Pomiary Automatyka Robotyka, ISSN 1427-9126, R. 23, Nr 4/2019, 31–40, DOI: 10.14313/PAR_234/31

Budowa struktury komunikacji: programowanie robotów off-line – MATLAB

Magdalena Muszyńska, Dariusz Szybicki, Paulina Pietruś
Politechnika Rzeszowska im. Ignacego Łukasiewicza, Katedra Mechaniki Stosowanej i Robotyki, al. Powstańców Warszawy 12, 35-959 Rzeszów

Streszczenie: Projektując i budując fabryki zgodnie z ideą Przemysłu 4.0 powszechne jest stosowanie systemów zrobotyzowanych. Wówczas istotne jest monitorowanie oraz analiza parametrów pracy tych systemów, co umożliwia zwiększenie wydajności, bezpieczeństwa pracy, wydajniejsze zarządzanie produkcją. W systemach zrobotyzowanych wymiana danych między urządzeniami odbywa się za pomocą standardów komunikacyjnych, np.: Profibus, DeviceNet, RS-485, OPC. W ramach pracy zaprezentowano własne rozwiązanie na bazie standardu OPC do wymiany danych między systemem zrobotyzowanym a środowiskiem MATLAB. Zaprezentowano aplikację wykorzystującą GUI, gromadzącą i przetwarzającą dane procesowe ze zrobotyzowanego stanowiska wyposażonego w manipulatory firmy ABB.

Słowa kluczowe: serwer OPC, pakiet MATLAB/Simulink, stanowisko zrobotyzowane

1. Wprowadzenie

Firmy zajmujące się produkcją robotów przemysłowych oferują narzędzia do projektowania i planowania w pełni spersonalizowanych zrobotyzowanych stacji. Przykładem środowiska do projektowania, programowania oraz symulowania stacji jest RobotStudio firmy ABB. Pozwala ono na pełną personalizację stanowisk, kompatybilność z systemami CAD, analizę procesów oraz sygnałów podczas działania symulacji.

Wraz z rozwojem idei Czwartej Rewolucji Przemysłowej (in. Przemysł 4.0) [1, 6, 7] wdraża się rozwiązania pozwalające na swobodny przepływ informacji między systemami w przedsiębiorstwie przemysłowym. Coraz częściej wprowadza się automatyzację procesów produkcyjnych, mającą na celu zwiększenie wydajności wytwarzania i możliwość wprowadzania zmian asortymentu [8, 9, 2].

Dla łatwiejszego dostępu do danych, np. dla operatorów maszyn, stosuję się panele operatorskie HMI (ang. *Human Machine Interface*). Są to urządzenia służące do kontrolowania i wymiany danych z innymi urządzeniami, stanowiskami zrobotyzowanymi, realizującymi procesy technologiczne i produkcyjne.

W celu pozyskiwania i przetwarzania danych ze zrobotyzowanych stanowisk można wykorzystać oprogramowanie zewnętrzne – pakiet MATLAB/Simulink. Jest to platforma do programowania, która umożliwia analizowanie danych, opracowywanie algorytmów oraz budowanie aplikacji. Najnowsze wersje programu MATLAB gwarantują komunikację z systemami za pomocą serwerów OPC. Dane pozyskane z serwera można dzięki prostemu modelowi w Simulinku przenieść na panel kontrolny zaprojektowany w GUI pakietu MATLAB. Rozszerzenie *Real-time* pozwala również na monitorowanie systemów w czasie rzeczywistym [10].

Autor korespondujący:
Paulina Pietruś, p.pietrus@prz.edu.pl

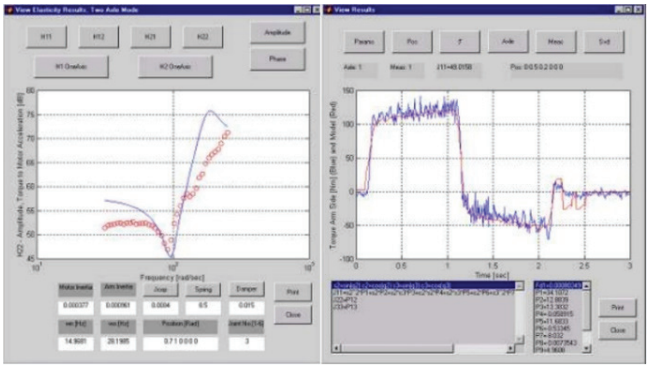
Artykuł recenzowany
nadesłany 14.10.2019 r., przyjęty do druku 26.11.2019 r.

2. Analiza istniejących rozwiązań

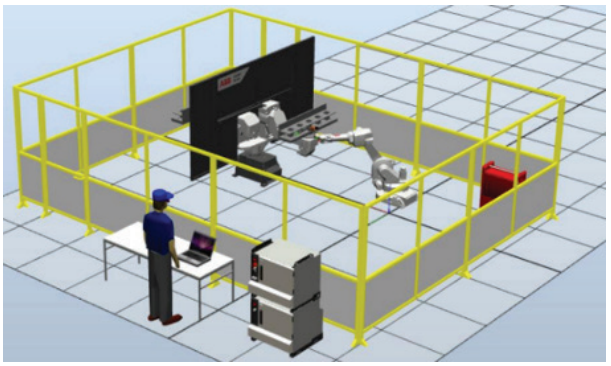
Identyfikacja parametrów jest jednym z ważniejszych etapów podczas projektowania sterowania manipulatorem. Podążanie za założoną ścieżką przez narzędzie robocze jest ściśle powiązane z jakością i dokładnością dynamicznego modelu użytego w kontrolerze czasu rzeczywistego robota. W pracy [5] przedstawiono metodę identyfikacji parametrów sześcioosiowego manipulatora ABB IRB 6400, stosującą aplikację graficzną napisaną w pakiecie MATLAB.

Aplikacja do automatycznej identyfikacji może rozpoznać parametry opisujące bezwładność, siłę ciężkości czy tłumienie w napędach manipulatora. Komponentami systemu użytego do badań symulacyjnych jest robot ABB, kontroler S4C oraz komputer ze środowiskiem MATLAB. Komunikacja między kontrolerem a aplikacją odbywa się przez Ethernet. Sterowanie aplikacją zostało oparte na graficznym interfejsie użytkownika. Dane z symulacji są przedstawiane na wykresach, mogą zostać poddane analizie. Po zakończeniu działania programu, otrzymane wyniki są zapisywane [5]. Na rys. 1 zaprezentowano wykres przedstawiający identyfikację sprężystości, bezwładności napędu i ramienia oraz parametrów związanych z bryłą sztywną robota [5].

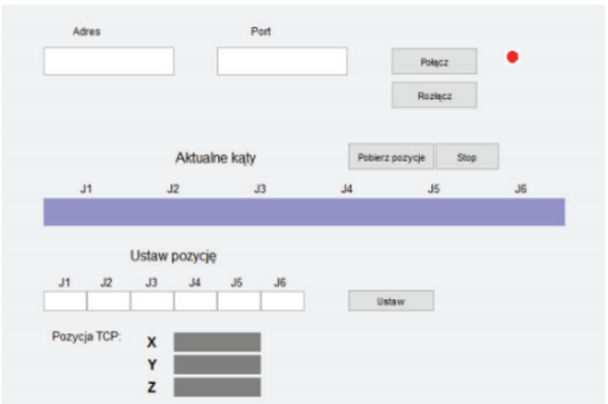
Kolejnym przykładem jest wykorzystanie pakietu MATLAB/Simulink do sterowania pracą manipulatora. W pracy [3] przedstawiono aplikację wykonaną przy użyciu pakietu MATLAB – GUIDE, który umożliwia tworzenie graficznych interfejsów



Rys. 1. Wykresy przedstawiające przykłady identyfikacji parametrów manipulatora [5]
Fig. 1. Graphs showing examples of keypad parameter identification [5]



Rys. 2. Widok stanowiska spawalniczego
Fig. 2. View of the welding station



Rys. 3. Widok aplikacji GUI [3]
Fig. 3. GUI application view [3]

użytkownika. Weryfikację zbudowanej aplikacji przeprowadzono dla stanowiska spawalniczego. Stację wyposażono w manipulator IRB 1600, narzędzie spawalnicze, spawarki, pozycjoner, kontroler IRC 5 oraz komputer (rys. 2).

