Politechnika Rzeszowska im. I. Łukasiewicza

Materiały szkoleniowe – Programowanie robotów ABB

Konfiguracja połączenia Modbus TCP

2026-06-15

1 Podstawy komunikacji Modbus TCP

Modbus TCP jest jednym z popularnych protokołów komunikacyjnych stosowanych w automatyce przemysłowej. Umożliwia wymianę danych pomiędzy urządzeniami podłączonymi do sieci Ethernet, na przykład pomiędzy sterownikiem PLC, komputerem, panelem HMI, czujnikiem, falownikiem albo robotem przemysłowym.

W tym ćwiczeniu protokół Modbus TCP zostanie wykorzystany do komunikacji z robotem ABB wyposażonym w kontroler OmniCore. Dzięki temu możliwe będzie między innymi odczytywanie wybranych stanów robota, zapisywanie sygnałów sterujących oraz odczytywanie wartości liczbowych, takich jak pozycje osi robota lub pozycja narzędzia TCP.

Modbus TCP można potraktować jako prosty sposób zadawania pytań i otrzymywania odpowiedzi. Jedno urządzenie wysyła zapytanie, a drugie odpowiada danymi albo wykonuje określony zapis.

Przykład:
Komputer pyta robota: jaki jest stan sygnału Motor On?

Robot odpowiada: sygnał Motor On = 1

albo:

Komputer wysyła do robota: ustaw sygnał Start na 1

Robot odbiera polecenie i zmienia stan odpowiedniego sygnału

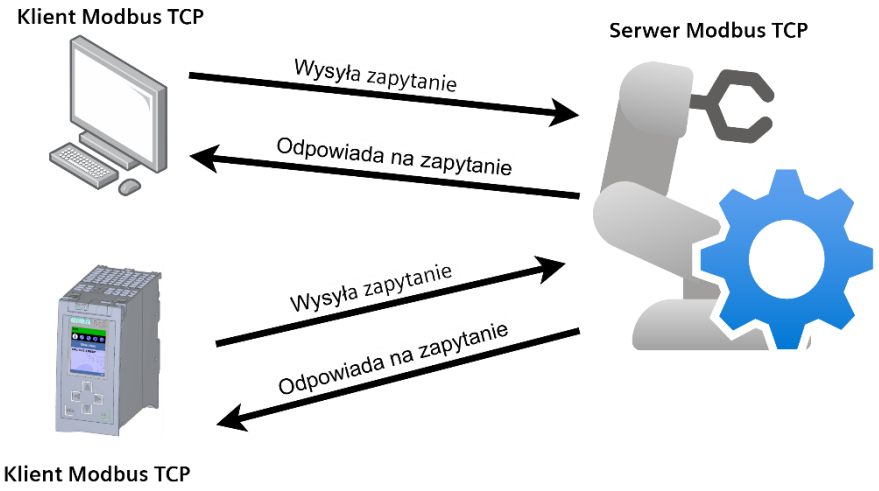
Komunikacja klient–serwer

W protokole Modbus TCP występują dwie podstawowe role: **klient** oraz **serwer**.

Klient Modbus TCP to urządzenie, które rozpoczyna komunikację. Klient wysyła zapytanie, na przykład prośbę o odczyt danych albo polecenie zapisu wartości.

Serwer Modbus TCP to urządzenie, które czeka na zapytania od klienta. Po odebraniu zapytania serwer odsyła odpowiedź lub zapisuje otrzymaną wartość w swojej pamięci.

W typowym układzie automatyki klientem może być sterownik PLC, komputer z programem diagnostycznym albo panel HMI. Serwerem może być robot, falownik, moduł wejść/wyjść, licznik energii albo inne urządzenie przemysłowe.



W tym ćwiczeniu najczęściej robot ABB będzie pracował jako **serwer Modbus TCP**. Oznacza to, że robot będzie czekał na zapytania z programu Modbus Poll albo ze sterownika PLC. Program lub sterownik będzie mógł odczytywać dane z robota oraz zapisywać wybrane sygnały.

Ważne jest zapamiętanie jednej zasady: Serwer nie rozpoczyna komunikacji samodzielnie. Serwer czeka, aż klient wyśle zapytanie.

Adres IP i port komunikacyjny

Ponieważ Modbus TCP korzysta z sieci Ethernet, każde urządzenie biorące udział w komunikacji musi mieć ustawiony adres IP. Adres IP pozwala wskazać, z którym urządzeniem w sieci chcemy się połączyć. Można go porównać do adresu budynku. Jeżeli znamy adres, możemy wysłać wiadomość do właściwego odbiorcy.

Przykładowe adresy IP:

192.168.125.1
192.168.0.10
127.0.0.1

Oprócz adresu IP używany jest także numer portu. Port wskazuje, z jaką usługą w danym urządzeniu chcemy się połączyć. W przypadku Modbus TCP standardowo używany jest port 502.

Dlatego podczas konfiguracji połączenia należy znać dwie podstawowe informacje:

Parametr	**Znaczenie**	**Przykład**
Adres IP	Adres urządzenia w sieci	192.168.125.1
Port TCP	Numer usługi Modbus TCP	502

Jeżeli adres IP lub port zostanie wpisany nieprawidłowo, klient Modbus nie połączy się z serwerem.

Obszary danych w Modbus TCP

W protokole Modbus dane są przechowywane w różnych obszarach pamięci, najważniejsze są dwa typy danych:

Typ danych	Opis	Przykład zastosowania
Cewka (Coil)	Pojedynczy bit, czyli wartość 0 albo 1	sygnał start, stop, gotowość, błąd
Rejestr (Holding register)	Rejestr 16-bitowy, czyli komórka pamięci o rozmiarze słowa, przechowująca wartość liczbową.	pozycja, prędkość, wartość zadana

Coil, czyli sygnały binarne

Cewka (coil) to pojedynczy sygnał binarny. Może mieć tylko dwa stany:

0 – sygnał nieaktywny
1 – sygnał aktywny

Można go porównać do przełącznika, który jest wyłączony albo włączony.

Przykłady sygnałów typu coil:

Motor On Status = 1	silniki robota są załączone
Error = 0	robot nie zgłasza błędu
Start = 1	wysłano polecenie startu
Soft Stop = 1	wysłano polecenie zatrzymania programu

Cewki są wygodne do przesyłania prostych informacji logicznych, na przykład:

- czy robot jest gotowy,
- czy wystąpił błąd,
- czy program robota jest uruchomiony,
- czy należy uruchomić określoną funkcję,
- czy aktywowano sygnał startu albo stopu.

W sterownikach PLC sygnały tego typu odpowiadają zwykle wejściom i wyjściom cyfrowym.

Holding Registers, czyli rejestry liczbowe

Register (rejestr) to komórka pamięci przeznaczona do przechowywania liczby. Pojedynczy rejestr Modbus ma długość 16 bitów.

W rejestrach można przesyłać na przykład:

- wartości zadane,
- wartości pomiarowe,
- numery programów,
- prędkości,
- pozycje,
- liczniki,
- kody błędów.

Przykład: Register 150 = 25

Może to oznaczać, że urządzenie zewnętrzne przesłało do robota wartość 25, którą program robota może później wykorzystać, na przykład jako numer programu, prędkość procesu albo wartość zadaną.

Ważne jest, że jeden rejestr Modbus przechowuje tylko 16 bitów danych. Jeżeli trzeba przesłać większą liczbę, na przykład liczbę 32-bitową albo liczbę zmiennoprzecinkową typu REAL, wtedy wykorzystywane są dwa kolejne rejestry.

Przykład: Register_0 + Register_1 = jedna liczba REAL

Dlatego przy odczycie pozycji robota często należy odczytywać pary rejestrów, a nie pojedyncze rejestry.

Kody funkcji Modbus

Klient Modbus, wysyłając zapytanie do serwera, musi określić, co chce zrobić. Do tego służy **kod funkcji**, czyli numer operacji.

Kod funkcji informuje serwer czy klient chce odczytać dane, zapisać pojedynczą wartość, czy zapisać wiele wartości naraz.

Najczęściej używane funkcje w tym ćwiczeniu przedstawiono w tabeli.

