwoju oprogramowania jest weryfikacja poprawności jego działania, która stanowi zwieńczenie wszystkich prac. W tym celu przeprowadzono szereg testów w celu ustalenia, czy przyjęte założenia zostały spełnione lub nie. Wymaga to zastosowania odpowiednich metodologii testów i przygotowania scenariuszy weryfikacji. Dla każdego interfejsu użytkownika przeprowadzono testy jednostkowe (przykładowe podano w Tab. 1) na podstawie przygotowanej listy kontrolnej zawierającej scenariusz testowy i oczekiwany wynik dla wszystkich widżetów i funkcjonalności. Drugim etapem było przeprowadzenie ogólnych testów w celu sprawdzenia opóźnień aplikacji, działania myszy 3D i niezawodności aplikacji. Zarówno testy jednostkowe, jak i test ogólne nie wykazały błędów. Opóźnienia nie były widoczne, a aplikacja działa niezawodnie.

Tabela 1. Przykładowe testy jednostkowe
Table 1. Example of unit tests

L.p.	Opis	Oczekiwany rezultat	Wynik
1.	Kliknięcie przycisku *Position sending*	Wysyłanie wartości 2 typu UInt8 na topic *manipulator/controlChoice*. Włączenie ramki *POSITION CONTROL.*	Pozytywny
2.	Publikowanie wartości 0, typu ChassisState na topic *chassis/state - chassisMode.*	Zmiana napisu etykiety Arduino na *None* i koloru tła na szary.	Pozytywny

6. Wnioski

Zadaniem do rozwiązania w projekcie było opracowanie aplikacji operatora, która pozwala sterować robotem w intuicyjny i przystępny sposób. Powstała aplikacja operatora pozwala na realizację założonych początkowo funkcjonalności. Aplikacja działa szybko i jest niezawodna, ale nie jest całkowicie odporna na uszkodzenia. Przeprowadzone testy wykazały błędy powstałe w trakcie implementacji oprogramowania, natomiast dzięki temu, że aplikacja jest rozproszona na mniejsze moduły, zapobiega to całkowitemu wyłączeniu i utracie kontroli nad robotem.

Literatura

1. Open Source Robotics Foundation, Documentation, http://wiki.ros.org/. (dostęp - 8.06.2020).
2. Jason M. O'Kane, A Gentle Introduction to ROS, 2013.
3. Wikipedia. Pattern model – view – controller. https://en.wikipedia.org/wiki/Model-View-Controller. (dostęp - 8.06.2020).