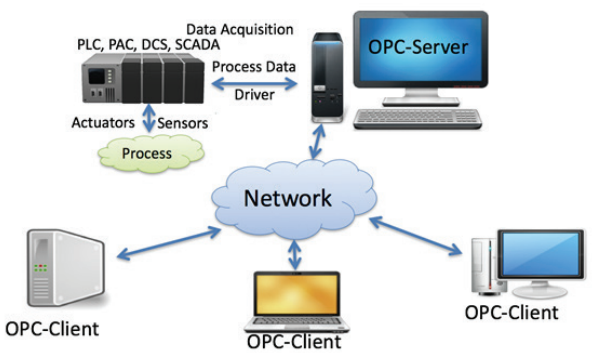
Komunikacja między kontrolerem ICR5 a aplikacją została nawiązana za pomocą protokołu TCP/IP. W zaprezentowanym przykładzie program Hercules spełnia funkcję serwera, z którego pobierane są dane procesowe wykorzystywane przez aplikację graficzną. Na rys. 3 przedstawiono główne okno aplikacji GUI.

Po wpisaniu adresu IP oraz numeru portu wystarczy wcisnąć przycisk *Połącz* w celu nawiązania połączenia ze stacją. Przycisk *Pobierz pozycję* odpowiada za pobieranie danych dotyczących aktualizacji wartości kątowych oraz pozycji TCP (ang. *Tool Center Point*) manipulatora. Dane są odświeżane co 0,1 sekundy. W zaprezentowanym rozwiązaniu możliwe jest zadanie wartości kątowych, które manipulator ma osiągnąć.

3. Komunikacja ze zrobotyzowaną stacją

OPC (ang. *OLE for Proces Control*) jest to otwarty standard komunikacyjny rozwijany przez *OPC Foundation*, umożliwiający przepływ danych między urządzeniami przemysłowymi a stacjami operatorskimi wchodzącymi w skład systemów informatycznych zarządzania przedsiębiorstwem. Standard OPC definiuje sposoby komunikacji między urządzeniami przemysłowymi, przez co pozwala uniezależnić oprogramowanie monitorujące i sterujące od producentów sprzętu. Standard ten wykorzystuje strukturę klient-serwer (rys. 4) [11].

Najczęściej serwerem OPC jest komputer PC z odpowiednim oprogramowaniem zgodnym ze standardem. Rolą serwera jest pobieranie danych procesowych z urządzeń, które następnie są przetwarzane i udostępniane poszczególnym zewnętrznym systemom, czyli klientom OPC. Odebrane dane serwer OPC zapisuje w postaci kolejnych rekordów bazy danych, która staje się źródłem informacji dla klientów i może zostać przetworzona przez ich oprogramowanie. Urządzenie komunikuje się z serwerem za pomocą dedykowanego typu protokołu komunikacyjnego, którym może być Modbus, Profibus, Profinet lub Ethernet. Serwer może nie tylko odbierać dane, ale również przesyłać je do połączonych z nim urządzeń, sterując ich pracą.



Rys. 4. Struktura komunikacji klient-serwer OPC [12]
Fig. 4. OPC client-server communication structure [12]

Rolę klienta OPC pełni oprogramowanie – aplikacja odpowiedzialna za odbieranie danych z serwera lub przesyłanie ich do kolejnych systemów. Przykładami takiego oprogramowania są systemy SCADA, HMI oraz autorskie aplikacje wykonane z wykorzystaniem pakietu MATLAB/Simulnik. W artykule zdecydowano się zastosować specyfikację OPC DA ze względu na to, iż zarówno ABB, jak i MathWorks oferują narzędzia do obsługi i konfiguracji tego typu serwera.

3.1. Konfiguracja ABB OPC IRC5 Serwer DA

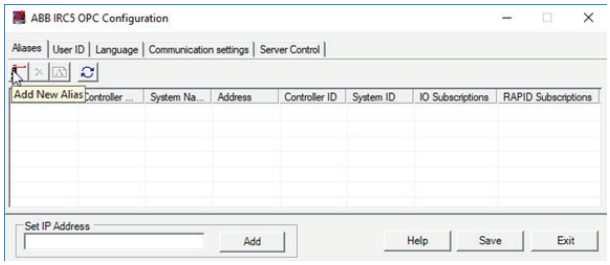
Kontroler IRC5 jest jednostką odpowiedzialną za sterowanie, komunikację oraz zarządzanie pracą zrobotyzowanego stanowiska. ABB *OPC IRC5 Server DA* jest to oprogramowanie, które zostało użyte w zaprezentowanym rozwiązaniu w roli serwera OPC. Umożliwia ono komunikację kontrolera IRC5 z zewnętrznym oprogramowaniem.

Komunikacja między kontrolerem a serwerem następuje przy użyciu standardu TCP/IP. Rodzina protokołów TCP zapewnia prawidłowe przesyłanie danych w odpowiedniej kolejności, stosując właściwości IP do nadawania i odbioru. Każdy segment, czyli jednostka czasu, którą wykorzystują przy komunikacji moduły TCP, ma sumę kontrolną, która po stronie odbiorcy podlega weryfikacji. Pozwala to na sprawdzenie poprawności odebranych danych. TCI/IP umożliwia komunikację wielu aplikacji w tym samym czasie, kierowanie przekazywaniem danych, rozpoznawanie jednostki komputerowej po indywidualnych adresach IP oraz portach TCP [4].

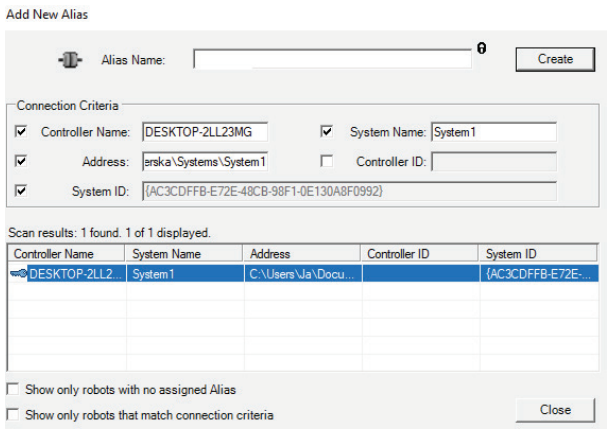
Konfiguracja sposobu komunikacji kontrolera sprowadza się do wybrania interfejsu, z którego będzie korzystał, nadania mu adresu IP oraz maski podsieci. W celu skonfigurowania serwera należy uruchomić aplikację ABB *OPC IRC5 Configuration* (rys. 5).

Definiowanie nowego aliasu odbywa się przez wybór polecenia *Add* (rys. 6) *New Alias*, a następnie określenie połączenia. Utworzenie nowego połączenia wymaga wprowadzenia określonych danych: *Alias Name*, *Controller Name*, *Address*, *System ID*, *System Name*. Wszystkie dane mogą zostać automatycznie uzupełnione po wybraniu polecenia *Scan*. W tym przypadku wyświetla się lista dostępnych kontrolerów. Wybranie jednego skutkuje automatycznym uzupełnieniem wszystkich pól. Po utworzeniu aliasu dane dotyczące *IO Subscriptions* i *RAPID Subscriptions* zostaną przypisane na podstawie wybranego kontrolera. *IO Subcriptions* informuje o liczbie wykorzystanych, jak i dostępnych maksymalnie sygnałów wejścia/wyjścia, które zostały utworzone w kontrolerze, natomiast *RAPID Subscription* odpowiada za liczbę zadeklarowanych zmiennych.

Po skonfigurowaniu oraz utworzeniu nowego aliasu należy w polu *Set IP Address* dodać adres IP kontrolera (rys. 6), z którym będzie następowała komunikacja, przez standard TCP/IP. Ostatnim etapem jest restart serwera, polega on na wciśnięciu po sobie przycisków *Stop* oraz *Start* oraz zapisaniu stanu. Tak przeprowadzona konfiguracja gwarantuje komunikację z wybranym, w tym przypadku wirtualnym kontrolerem IRC5.