Tab. 1. Tabela najczęściej używanych funkcji Modbus

Kod funkcji	Nazwa angielska	Znaczenie
01	Read Coils	Odczyt sygnałów binarnych typu cewka
03	Read Holding Registers	Odczyt rejestrów liczbowych
05	Write Single Coil	Zapis pojedynczego sygnału cewki
06	Write Single Register	Zapis pojedynczego rejestru

15 / 0F	Write Multiple Coils	Zapis wielu sygnałów cewki
16 / 10	Write Multiple Registers	Zapis wielu rejestrów

W niektórych środowiskach kody funkcji są podawane w systemie dziesiętnym, a w innych w systemie szesnastkowym. Dlatego można spotkać dwa zapisy tej samej funkcji.

Przykład: **15** dziesiętnie = **0F** szesnastkowo, **16** dziesiętnie = **10** szesnastkowo

Można spotkać również rozwiązania, gdzie kody funkcji modbus nie są podawane jawnie, a są ukryte pod działaniem elementów GUI

Odczyt danych a zapis danych

W komunikacji Modbus TCP należy odróżniać odczyt od zapisu. **Odczyt** oznacza, że klient pobiera dane z serwera. Przykładowo komputer może odczytać z robota informację czy silniki są załączone.

Klient pyta: jaki jest stan sygnału Motor On?

Serwer odpowiada: Motor On = 1

Zapis oznacza, że klient wysyła wartość do serwera. Przykładowo komputer może ustawić wybrany sygnał wejściowy robota.

Klient wysyła: ustaw sygnał mb_di96 na 1

Serwer zapisuje: mb_di96 = 1

Nie wszystkie adresy można zapisywać. Niektóre obszary pamięci są przeznaczone tylko do odczytu. Jest to zabezpieczenie przed przypadkową zmianą danych, które powinny być jedynie obserwowane. Przykład:

Stan robota można odczytać.
Polecenie startu można zapisać.
Pozycji osi robota nie należy zapisywać przez Modbus, tylko odczytywać.

Dlaczego Modbus TCP jest często stosowany w automatyce?

Modbus TCP jest popularny, ponieważ jest prosty, czytelny i obsługiwany przez wiele urządzeń przemysłowych. Można go spotkać w sterownikach PLC, panelach operatorskich, modułach wejść/wyjść, falownikach, licznikach energii oraz urządzeniach pomiarowych.

Najważniejsze zalety protokołu Modbus TCP to:

- prosta zasada działania,
- komunikacja przez standardową sieć Ethernet,
- łatwe testowanie za pomocą programów diagnostycznych,
- możliwość odczytu i zapisu danych,
- szeroka dostępność w urządzeniach automatyki.

Trzeba jednak pamiętać, że prostota protokołu oznacza również ograniczenia. Modbus TCP sam w sobie nie sprawdza, czy użytkownik ma uprawnienia do wykonania danej operacji. Nie szyfruje również przesyłanych danych. Z tego powodu nie powinien być udostępniany w niezabezpieczonej sieci ani bezpośrednio przez Internet.

Podsumowanie

Modbus TCP służy do wymiany danych pomiędzy urządzeniami w sieci Ethernet. Jedno urządzenie pełni rolę klienta i wysyła zapytania, a drugie pełni rolę serwera i odpowiada na te zapytania.

W dalszej części instrukcji robot ABB OmniCore będzie najczęściej traktowany jako serwer Modbus TCP. Program Modbus Poll lub sterownik PLC będzie klientem, który odczytuje albo zapisuje dane w robocie.

Najważniejsze pojęcia z tego rozdziału:

Klient Modbus	Urządzenie wysyłające zapytanie.
Serwer Modbus	Urządzenie odpowiadające na zapytanie.
Adres IP	Adres urządzenia w sieci Ethernet.
Port 502	Standardowy port używany przez Modbus TCP.
Coil	Pojedynczy sygnał 0 lub 1.
Register	Rejestr przechowujący liczbę.
Kod funkcji	Numer operacji, np. odczyt albo zapis.

Po zrozumieniu tych pojęć można przejść do konfiguracji dodatku Modbus TCP w RobotStudio oraz do pierwszego połączenia z robotem.

2 Dodatek Modbus TCP dla ABB OmniCore

Aby robot ABB z kontrolerem OmniCore mógł komunikować się przez Modbus TCP, w systemie robota należy zainstalować odpowiedni dodatek programowy. W dokumentacji ABB dodatek ten występuje pod nazwą **Modbus TCP Add-in for OmniCore systems**.

Dodatek Modbus TCP rozszerza możliwości kontrolera robota o komunikację przez sieć Ethernet z wykorzystaniem protokołu Modbus. Dzięki temu robot może wymieniać dane z innymi urządzeniami automatyki, na przykład ze sterownikiem PLC, komputerem, panelem HMI, czujnikiem, falownikiem lub chwytakiem.

W praktyce oznacza to, że zewnętrzne urządzenie może:

- odczytywać wybrane stany robota,
- zapisywać wybrane sygnały sterujące,
- odczytywać pozycje osi robota,
- odczytywać pozycję TCP,
- wymieniać dane liczbowe z programem RAPID,
- komunikować się z urządzeniami obsługującymi Modbus TCP.

Dodatek przeznaczony jest dla kontrolerów **ABB OmniCore** i działa z systemem **RobotWare 7.13 lub nowszym**. Według dokumentacji ABB dodatek obsługuje zarówno opcję **Modbus TCP Server**, jak i Modbus **TCP Client**.

Opcje dostępne w dodatku

Podczas dodawania Modbus TCP do systemu robota można wybrać odpowiednie opcje. Każda z nich przeznaczona jest do innego sposobu komunikacji.

Opcja	Rola robota	Zastosowanie
Modbus TCP Server	Robot udostępnia dane innym urządzeniom	PLC lub komputer odczytuje i zapisuje dane w robocie
Modbus TCP Client	Robot sam komunikuje się z innym urządzeniem	Robot odczytuje dane z czujnika, falownika lub modułu I/O
Gripper Template	Robot obsługuje chwytak przez Modbus	Sterowanie chwytakiem obsługującym Modbus TCP/RTU

Robot jako Modbus TCP Server

W trybie **Modbus TCP Server** robot pełni rolę serwera. Oznacza to, że czeka na zapytania wysyłane przez inne urządzenie, na przykład komputer z programem Modbus Poll albo sterownik PLC.

W tym trybie robot nie rozpoczyna komunikacji samodzielnie. To klient decyduje, kiedy chce odczytać albo zapisać dane.

Przykłady zapytań wysyłanych przez klienta:

- Odczytaj stan silników robota.
- Odczytaj informację czy robot jest w trybie automatycznym.
- Ustaw sygnał Start na 1.
- Odczytaj pozycję osi J1.
- Odczytaj aktualną pozycję TCP.

W trybie serwera robot nasłuchuje na porcie TCP **502**. Jest to standardowy port używany przez Modbus TCP.

W dokumentacji ABB podano, że serwer Modbus TCP w OmniCore obsługuje między innymi funkcje odczytu i zapisu cewek oraz rejestrów, a komunikacja jest realizowana cyklicznie z czasem około **200 ms**.

Robot jako Modbus TCP Client

W trybie **Modbus TCP Client** sytuacja jest odwrotna. Robot sam wysyła zapytania do innego urządzenia, które pełni rolę serwera Modbus. Ten tryb może być używany wtedy, gdy robot ma samodzielnie odczytywać dane z urządzenia zewnętrznego lub nim sterować.

Przykłady zastosowania:

- Robot odczytuje wartość z czujnika. Robot zapisuje wartość zadaną do falownika.
- Robot odczytuje stan wejść z modułu I/O. Robot wysyła polecenie do urządzenia wykonawczego.

Tryb klienta jest bardziej zaawansowany niż tryb serwera, ponieważ wymaga przygotowania odpowiednich danych i instrukcji w programie RAPID. Z tego powodu w podstawowej części ćwiczenia skupimy się głównie na trybie **Modbus TCP Server**.