Rys. 5. Okno startowe narzędzi konfiguracyjnych serwera
Fig. 5. Start window of server configuration tools



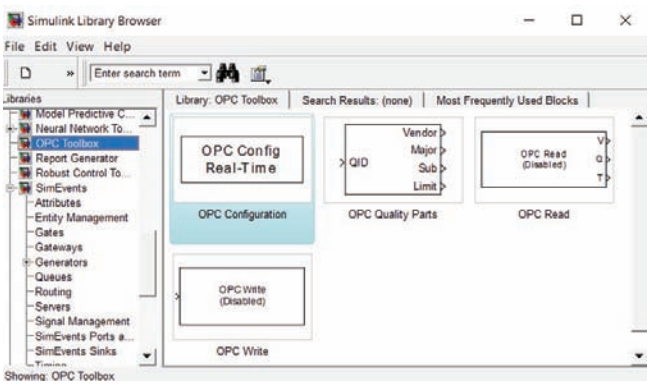
Rys. 6. Definiowanie aliasu
Fig. 6. Defining an alias

3.2. Komunikacja z serwerem OPC za pomocą MATLAB OPC Toolbox

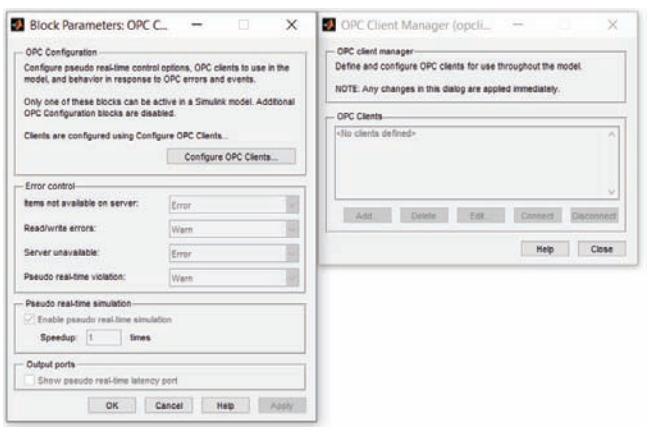
Nowsze wersje pakietu MATLAB/Simulink mają dodatek *OPC Toolbox*, który umożliwia komunikację z wykorzystaniem standardu OPC. Można też korzystać zarówno ze standardu DA, jak i HDA. Pozwala na odczytywanie, zapisywanie informacji od serwera i wysyłanie danych do serwera z różnych urządzeń, systemów kontrolnych/nadzorczych, zbierających dane czy programowalnych kontrolerów. *OPC Toolbox* zawiera aplikację *OPC Data Access Explorer*, dzięki której możemy połączyć się z serwerem OPC.

W artykule zdecydowano się zastosować bibliotekę Simulink – *OPC Toolbox*, która umożliwia obsługę *OPC Data Access* (rys. 7). Podstawowymi blokami tej biblioteki są: *OPC Configuration, OPC Read, OPC Write*. W pierwszym bloku konfiguruje się połączenie z serwerem, kontrolę błędów oraz ustawienia związane z działaniem symulacji w czasie rzeczywistym. Następnie wybiera się lub tworzy *OPC Client* (rys. 8).

Kolejny blok *OPC Write* umożliwia konfigurację serwera/klienta oraz danych, jakie będą do niego przesyłane. W tym bloku można wybrać sposób wysyłania danych – synchroniczny, asynchroniczny oraz czas próbkowania. Dzięki temu, iż Simulink pozwala na zapis danych w trakcie działania symulacji, wykorzystując specjalne komendy oraz obiekty typu *RunTime Object*, jest możliwe zbudowanie modelu, który umożliwi działanie aplikacji w czasie rzeczywistym.



Rys. 7. Biblioteka OPC Toolbox w Simulinku
Fig. 7. OPC Toolbox library in Simulink

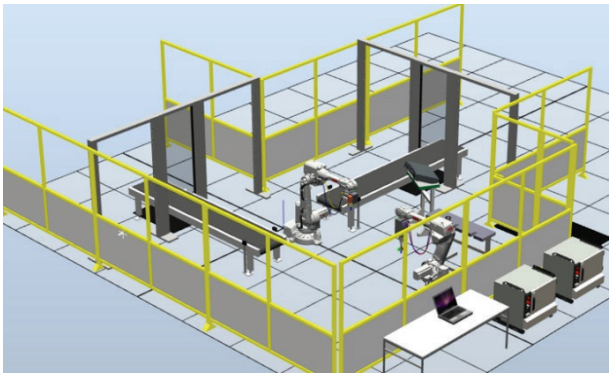


Rys. 8. Ustawienia bloku OPC Configuration
Fig. 8. OPC Configuration block settings

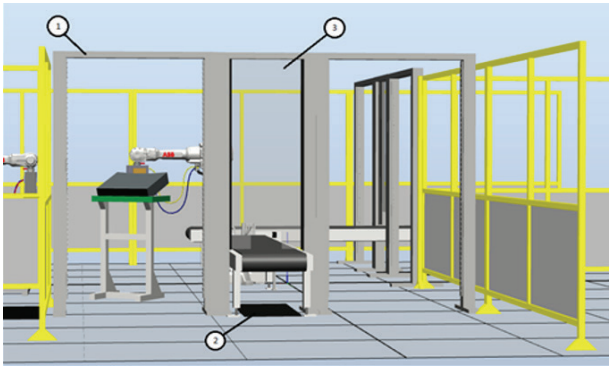
4. Projekt zrobotyzowanej stacji

Weryfikację opracowanej aplikacji zrealizowano na specjalnie zaprojektowanym oraz zbudowanym w środowisku RobotStudio zrobotyzowanym stanowisku. Ma ona postać stacji przeznaczonej do obróbki wstępnej odlewanych ze stali korpusów reduktorów (rys. 9).

Stanowisko zostało wyposażone w manipulatory ABB IRB 2600, kontrolery IRC5, podajniki taśmowe, skaner 3D, stojak na narzędzia, elektromagnes, frezy, maty bezpieczeństwa oraz barierę z tworzywa sztucznego wyposażoną w czujniki nacisku (rys. 10). Za sterowanie stacją odpowiada wirtualny



Rys. 9. Widok zrobotyzowanego stanowiska w środowisku RobotStudio
Fig. 9. View of the robotic station in the RobotStudio environment



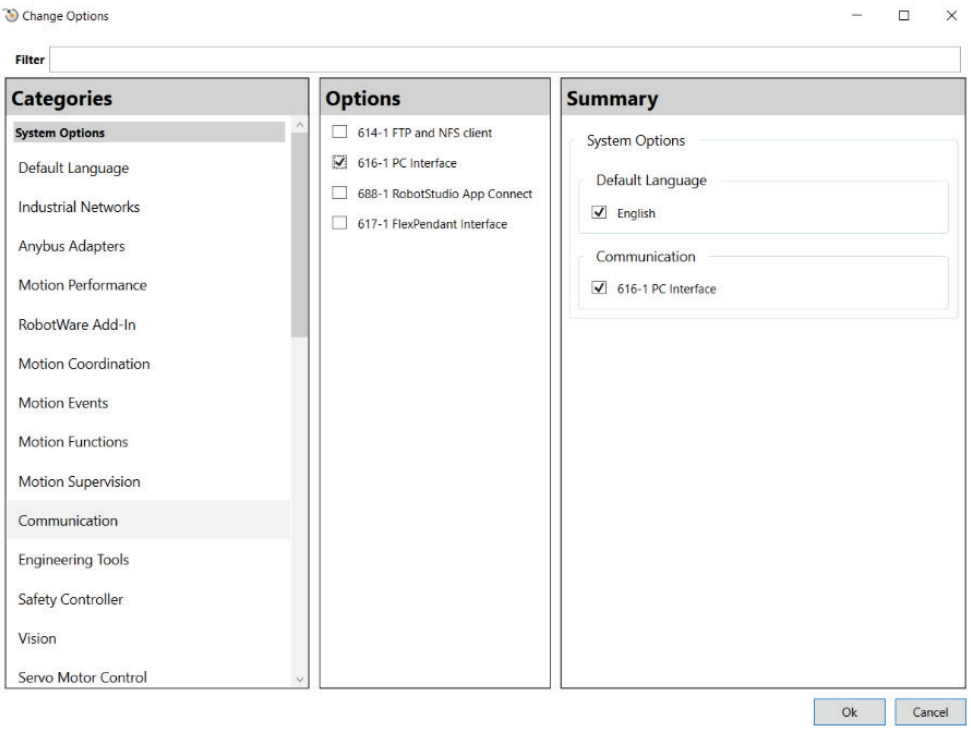
Rys. 10. Bariera laserowa (1), mata bezpieczeństwa (2), bariera z czujnikami nacisku (3)
Fig. 10. Laser barrier (1), safety mat (2), barrier with pressure sensors (3)

kontroler IRC5, który dzięki możliwości generowania sygnałów cyfrowych i analogowych będzie źródłem danych dla aplikacji napisanej w środowisku MATLAB.

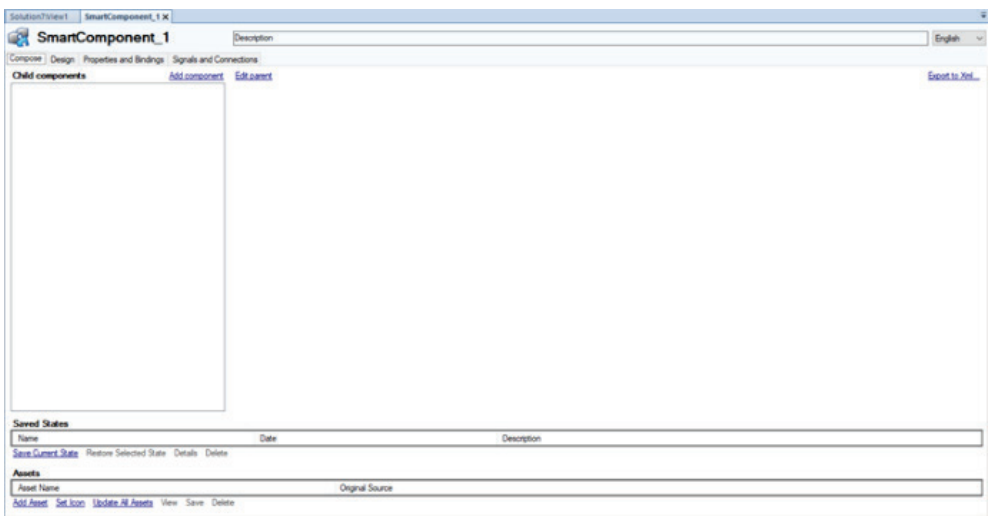
4.1. Budowa systemu stacji

Do poprawnego działania stanowiska, wymagany jest odpowiednio skonfigurowany system robotów. W przypadku rozpatrywanej stacji, w której występuje współpraca dwóch manipulatorów, zaimplementowany musi zostać moduł *MultiMove*. W trakcie tworzenia systemu kreator automatycznie dodaje daną funkcję. Warunkiem, który musi zostać spełniony, aby zrobotyzowana stacja mogła komunikować się z systemami zewnętrznymi, jest dodanie opcji *PC Interface* w trakcie tworzenia systemu robota (rys. 11). Moduł *PC Interface* pozwala na komunikację za pomocą standardu TCP/IP z panelami Flex Pendant czy serwer OPC.

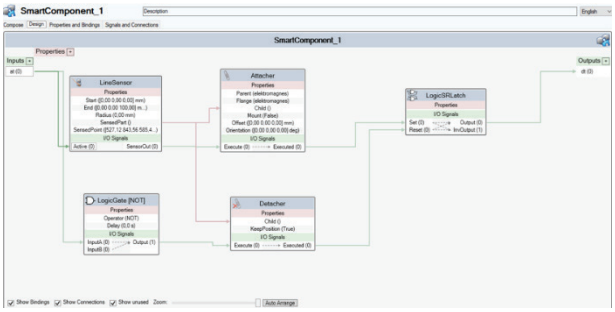
W projekcie zrobotyzowanego stanowiska zostały użyte obiekty typu *Smart Component*. Ułatwiają one tworzenie logiki i algorytmów odpowiadających m.in. za pracę chwytaków oraz podajników. Umożliwiają generowanie oraz usuwanie obiektów w czasie symulacji. Do sterowania nimi wystarczą proste sygnały cyfrowe DI/DO. Aby utworzyć tego typu obiekty należy z głównego paska wybrać



Rys. 11. Dodanie opcji PC Interface
Fig. 11. Adding the PC Interface option



Rys. 12. Główne okno konfiguracji komponentów w środowisku RobotStudio
Fig. 12. The main component configuration window in the RobotStudio environment



Rys. 13. Schemat działania zbudowanego SmartComponentu w środowisku RobotStudio
Fig. 13. The main component configuration window in the RobotStudio environment

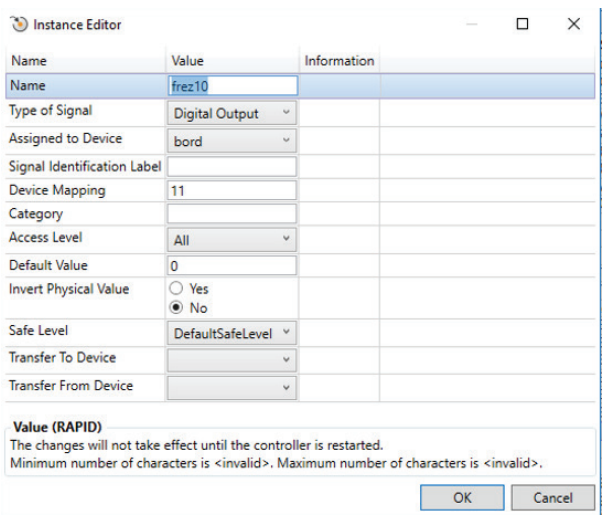
zakładkę *Modelling* → *Smart Component*. Otworzy się wówczas okno odpowiedzialne za dodawanie oraz modyfikację komponentów. W zakładce *Compose* (rys. 12) można dodawać obiekty typu *Attacher*, *Detacher*, *LineSensor*, *Plane Sensor*, *Logic Gate*, *JointMover* oraz wiele innych.

Po wybraniu dodanego elementu wyświetli się okno konfiguracji (rys. 13), gdzie można sprecyzować jego pozycję oraz działanie. Przechodząc do zakładki *Design*, należy rozstawić wstawione elementy oraz utworzyć logikę, która będzie odpowiedzialna za sterowanie.

Przedstawiony przykład (rys. 13) prezentuje komponent odpowiedzialny za sterowanie pracą elektromagnesu. Gdy sygnał wejściowy zmieni wartość na 1, uruchomiony zostanie sensor odpowiedzialny za wykrycie przedmiotu. Po udanym wykryciu sygnał od sensora zostaje przesłany do elementu *Attacher*, który odpowiada za podłączenie wykrytego przez sensor przedmiotu dla wybranego narzędzia. Podłączenie przedmiotu generuję zmianę sygnału wyjściowego komponentu na 1. Po zresetowaniu sygnału wejściowego sensor zostaje wyłączony, uruchomiony zostaje element *Detacher*, który odłącza przenoszony przez elektromagnes przedmiot i ustawia sygnał wyjściowy na 0.

4.2. Program sterujący

Środowisko RobotStudio umożliwia dodawanie, symulację oraz monitorowanie sygnałów wejść/wyjść, które są obsługiwane przez układ wejść/wyjść kontrolera. W zaprojektowanej stacji do kontrolowania obiektów typu *Smart Component*, które odpowiadają za elektromagnes, zmiany narzędzia oraz podajniki taśmowe, wykorzystane zostały sygnały cyfrowe. Do monitorowania pozycji oraz prędkości TCP manipulatorów użyto sygnałów analogowych. Na potrzeby pracy zostało utworzone urządzenie do komunikacji z wykorzystaniem sieci Profinet, o nazwie *bord*. Nowemu sygnałowi należało przypisać bity oraz zmienić jego poziom dostępu na *All* – umożliwiający odczyt oraz zapis informacji (rys. 14). W przypadku sygnałów analogowych należy określić ich dopuszczalne wartości. Po poprawnym skonfigurowaniu sygnału należy zapisać zmiany i zrestartować kontroler. Utworzenie sygnałów cyfrowych oraz analogowych umożliwiło napisanie programu sterującego.