Szablon chwytaka — Gripper Template

Dodatek zawiera również opcję **Gripper Template**. Jest to przykładowy szablon przeznaczony do obsługi chwytaka przez Modbus.

W praktyce chwytak może być urządzeniem, które komunikuje się z robotem za pomocą Modbus TCP lub Modbus RTU. Dzięki szablonowi można przygotować prosty interfejs do jego sterowania, na przykład do otwierania i zamykania szczęk.

Przykładowe polecenia dla chwytaka:

- Otwórz chwytak.
- Zamknij chwytak.
- Ustaw pozycję chwytaka.
- Odczytaj aktualną pozycję chwytaka.
- Sprawdź, czy wystąpił błąd chwytaka.

W tej instrukcji opcja **Gripper Template** nie będzie szczegółowo omawiana. Można ją potraktować jako przykład rozszerzenia systemu robota o dodatkowe urządzenie wykonawcze.

Wymagania dla trybu Modbus TCP Server

Aby robot mógł pracować jako serwer Modbus TCP, należy spełnić kilka warunków.

Tab. 2. Wymagania dla dodatku Modbus TCP for Omnicore

Wymaganie	**Znaczenie**
Kontroler OmniCore	Robot musi być wyposażony w kontroler ABB OmniCore
RobotWare 7.13 lub nowszy	Dodatek wymaga odpowiedniej wersji systemu robota
Dodatek Modbus TCP	Dodatek musi być zainstalowany i dodany do systemu
Opcja Modbus TCP Server	Podczas konfiguracji należy zaznaczyć tryb serwera
Opcja Multitasking	Dla serwera wymagana jest opcja **3114-1 Multitasking**
Poprawna konfiguracja sieci	Komputer i robot muszą mieć poprawne adresy IP

Opcja **Multitasking** jest potrzebna dlatego, że serwer Modbus działa jako zadanie pracujące w tle. Dzięki temu robot może jednocześnie wykonywać program RAPID i obsługiwać komunikację Modbus.

Jakie dane udostępnia robot w trybie serwera?

Po uruchomieniu trybu **Modbus TCP Server** robot udostępnia klientowi określone obszary danych. Są to przede wszystkim:

- cewki systemowe,
- cewki użytkownika,
- rejestry systemowe,
- rejestry użytkownika.

Najważniejsze obszary danych przedstawiono w tabeli.

Tab. 3. Zakresy serwera Modbus kontrolera robota

Zakres adresów	Typ danych	Znaczenie
0–15	Coils	Stany systemowe robota, np. Motor On, Auto, Error
16–20	Coils	Polecenia systemowe, np. Motor On, Motor Off, Start
32–95	Coils	Sygnały użytkownika do odczytu
96–159	Coils	Sygnały użytkownika do zapisu i odczytu
0–37	Registers	Pozycje osi, pozycja TCP i prędkość TCP
50–149	Registers	Rejestry użytkownika do odczytu
150–299	Registers	Rejestry użytkownika do zapisu i odczytu

Na tym etapie nie trzeba zapamiętywać wszystkich adresów. Ważne jest zrozumienie, że każdy sygnał lub rejestr ma swój numer. Klient Modbus musi znać ten numer, aby odczytać lub zapisać właściwe dane.

Tab. 4. Cewki serwera Modbus kontrolera robota

Adres	Funkcja	Odczyt	Zapis	Nazwa sygnału	Typ sygnału
0	Motor On Status	✓		mb_sdo0	Wyjścia systemowe
1	Auto On Status	✓		mb_sdo1	
2	Cycle On Status	✓		mb_sdo2	
3	Error	✓		mb_sdo3	
4	Nie dotyczy	✓		mb_sdo4	
5		✓		mb_sdo5	
...		✓		...	
14		✓		mb_sdo14	
15		✓		mb_sdo15	
16	Motor On Action	✓	✓	mb_sdi16	Wejścia systemowe
17	Motor Off Action	✓	✓	mb_sdi17	
18	PP To Main	✓	✓	mb_sdi18	
19	Start	✓	✓	mb_sdi19	
20	Soft Stop	✓	✓	mb_sdi20	
21	Nie dotyczy	✓	✓	mb_sdi21	
22		✓	✓	mb_sdi22	
...		✓	✓	...	
30		✓	✓	mb_sdi30	
31		✓	✓	mb_sdi31	
32		✓		mb_do32	Wyjścia użytkownika
33		✓		mb_do33	
...		✓		...	
94		✓		mb_do94	
95		✓		mb_do95	
96		✓	✓	mb_di96	Wejścia użytkownika
97		✓	✓	mb_di97	

...		✓	✓	...	
158		✓	✓	mb_di158	
159		✓	✓	mb_di159	

Sygnały systemowe są związane bezpośrednio ze stanem robota. Informują na przykład, czy robot ma załączone silniki, czy jest w trybie automatycznym albo czy wystąpił błąd.

Sygnały użytkownika są przeznaczone do wykorzystania w programie użytkownika. Można ich używać do własnych celów, na przykład do wymiany informacji pomiędzy robotem a PLC.

Tab. 5. Rejestry serwera Modbus kontrolera robota

Adres	Nazwa sygnału	Typ danych	Odczyt	Zapis	Format	Jednostka
0-1	ROB1_J1	FLOAT32	✓		IEEE 754 Big-Ending	stopnie
2-3	ROB1_J2		✓			stopnie
4-5	ROB1_J3		✓			stopnie
6-7	ROB1_J4		✓			stopnie
8-9	ROB1_J5		✓			stopnie
10-11	ROB1_J6		✓			stopnie
12-13	ROB1_EXA		✓			stopnie/mm
14-15	ROB1_EXB		✓			stopnie/mm
16-17	ROB1_EXC		✓			stopnie/mm
18-19	ROB1_EXD		✓			stopnie/mm
20-21	ROB1_EXE		✓			stopnie/mm
22-23	ROB1_EXF		✓			stopnie/mm
24-25	ROB1_X		✓			mm
26-27	ROB1_Y		✓			mm
28-29	ROB1_Z		✓			mm
30-31	ROB1_Rz		✓			stopnie
32-33	ROB1_Ry		✓			stopnie
34-35	ROB1_Rx		✓			stopnie
36-37	ROB1_TCP Speed		✓			mm/s
38	Adresy rejestrów zarezerwowane dla systemu	UINT16(default)/ INT16 / UINT32 /INT32 /FLOAT32	✓			
39			✓			
...			✓			
48			✓			
49			✓			
50	Rejestry użytkownika tylko do odczytu		✓			
51			✓			
...			✓			
148			✓			
149			✓			
150	Rejestry użytkownika do odczytu i zapisu		✓	✓		
151			✓	✓		
...			✓	✓		

298			✓	✓		
299			✓	✓		

Podsumowanie

Dodatek **Modbus TCP for OmniCore** umożliwia komunikację robota ABB z innymi urządzeniami przez protokół Modbus TCP. Robot może pracować jako serwer, który udostępnia dane, albo jako klient, który sam wysyła zapytania do innych urządzeń.