Rys. 14. Okno dodawania nowego sygnału do kontrolera
Fig. 14. Window for adding a new signal to the controller

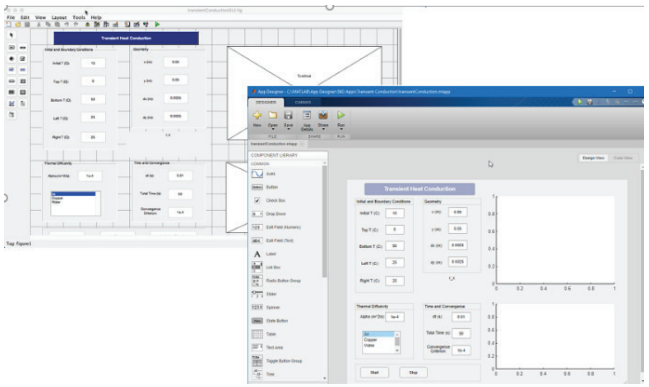
Program sterujący zrobotyzowaną stacją został utworzony w języku programowania RAPID wykorzystując metody programowania w widoku graficznym, jak i programowania tekstowego w edytorze języka RAPID. Do sterowania stacją zostały wykorzystane podstawowe komendy ruchu, oczekiwania czy zmiany wartości sygnału. Użyto również poleceń związanych z wykonywaniem operacji *MultiMove*. Utworzono dwa dodatkowe zadania (ang. *task*) przeznaczone do wykonywania obliczeń. Poniżej przedstawiono kod jednego z nich:

```
MODULE modul1

    VAR pos elek;

    VAR num elekx;

    VAR num eleky;

    VAR num elekz;

    PROC main()

        WHILE startcc0=1 DO

        WaitTime 0.05;

        elek:=CPos(\Tool:=elektromagnes\
        WObj:=Workobject_1);

        elekx := elek.x;

        eleky := elek.y;

        elekz := elek.z;

        SetAO rob1tx,elekx;

        SetAO rob1ty,eleky;

        SetAO rob1tz,elekz;

        ENDWHILE

    ENDPROC

ENDMODULE
```

Procedura zawarta w *modul1* służy do iteracyjnego odczytywania co 0,05 sekundy pozycji TCP manipulatora wyposażonego w elektromagnes. Dzięki zastosowaniu komendy *WaitTime* program nadal pozostaje responsywny w trakcie jego działania. Do zmiennej typu *pos* w pętli przypisywana jest pozycja narzędzia *elektromagnes* dla wybranego elementu *WObj*, w tym przypadku *Workobject*_1. Następnie do zmiennych zostają przypisane poszczególne współrzędne *X*, *Y*, *Z*, które w kolejnym kroku są przypisywane sygnałom analogowym. W sposób analogiczny odczytywane są dane dotyczące pozycji TCP drugiego manipulatora w osobnym tasku, ponieważ każdy task przypisany jest do określonej jednostki mechanicznej. Wykonywanie obliczeń w jednym tasku skutkowałoby poprawnością tylko dla tego manipulatora, do którego jest przypisany dany task.

5. Projekt aplikacji GUI w środowisku MATLAB/Simulink

Pakiet MATLAB w wersji R2016a umożliwia zastosowanie nowego narzędzia App Designer do budowy graficznego interfejsu użytkownika GUI. App Designer ma więcej funkcji oraz obiektów, które można wykorzystać podczas budowy aplikacji. Na rys. 15 porównano budowę głównego okna aplikacji GUIDE oraz App Designer. Interfejs GUI zawiera następujące elementy – menu, paski narzędzi, przyciski oraz suwaki. Dodatek GUI w środowisku MATLAB umożliwia budowę niestandardowych aplikacji.

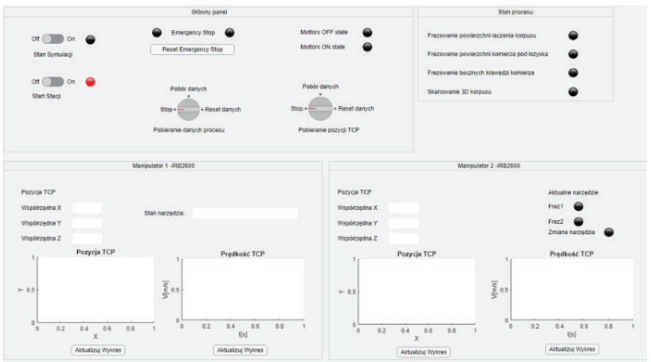


Rys. 15. Porównanie budowania aplikacji w GUIDE oraz w App Desinger [13]
Fig. 15. Comparison of building applications in GUIDE and in App Designer

5.1. Opis zbudowanej aplikacji

W ramach prac zbudowano aplikację GUI z wykorzystaniem narzędzia App Designer. Zadaniem aplikacji jest monitorowanie procesów oraz odczyt określonych parametrów pracy zbudowanej stacji. Graficzny interfejs użytkownika zaprojektowanej aplikacji przedstawiono na rys. 16.

Główny ekran został podzielony na cztery panele. Pierwszy panel – (Główny panel) jest złożony z następujących elementów (rys. 17):



Rys. 16. Główne okno zbudowanego graficznego interfejsu użytkownika
Fig. 16. The main window of the built-in graphical user interface

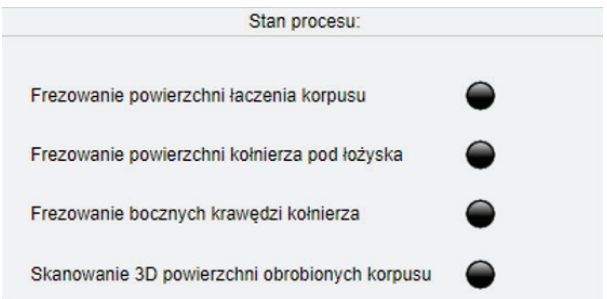
[13]

Rys. 17. Główny panel aplikacji GUI
Fig. 17. The main GUI application panel

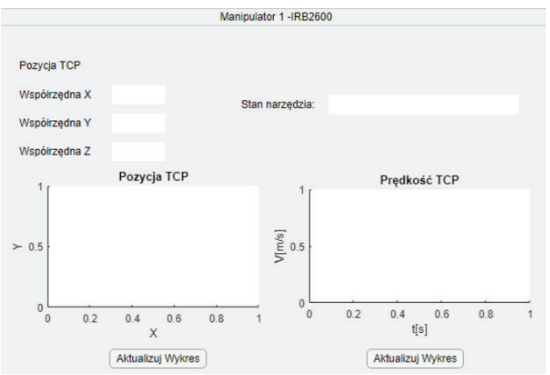
— przełącznik Stan Symulacji odpowiada za włączanie i wyłączanie symulacji w Simulinku, która służy do komunikowania się z serwerem OPC;
— przełącznik Stan Stacji włącza lub wyłącza pracę zrobotyzowanej stacji w środowisku RobotStudio, zmieniając stan sygnału sterującego;
— kontrolka *Emergency Stop* monitoruje sygnał awaryjnego postoju w systemie *RobotWare* zaprojektowanej stacji;
— przycisk *Reset Emergency Stop* odpowiada za przywracanie stacji do normlanego trybu po wystąpieniu awaryjnego postoju;
— kontrolki *Motors OFF state/Motors ON state* pokazują stan napędów manipulatorów stanowiska – gdy są nieaktywne, podświetlenie jest koloru czarnego, natomiast przy uaktywnieniu zmienia się na kolor zielony;
— przełącznik *Pobieranie danych procesu* ma trzy stany: *Start*, *Pobieranie Danych*, *Reset Danych*. Opcja *Pobieranie Danych* uruchamia *timer*, który odświeża komendy związane z pobieraniem danych procesów obróbki z serwera OPC. *Stop* zatrzymuje *timer* natomiast *Reset Danych* zeruje wskazania kontrolek oraz pól powiązanych z timerem;
— przełącznik *Pobieranie pozycji TCP* spełnia tę samą funkcję, co przełącznik *Pobieranie danych procesu* z tą różnicą, iż operuje timerem oraz polami związanymi z wyświetlaniem pozycji TCP manipulatorów. Do przełączników zostały dodane kontrolki w celu lepszej sygnalizacji zmiany stanów.