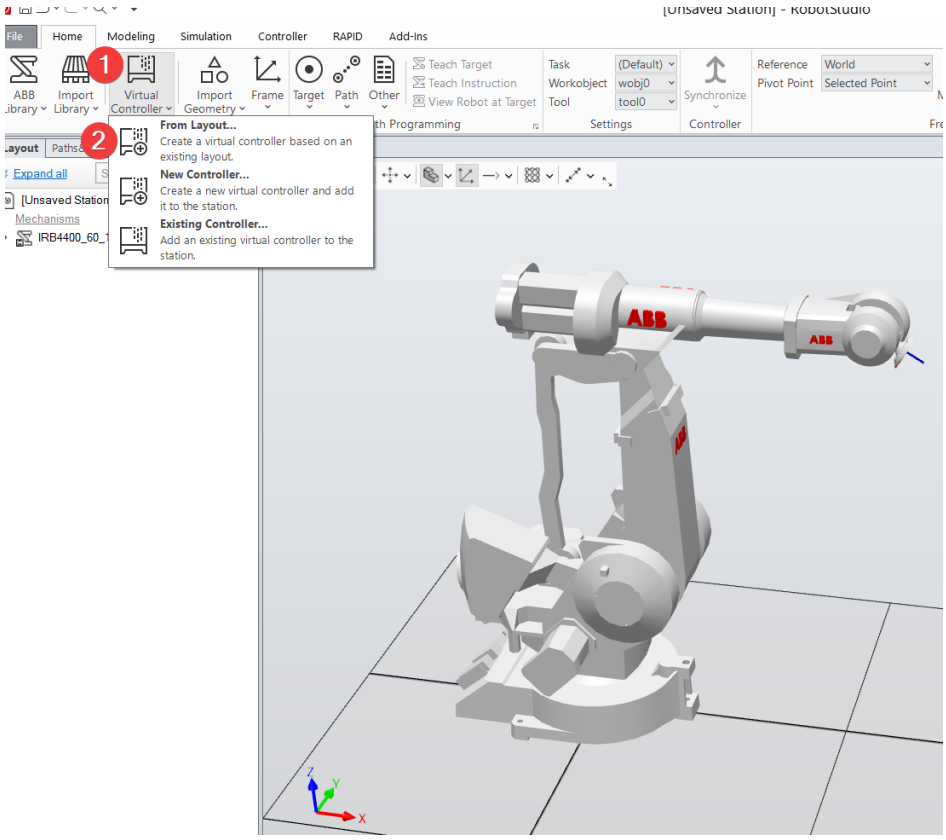
3 Instalacja dodatku Modbus TCP w RobotStudio

Przed rozpoczęciem komunikacji przez Modbus TCP należy przygotować system robota. Sam kontroler OmniCore nie udostępnia tej funkcji w każdym projekcie automatycznie. Trzeba dodać do systemu odpowiedni pakiet programowy, czyli **Modbus TCP for OmniCore**.

Instalację i konfigurację wykonuje się w środowisku **RobotStudio**. W tym rozdziale opisano, jak dodać obsługę Modbus TCP do systemu robota oraz które opcje należy wybrać na potrzeby ćwiczenia.

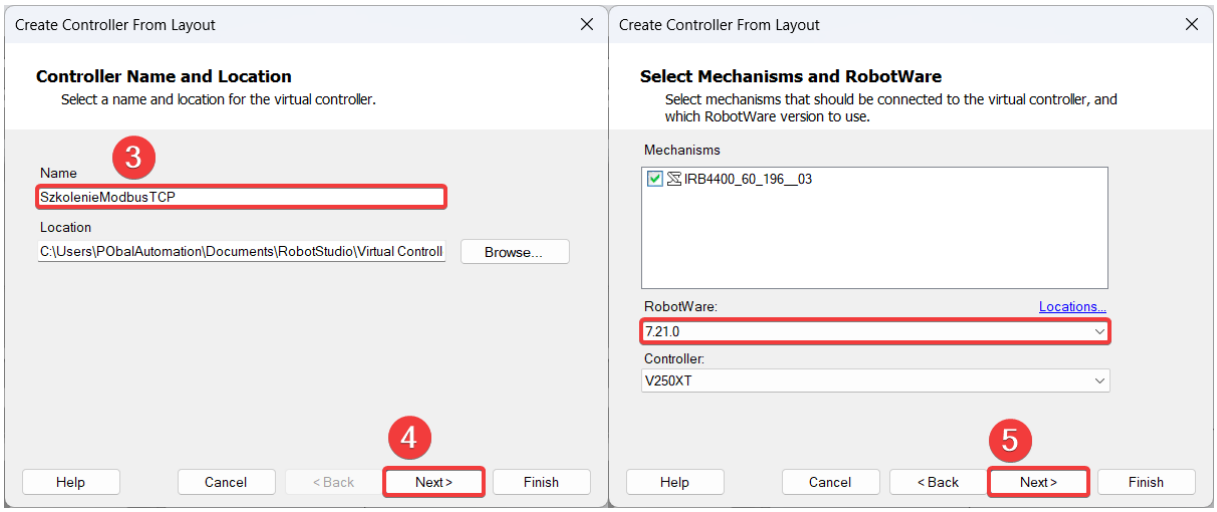
Przygotowanie systemu w RobotStudio

Utwórz system z robotem IRB 4400. System musi zawierać opcję *3114-1 Multitasking*. Standardowo można dodać robota do pustej stacji i utworzyć wirtualny kontroler z layoutu (rys. 1).



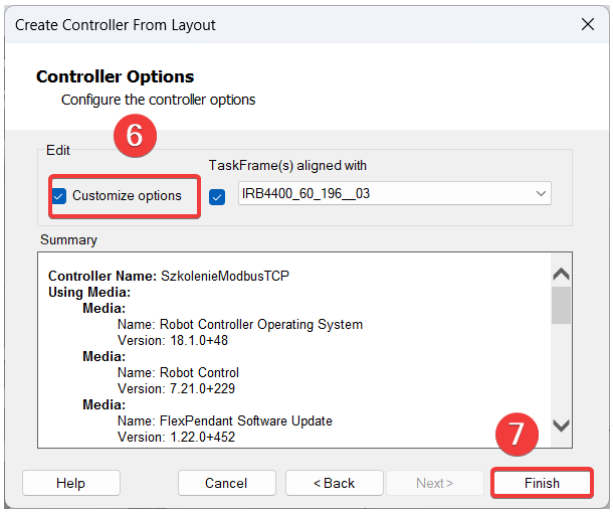
Rys. 1. Dodanie wirtualnego kontrolera z layoutu stacji

Pojawi się okno kreatora wirtualnego kontrolera, gdzie należy podać nazwę oraz wybrać wersję systemu RobotWare. Opcja Modbus TCP jest dostępna tylko na systemy w wersjach 7 i 8.



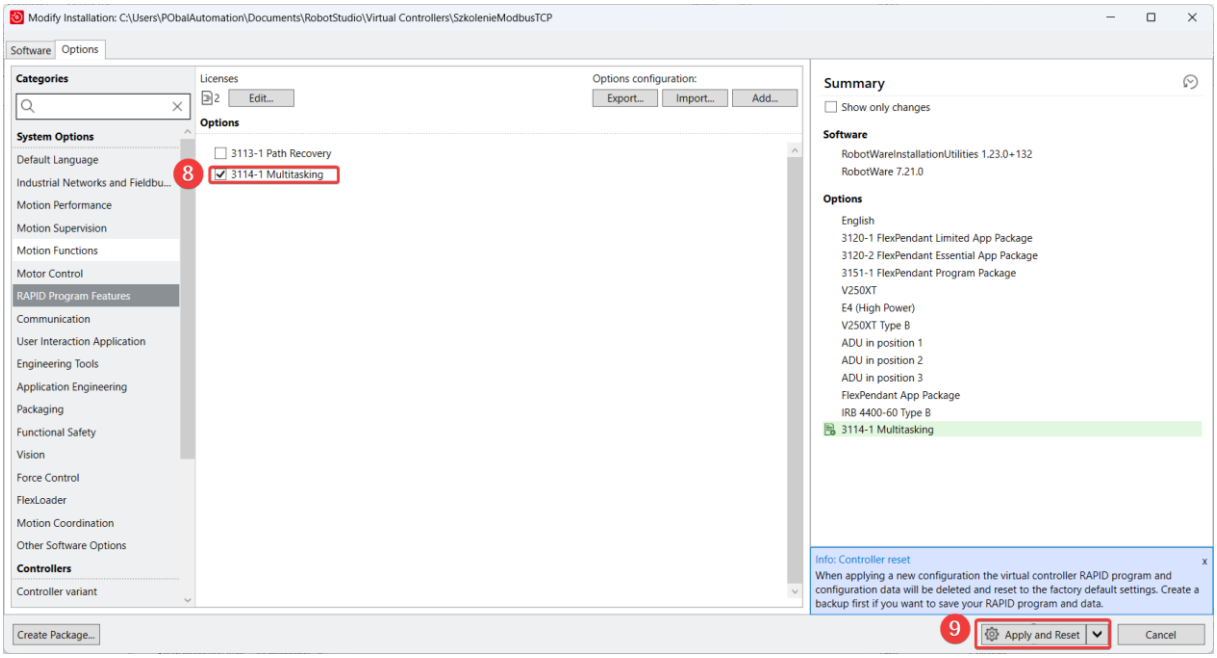
Rys. 2. Okno kreatora wirtualnego kontrolera

W ostatnim kroku kreatora pojawi się możliwość przejścia do edycji opcji kontrolera. Zaznacz ją i kliknij *Finish*.



Rys. 3. Okno podsumowujące kreatora

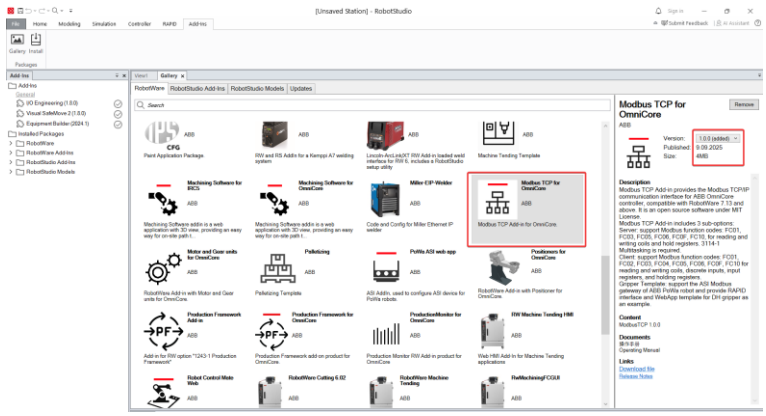
Pojawi się wtedy okno pozwalające wybrać dodatkowe opcje kontrolera. Wybierz opcję *Multitasking* z zakładki *RAPID Program Features* i kliknij *Apply and Reset*.



Rys. 4. Dodanie opcji Multitasking

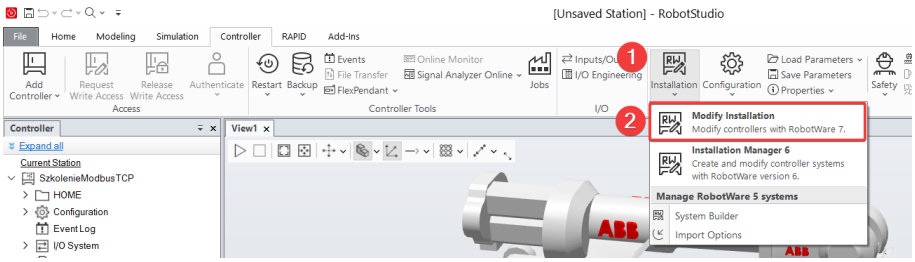
Instalacja dodatku Modbus TCP for Omnicore na kontrolerze robota

Zanim będzie można doinstalować opcję Modbus TCP do wirtualnego kontrolera, należy zainstalować dodatek Modbus TCP for Omnicore w RobotStudio. Dodatek jest dostępny w galerii Add-Ins.



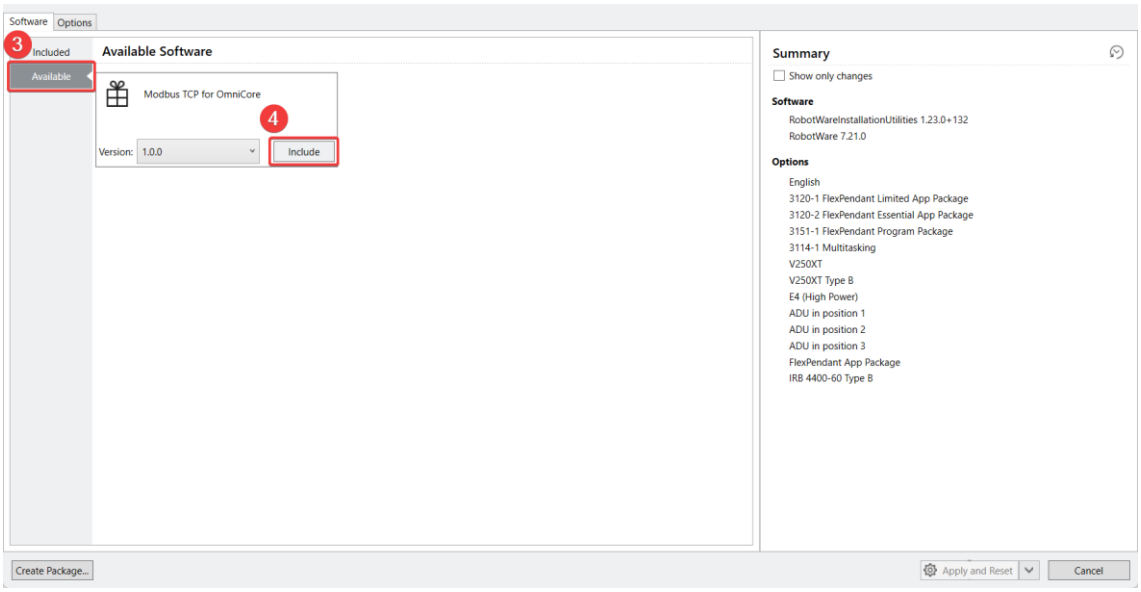
Rys. 5. Dodatek Modbus TCP fro OmniCore galerii dodatków RobotStudio

Następnie, aby zainstalować dodatek na kontrolerze przejdź do Controler/Installation i kliknij Modify Installation.



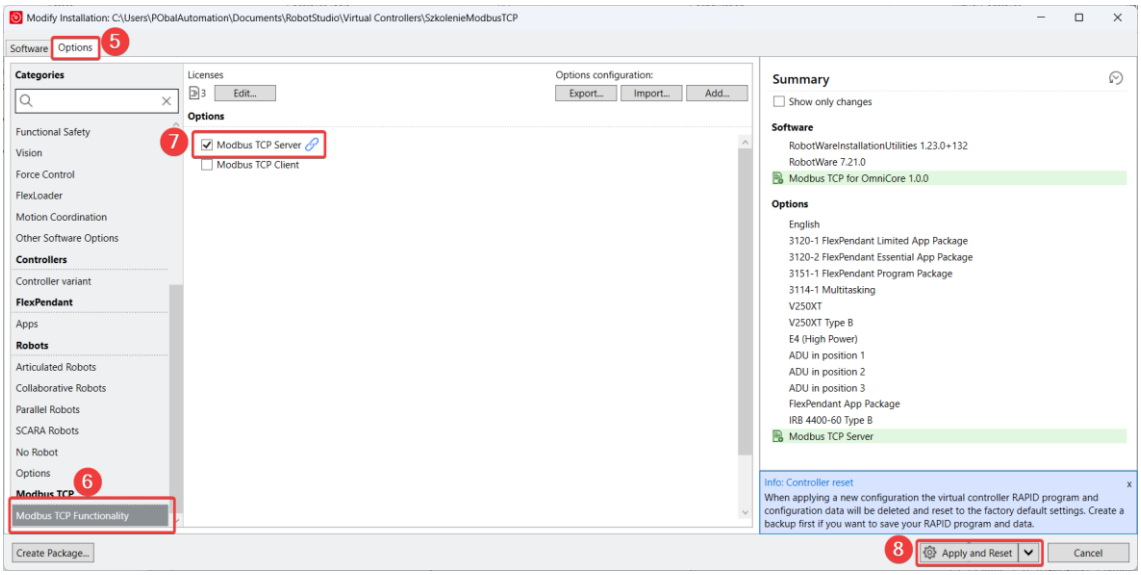
Rys. 6. Otwarcie edytora opcji kontrolera

Pojawi się okno edycji opcji kontrole, gdzie w zakładce Available pojawi się możliwość dodania opcji Modbus TCP.