Panel *Stan Procesu* (rys. 18) zawiera cztery kontrolki odpowiedzialne za monitorowanie procesu obróbki wstępnej odlewanych korpusów:
— Frezowanie powierzchni łączenia korpusu,
— Frezowanie powierzchni kołnierza pod łożyska,
— Frezowanie bocznych krawędzi kołnierza,
— Skanowanie 3D powierzchni obrobionych korpusu.

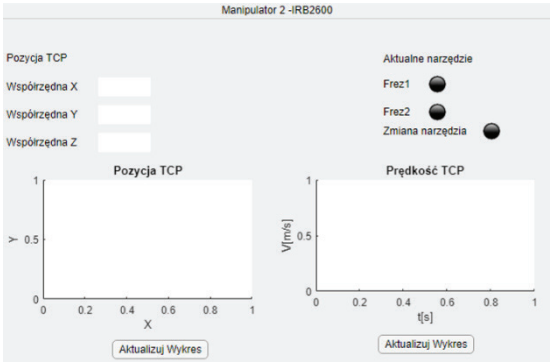
W panelach Manipulator 1 (rys. 19) oraz Manipulator 2 (rys. 20) znajdują się pola, w których wyświetlane będą w czasie rzeczywistym współrzędne X, Y, Z pozycji TCP manipulato-



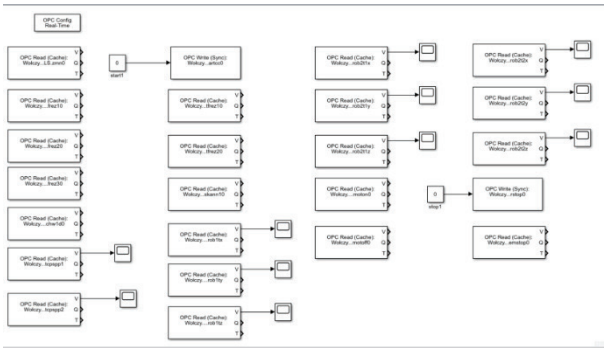
Rys. 18. Widok panelu Stan procesu aplikacji GUI
Fig. 18. View of the GUI Application Process Status panel



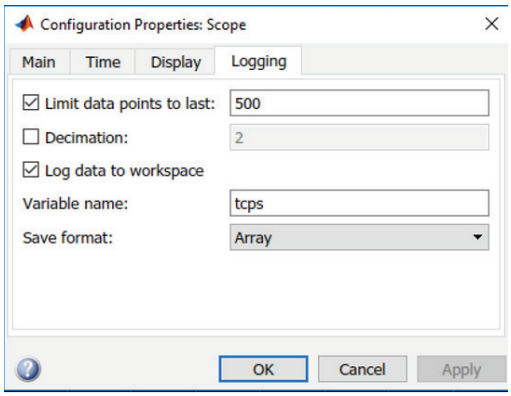
Rys. 19. Panel Manipulator 1 IRB2600
Fig. 19. Panel IRB2600 Manipulator 1



Rys. 20. Panel Manipulator 2 IRB2600
Fig. 20. Panel IRB2600 Manipulator 2



Rys. 21. Model komunikacji w Simulinku
Fig. 21. Model of communication in Simulink



Rys. 22. Okno konfiguracyjne bloku Scope w Simulink
Fig. 22. Scope block configuration window in Simulink

rów z pewnym opóźnieniem. W celu wizualizacji trasy pokonanej przez końcówkę roboczą użyto wykresu 3D z możliwością obracania – Pozycja TCP. W panelach manipulatorów zawarto wykresy odpowiedzialne za przedstawianie zmian prędkości TCP. Czas, który jest zaprezentowany na wykresach, nie odpowiada czasowi symulacji ze środowiska RobotStudio. Związane jest to z częstym odświeżaniem symulacji w Simulink celem odczytu jak największej liczby zmian sygnałów. W obydwu panelach pod wykresami znajdują się przyciski *Aktualizuj Wykres*, których celem jest dodawanie danych do wykresów, gdyż samoczynna aktualizacja tych wykresów znacząco ogranicza częstotliwość odświeżania okna aplikacji. W panelu Manipulator 1 (rys. 19) znajduje się pole wyświetlające stan narzędzia manipulatora. Panel Manipulator 2 (rys. 20) zawiera trzy kontrolki *Frez 1*, *Frez 2* oraz *Zmiana narzędzia*, które wskazują aktualnie używane narzędzie oraz proces wymiany końcówki roboczej.

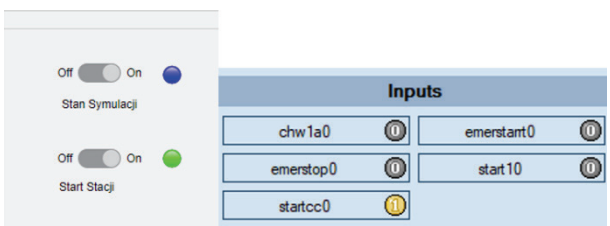
Za komunikację z serwerem IRC5 OPC odpowiada model komunikacji zbudowany w Simulinku (rys. 21). Do jego budowy zastosowano bibliotekę OPC Toolbox.

W modelu dla każdego sygnału przypisano osobny blok odpowiedzialny za odbieranie danych z serwera. Jest to związane ze sposobem odczytywania danych w trakcie działania symulacji, a mianowicie zastosowaniem obiektów typu *Runtime*. Obiekt tego typu przechowuje tylko aktualne dane i zapisuje je w formie wektora. Zmienne, do których zapisują wartości sygnałów są jednoelementowymi tablicami. Dzięki temu można zmniejszyć opóźnienia związane z aktualizowaniem danych. W celu wyświetlania wykresów w aplikacji dane zostają zapisane za pomocą bloków typu *Scope* do głównego obszaru roboczego w postaci tablic (rys. 22). W zakładce *Logging* zastała wybrana opcja *Log data to workspace*, by zapisywać dane do obszaru roboczego, gdzie ich liczba została ograniczona do 500 a po upływie 5 sekund symulacji wykresy będą się resetowały. Celem jest poprawienie czytelności wykresów ze względu na to, iż stacja działa w pętli, więc dane będą aktualizowane po każdym cyklu.