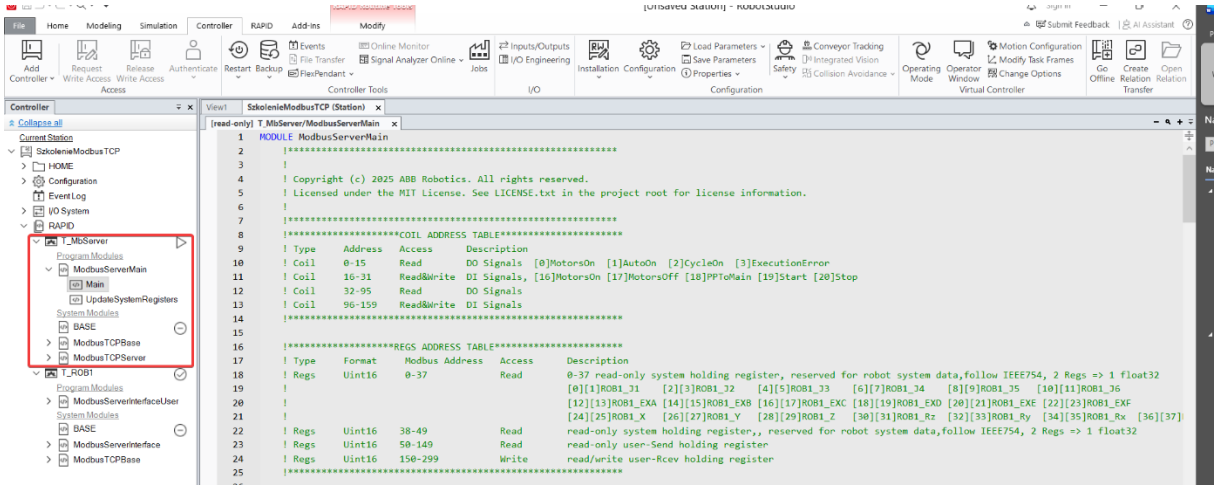
Rys. 7. Dodawanie opcji do kontrolera

Klikamy *Include* i przechodzimy do zakładki *Options*. W drzewku opcji po lewej stronie okna znajduje na samym dole odszukaj zakładki Modbus TCP Functionality i kliknij w nią. Następnie zaznacz opcję Modbus TCP Server i kliknij



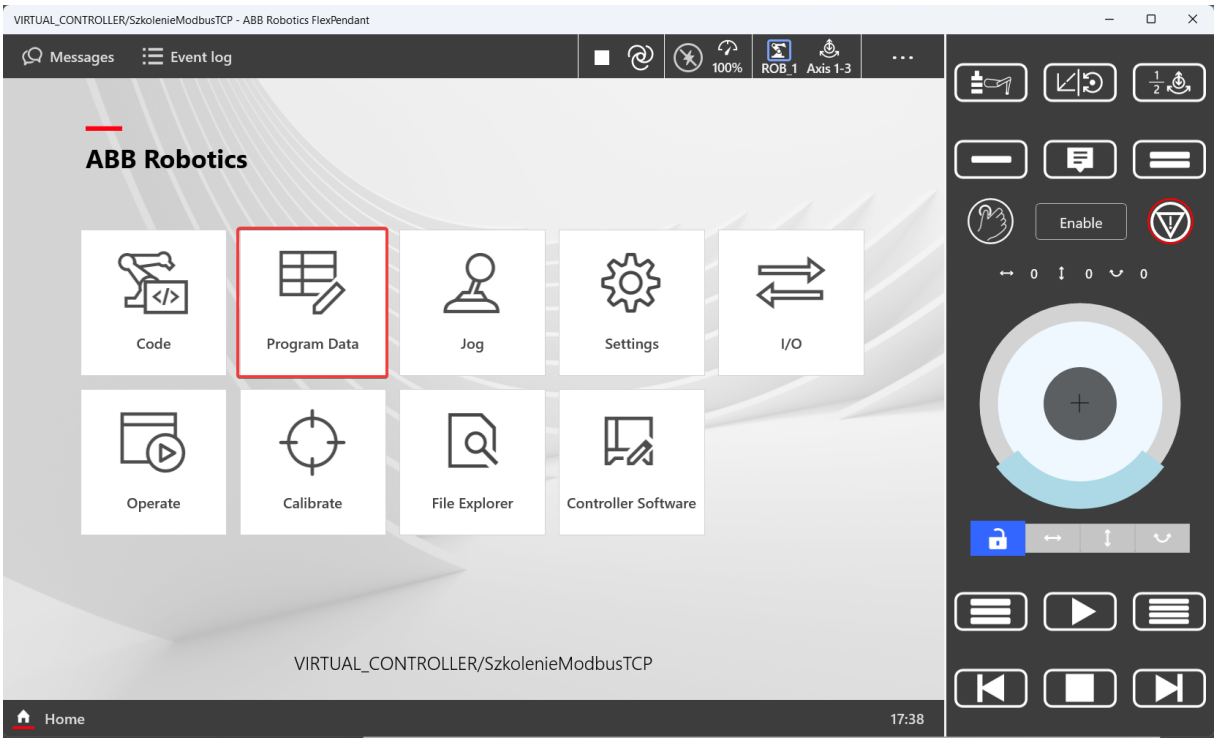
Rys. 8. Instalacja funkcji Modbus TCP Server

Kontroler się zrestartuje a po restarcie na kontrolerze uruchomi się dodatkowy Task pełniący rolę serwera Modbus TCP.



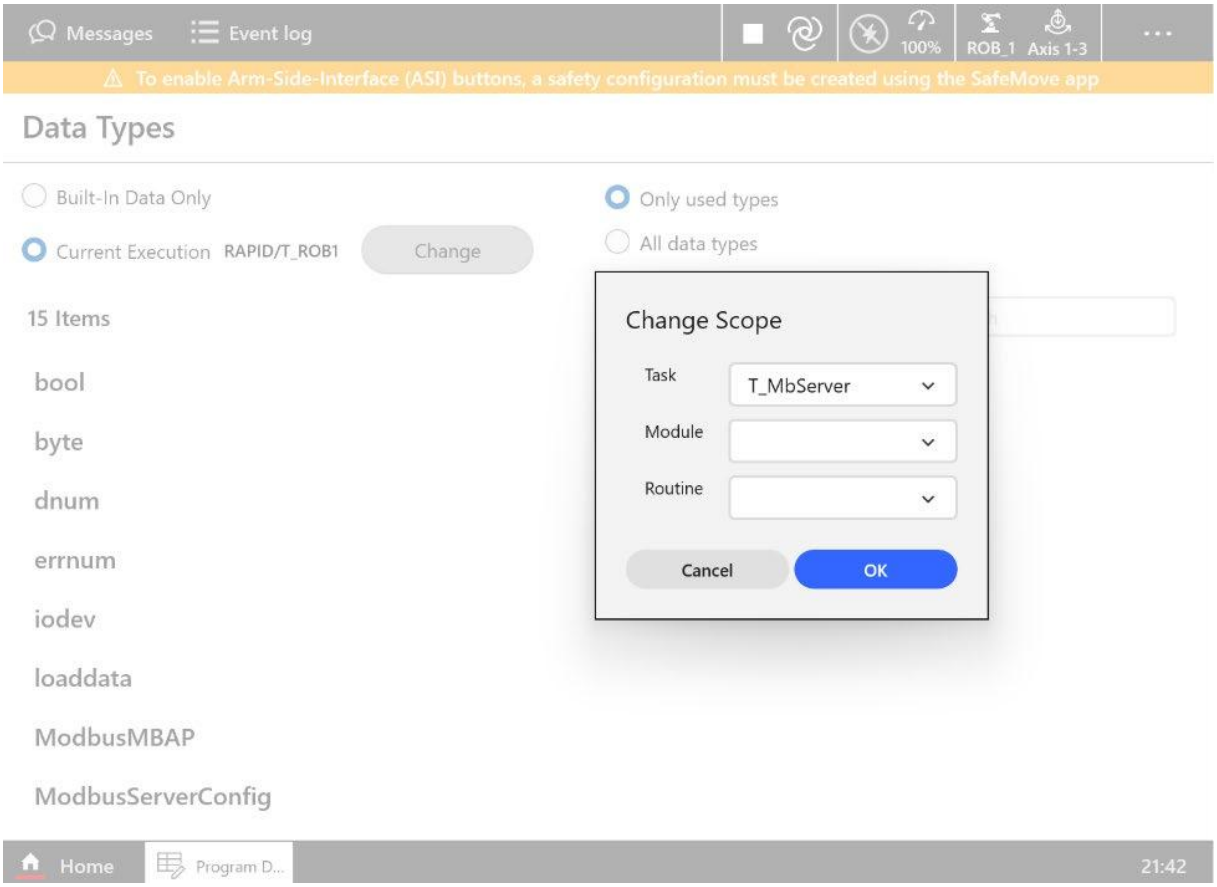
Rys. 9. Uruchomiony task serwera Mobus TCP

Od tego momentu serwer jest uruchomiony i oczekuje na połączenie z klientem. Jeżeli chcemy dokonać zmian w konfiguracji, np. zmienić port, na którym serwer nasłuchuje połączenia, można to zrobić edytując zmienną konfiguracyjną MbServerConfig – w przypadku rzeczywistego kontrolera albo zmienna VirtualConfig – w przypadku kontrolera wirtualnego. Zmiany można dokonać z poziomu flexpendanta wchodząc w Data Types.



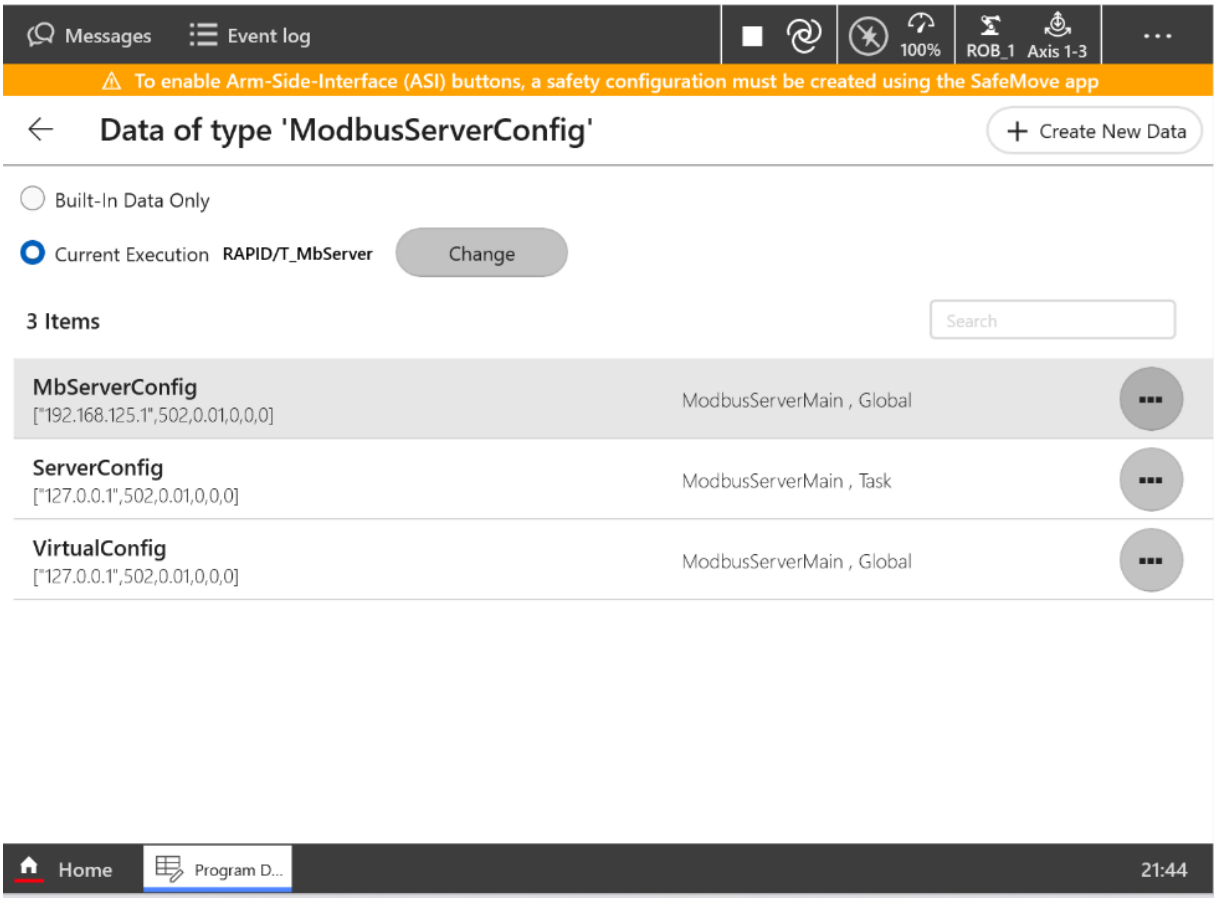
Rys. 10. Okno flexpendanta

Następnie należy wybrać dane typu ModbusServerConfig dla zadania (ang. Task) w którym działa serwer Modbus, czyli T_MbServer.



Rys. 11. Zmiana zadania, w którym są poszukiwane zmienne

Wtedy pojawi się okno wyboru zmiennych dla kontrolera wirtualnego oraz rzeczywistego.



Rys. 12. Przegląd zmiennych konfiguracyjnych Modbus

Typ zmiennej ModbusServerConfig zawiera:

- Adres IP serwera
- Numer portu serwera
- Czas oczekiwania na odbiór danych
- Czas oczekiwania na połączenie
- Liczbę prób wznowienia połącznia
- Liczbę prób wysłania danych

Te elementy można swobodnie modyfikować wedle potrzeb aplikacji.

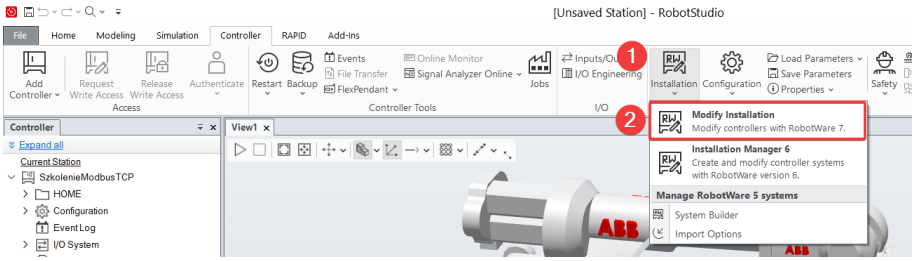
4 Testy działania serwera

Wykorzystamy do tego narzędzie Modbus Poll. Jako zadanie proszę:

- Odczytać dane z rejestrów systemowych
- Włączyć i wyłączyć napędy robota
- Zapisać i odczytać dane z rejestru użytkownika.

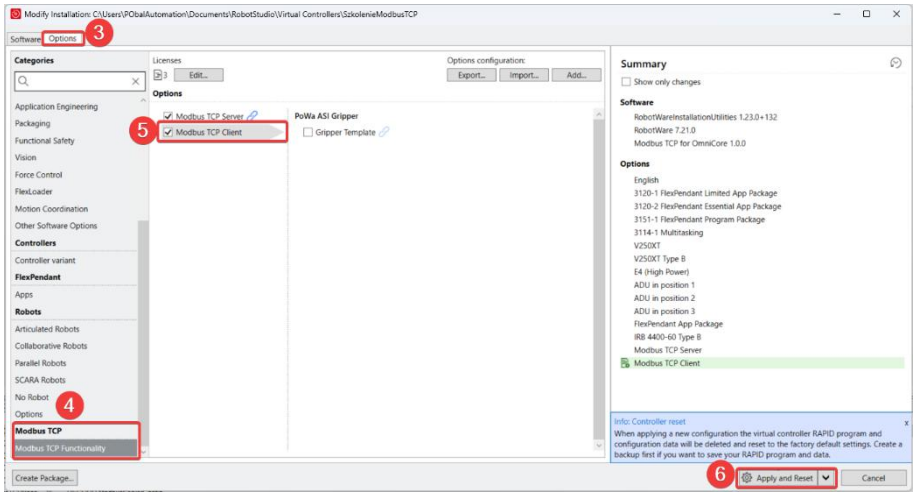
5 Uruchomienie i obsługa klienta Modbus TCP na kontrolerze robota.