5.2. Symulacja oraz weryfikacja wykonanego oprogramowania

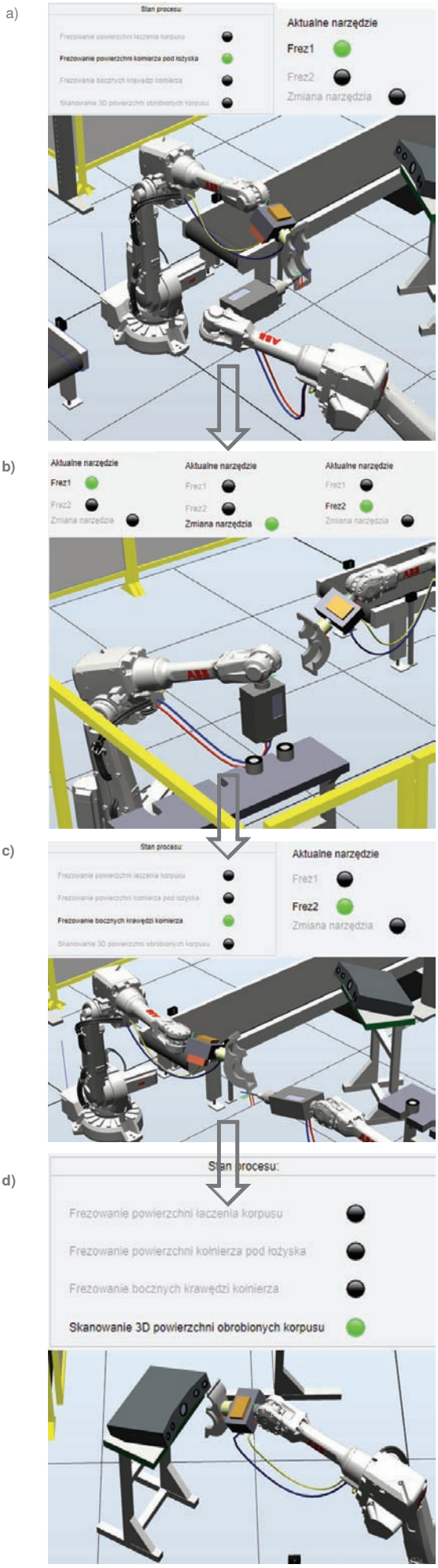
Weryfikacja uzyskanych wyników oraz testowanie napisanego oprogramowania jest jednym z istotnych etapów projektowania zrobotyzowanego stanowiska. Symulacja komputerowa w sposób przybliżony pozwala odtworzyć zachowanie zaprojektowanego modelu oraz umożliwia wykrycie błędów na etapie projektowym. Środowisko RobotStudio umożliwia weryfikację napisanego oprogramowania dla zaprojektowanej stacji zrobotyzowanej. Sygnały generowane podczas symulacji są przesyłane za pośrednictwem serwera OPC do aplikacji napisanej w pakiecie MATLAB/Simulink, która odpowiada za monitorowanie parametrów, procesów.

Przed przystąpieniem do weryfikacji zaprojektowanego oprogramowania należy sprawdzić, czy wirtualny kontroler w RobotStudio oraz serwer OPC mają ten sam numer IP. Aby komunikacja została nawiązana należy otworzyć model w Simulinku, który będzie wymieniał dane z serwerem OPC. Stacja jest tak skonfigurowana, iż czeka na sygnał wejściowy z aplikacji, aby rozpocząć pracę. Po uruchomieniu aplikacji wykonanej w App Designer pojawia się główne okno programu. Symulacja modelu w Simulink zostanie uruchomiona po wybraniu pozycji *On* przy poleceniu *Stan Symulacji* (rys. 23).

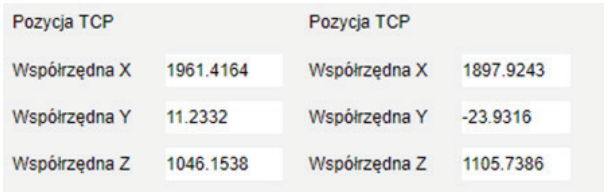


Rys. 23. Weryfikacja połączenia aplikacji ze stacją
Fig. 23. Verification of the application connection to the station

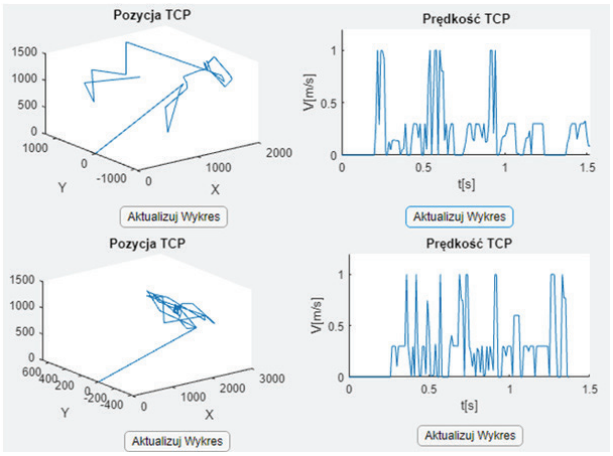
Kolejnym krokiem jest wybór polecenia *On* przy poleceniu *Start Stacji* w celu sprawdzenia, czy dane są przesyłane od aplikacji do stanowiska (rys. 23). Zmiana koloru kontrolki na kolor zielony przy poleceniu *Start Stacji* oznacza, że w RobotStudio zarejestrowano zmianę sygnału startcc0 (rys. 23) na 1, co świadczy o poprawnym nawiązaniu komunikacji. Wybór polecenia *Pobór danych procesu* uruchamia *timer*. Opóźnienie związane z odświeżaniem prezentowanych danych waha się w granicach 0,2–0,3 sekundy, co pozwala na dokładne śledzenie procesu. Po rozpoczęciu pobierania danych kontrolki *Motor ON state*, Frezowanie powierzchni łączenia korpusu i Frez1 zmieniły kolor na zielony. W oknie RobotStudio można obserwować pracę stanowiska, jest wykonywany pierwszy etap obróbki za pomocą frezu oznaczonego Frez1. Obrabiany przedmiot jest pobrany



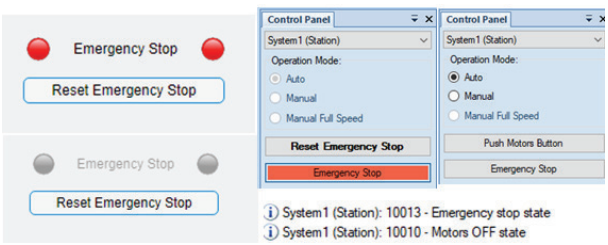
Rys. 24. a) obróbka korpusu, b) zmiana frezów, c) kolejny etap obróbki korpusu, d) skanowanie 3D obrobionego korpusu
Fig. 24. a) machining the body, b) changing the cutters, c) the next stage of machining the body, d) 3D scanning of the machined body



Rys. 25. Pozycje TCP narzędzia w przestrzeni dla manipulatorów 1 oraz 2
Fig 25. TCP tool positions in the space for manipulators 1 and 2



Rys. 26. Wykres pozycji i prędkości końcówki roboczej manipulatora 1 oraz 2
Fig 26. Graph of position and speed of the working tip of manipulators 1 and 2



Rys. 27. Test awaryjnego zatrzymania
Fig. 27. Emergency stop test

przez pierwszy manipulator (rys. 24a) za pomocą elektromagnesu, co jest zgodne ze wskazaniem w etykiecie *Stan narzędzia w aplikacji*. Dalej następują kolejne etapy obróbki oraz zmiany narzędzia (rys. 24).

Kolejnym etapem jest wybór polecenia *Pobieranie pozycji TCP*, który przedstawia współrzędne TCP narzędzia (rys. 25) oraz wykresy odpowiadające za ścieżkę pokonaną przez końcówkę roboczą oraz zmiany prędkości w czasie symulacji w Simulink (rys. 26).

Na końcu przedstawiono weryfikację funkcji alarmowania dla opracowanej aplikacji. Po wybraniu polecenia *Emergency Stop* symulacja w RobotStudio zatrzymuje się, na panelu głównym aplikacji kontrolki, które są odpowiedzialne za monitorowanie sygnału zatrzymania, zmieniają kolor na czerwony (rys. 27).

Po wyłączeniu polecenia *Emergency Stop* na kontrolerze oraz wybraniu polecenia *Reset Emergency Stop* napędy przechodzą w stan *Motors OFF state* (rys. 27). Podczas weryfikacji zauważono, iż w trakcie włączania kolejnych timerów pojawiało się opóźnienie. Timer monitorujący przebieg procesów odczytywał dane z opóźnieniem około 0,2–0,3 sekundy. Podobną tendencję wykazywał timer odpowiedzialny za pobieranie pozycji TCP narzędzia przy samodzielnym działaniu. Przy jednoczesnym włączeniu dwóch timerów opóźnienie zwiększało się do około 0,5–0,8 sekundy. Ma to związek z tym, że do timerów odpowiadających za pobór danych przypisywane są złożone funkcje, które w krótkim czasie są odświeżane. W związku z tym próba generowania nowych wykresów oraz włączenie kolejnego timera powoduje zatrzymanie prezentacji danych na głównym ekranie aplikacji. Duże opóźnienie występowało w chwili przełączania pokręteł oraz wyboru przycisków odpowiedzialnych za aktualizację wykresów. Kolejność włączania obiektów miała wpływ na odświeżanie okna aplikacji oraz aktualizację etykiet czy kontrolek. Mimo występujących opóźnień, aplikacja umożliwia sprawne monitorowanie zachowań zrobotyzowanego stanowiska.

6. Podsumowanie i wnioski

W artykule przedstawiono analizę istniejących rozwiązań. Zaprezentowano aplikacje zbudowane w środowisku MATLAB, których zadaniem jest monitorowanie i sterowanie stacjami oraz identyfikacja parametrów manipulatora. Zaprezentowano własne rozwiązanie złożone z aplikacji GUI gromadzącej i przetwarzającej dane procesowe ze zrobotyzowanego środowiska wyposażonego w manipulatory ABB. Przedstawiono sposób komunikacji zrobotyzowanej stacji z aplikacją zbudowaną w środowisku MATLAB.

Zaprojektowano oraz zbudowano zrobotyzowaną stację w środowisku RobotStudio przeznaczoną do obróbki stalowych korpusów po odlewie. Stanowisko składa się z dwóch manipulatorów firmy ABB, skanera 3D, stojaka na narzędzia, taśmociągów, systemów zabezpieczeń, dedykowanego kontrolera i komputera. Stacja została zaprogramowana przy użyciu języka wysokiego poziomu – RAPID.

W artykule zaprezentowano również projekt aplikacji GUI, która umożliwia monitorowanie stanów obróbki elementów na stanowisku zrobotyzowanym. Zbudowana aplikacja może zostać rozbudowana i udoskonalona. W zależności od złożoności kontrolowanego przez aplikację stanowiska, możliwe jest łatwe dodawanie nowych paneli z kolejnymi funkcjami. W prosty sposób można też rozszerzyć możliwości modelu odpowiadającego za komunikację z serwerem OPC, do którego podłączona jest monitorowana stacja. Ze względu na wystąpienie opóźnień w wersji testowej aplikacji, należy dokonać optymalizacji kodu sterującego aplikacją.

Wykonana aplikacja oraz jej weryfikacja na stanowisku zbudowanym w środowisku RobotStudio wykazały celowość prowadzenia prac w tym zakresie.

Bibliografia

1. Badurek J., *System ERP dla wytwórczości nowej generacji*. „Przedsiębiorstwo we współczesnej gospodarce – teoria i praktyka", Nr 2, 2014, 79–90.
2. Burghardt A., Kurc K., Szybicki D., *Robotic automation of the turbo-propeller engine blade grinding process*. "Applied Mechanics & Materials", Vol. 817, 2016, 206–213, DOI: 10.4028/www.scientific.net/AMM.817.206.
3. Burghardt A., Szybicki D., Pietruś P., *Zastosowanie architektury klient–serwer oraz protokołów TCP/IP do sterowania i monitorowania pracy manipulatorów przemysłowych*. „Modelowanie Inżynierskie", Vol. 36, Nr 67, 2018, 16–22.
4. Giergiel J., Giergiel M., Kurc K., *Sieci komputerowe i bazy danych – Wykłady i laboratoria*, Oficyna Wydawnicza Politechniki Rzeszowskiej, Rzeszów 2010.
5. Iori M., Martello S., Monaci M., *Metaheuristic algorithms for the strip packing problem*. [in:] Optimization and industry: new frontiers. Springer US, 2003. 159–179, DOI: 10.1007/978-1-4613-0233-9_7.
6. Szulewski P., *Koncepcje automatyki przemysłowej w środowisku Industry 4.0*. „Mechanik", Nr 7, 2016, 574–578, DOI: 10.17814/mechanik.2016.7.221.
7. Wittbrodt P., Łapuńka I., *Przemysł 4.0 – Wyzwanie dla współczesnych przedsiębiorstw produkcyjnych*. „Innowacje w Zarządzaniu i Inżynierii Produkcji", 2017, 793–799.
8. Szulewski P., *Urządzenia automatyki przemysłowej w środowisku Industry 4.0*. „Mechanik", R. 89, Nr 8–9, 2016, 926–933, DOI: 10.17814/mechanik.2016.8-9.329.
9. Tjahjono B., Esplugues C., Ares E., Pelaez G., *What does Industry 4.0 mean to Supply Chain?* "Procedia Manufacturing", Vol. 13, 2017, 1175–1182, DOI: 10.1016/j.promfg.2017.09.191.
10. Płatek R., *MATLAB i Simulink w automatyce przemysłowej*. „Automatyka", R. 1, Nr 3, 2015, 126–128.
11. Schwarz M.H.; Boercsoek J., *Advances of OPC Client Server Architectures for Maintenance Strategies: a Research and Development Area not only for Industries*. WSEAS Transactions on Systems and Control, Vol. 3, No. 3, 2008, 195–207.
12. Pendli P.K., Gorbatchev V., Schwarz M.H., Börcsök J., *OPC and its Strategies for Redundancy*. International Control Conference (ICC 2006-Control 2006).
13. Aguado J.V., Borzacchiello D., Ghnatios C., Lebel F., Upadhyay R., Binetruy C., Chinesta F., *A Simulation App based on reduced order modeling for manufacturing optimization of composite outlet guide vanes*. Advanced Modeling and Simulation in Engineering Sciences, 4(1), 2017, DOI: 10.1186/s40323-017-0087-y.

Construction of Communication Structure: Programming Software for Off-line Robots – MATLAB

Abstract: When designing and building factories in accordance with the idea of Industry 4.0, it is common to use robotic systems. Using such systems, it is important to monitor and analyze the operating parameters of these systems. In that it is possible to increase the efficiency, work safety, more efficient production management. For this type of systems, data exchange between devices are done using communication standards, e.g.: Profibus, DeviceNet, RS-485, OPC. Within the work, we presented our own solution using the OPC standard to exchange data between a robotic system and the MATLAB environment. An application using the GUI was presented, collecting and processing process data from a robotic station equipped with ABB manipulators.

Keywords: server OPC, MATLAB/Simulink package, robotic position

dr inż. Magdalena Muszyńska
magdaw@prz.edu.pl
ORCID: 0000-0002-0113-6159

Ukończyła studia w zakresie mechatronika na Politechnice Rzeszowskiej im. Ignacego Łukasiewicza w 2005 r. W roku akademickim 2005/2006 podjęła studia doktoranckie na Wydziale Budowy Maszyn i Lotnictwa Politechniki Rzeszowskiej. Od tego też roku jest pracownikiem Katedry Mechaniki Stosowanej i Robotyki, gdzie obecnie pracuje na stanowisku adiunkta. Stopień doktora nauk technicznych uzyskała w 2012 r. Jej zainteresowania naukowe dotyczą m.in. zagadnień sterowania robotami kołowymi z uwzględnieniem metod adaptacyjnych i inteligentnych. Jest współautorem publikacji krajowych i międzynarodowych.

dr inż. Dariusz Szybicki
dszybicki@prz.edu.pl
ORCID: 0000-0003-3648-9808

Urodził się w Przeworsku. Studia wyższe ukończył na Politechnice Rzeszowskiej im. Ignacego Łukasiewicza. W latach 2009–2013 był uczestnikiem studiów doktoranckich z zakresu Mechaniki na Wydziale Budowy Maszyn i Lotnictwa Politechniki Rzeszowskiej. W 2014 r. obronił pracę doktorską w dyscyplinie Mechanika. Od 2014 r. jest zatrudniony na stanowisku adiunkta w Katedrze Mechaniki Stosowanej i Robotyki. Jego zainteresowania naukowe dotyczą szeroko pojętej mechatroniki. W ramach projektów badawczych zajmuje się projektowaniem stacji zrobotyzowanych oraz programowaniem robotów przemysłowych.

mgr inż. Paulina Pietruś
p.pietrus@prz.edu.pl
ORCID: 0000-0002-6428-0959

Urodziła się w Rzeszowie. Studia wyższe ukończyła na Politechnice Rzeszowskiej im. Ignacego Łukasiewicza. W roku akademickim 2016/2017 podjęła studia doktoranckie na Wydziale Budowy Maszyn i Lotnictwa Politechniki Rzeszowskiej. W tym samym roku rozpoczęła pracę w Katedrze Mechaniki Stosowanej i Robotyki, gdzie obecnie jest asystentem. Jej zainteresowania naukowe – szeroko pojęta mechatronika, programowanie robotów przemysłowych.