Dodatek Modbus TCP posiada opcje pracy kontrolera robota jako klienta Modbus TCP i połączenia go do innego serwera. Aby aktywować tę opcję należy wrócić do opcji instalacji dodatków do kontrolera. W tym celuprzejdź do Controler/Installation i kliknij Modify Installation.



Rys. 13. Otwarcie edytora opcji kontrolera

W oknie, które się pojawi klikamy *Options* i na dole listy wybieramy opcje *Modbus TCP Functionality*. Zaznaczamy opcję *Modbus TCP Client* i klikamy *Apply and Reset*.

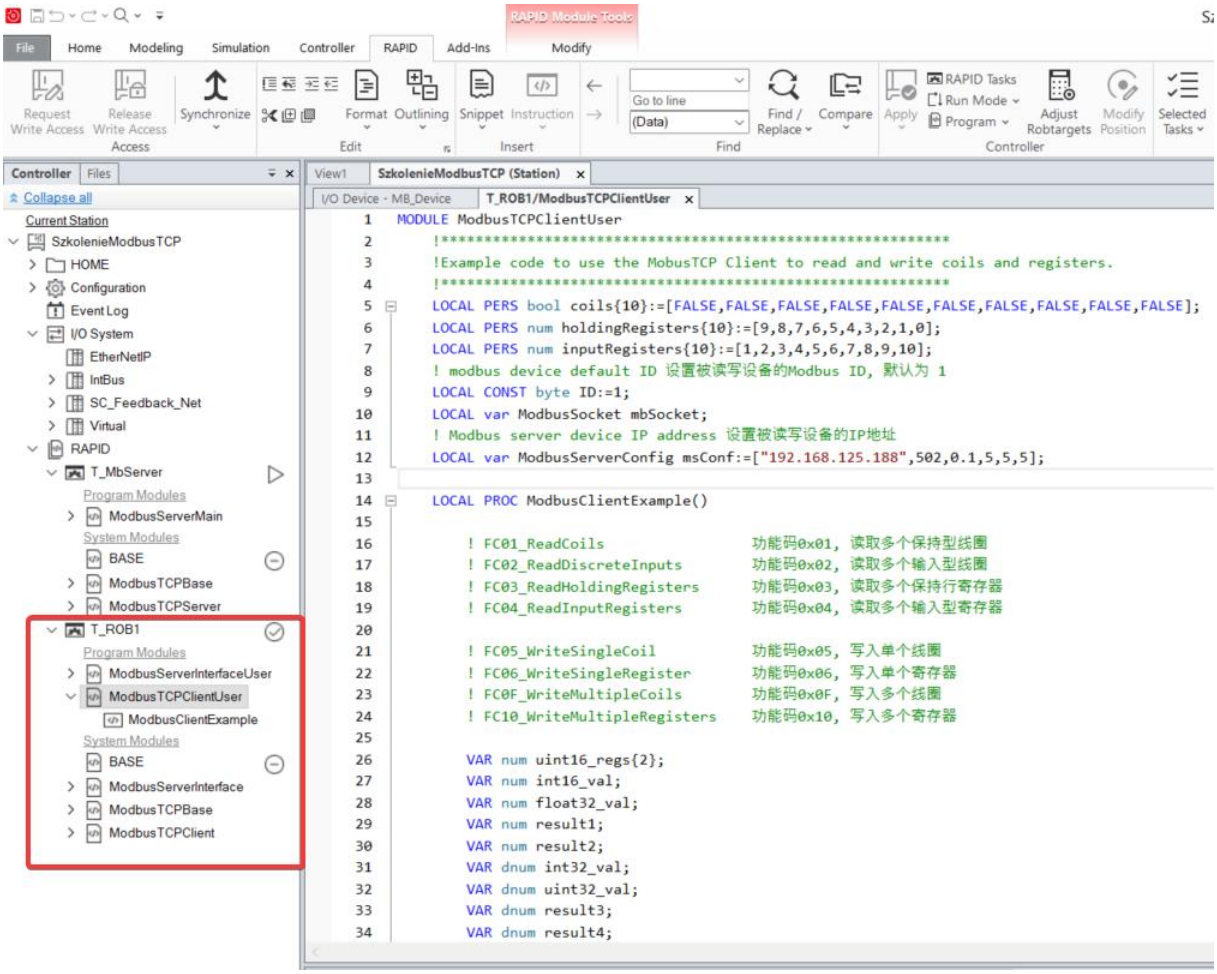


Rys. 14. Okno ustawień instalacji kontrolera

Następnie zgadamy się na restart serwera.

Rys. 15. Okno potwierdzające wprowadzone zmiany

Po restarcie w głównym zadaniu (Tasku) robota pojawią się moduły z programami do obsługi połączenia z serwerem Modbus.



Rys. 16. Stan kontrolera po zmianie

Teraz korzystając odpowiednich funkcji można obsługiwać połączenie z serwerem Modbus.

6 Testy działania klienta Modbus

Do testów wykorzystamy program Modbus Slave. Naszym zadaniem będzie odczyt stanu cewek i zapisanie pozycji robota do pamięci serwera Modbus. Przykładowy program odczytuje cewki ze serwera i zapisuje do niego pozycję robota.

```
1  MODULE MainModule
2      CONST ModbusServerConfig msConf:=["127.0.0.1",1027,0.1,5,5,5];
3      CONST byte ID:=1;
4      VAR ModbusSocket mbSocket;
5      VAR bool cewki{10};
6      VAR robtarget rRobPose;
7
8
9      PROC main()
10         VAR num pos_x;
11         VAR num pos_y;
12         VAR num pos_z;
13         VAR num quat_1;
14         VAR num quat_2;
15         VAR num quat_3;
16         VAR num quat_4;
17         VAR num regs{20};
18
19         WHILE TRUE DO
20             FC01_ReadCoils 0,10,cewki,ID,mbSocket,msConf;
21             WaitTime 2;
22             rRobPose:= CRobT();
23             pos_x:=rRobPose.trans.x;
24             pos_y:=rRobPose.trans.y;
25             pos_z:=rRobPose.trans.z;
26             quat_1:=rRobPose.rot.q1;
27             quat_2:=rRobPose.rot.q2;
28             quat_3:=rRobPose.rot.q3;
29             quat_4:=rRobPose.rot.q4;
30
31             Float32ToTwoUint16 pos_x, regs{1},regs{2};
32             Float32ToTwoUint16 pos_y, regs{3},regs{4};
33             Float32ToTwoUint16 pos_z, regs{5},regs{6};
34             Float32ToTwoUint16 quat_1, regs{7}, regs{8};
35             Float32ToTwoUint16 quat_2, regs{9}, regs{10};
36             Float32ToTwoUint16 quat_3, regs{11}, regs{12};
37             Float32ToTwoUint16 quat_4, regs{13}, regs{14};
38
39             FC10_WriteMultipleRegisters 0,13, regs, ID, mbSocket, msConf;
40         ENDWHILE
41     ENDPROC
42 ENDMODULE
```

Rys. 17. Przykładowy program

Dodatek A — mapa adresów Modbus TCP ABB

Poniższa tabela zawiera uproszczoną mapę adresów wykorzystywanych w ćwiczeniu.
Cewki systemowe:

Adres	Nazwa	Dostęp	Znaczenie
0	Motor On Status	odczyt	stan załączenia silników
1	Auto On Status	odczyt	informacja o trybie automatycznym
2	Cycle On Status	odczyt	informacja o wykonywaniu programu
3	Error	odczyt	informacja o błędzie
16	Motor On Action	zapis/odczyt	polecenie załączenia silników
17	Motor Off Action	zapis/odczyt	polecenie wyłączenia silników
18	PP To Main	zapis/odczyt	ustawienie wskaźnika programu na main
19	Start	zapis/odczyt	uruchomienie programu
20	Soft Stop	zapis/odczyt	zatrzymanie programu

Cewki użytkownika:

Zakres	Nazwy sygnałów	Dostęp	Znaczenie
32–95	mb_do32 – mb_do95	odczyt	wyjścia użytkownika
96–159	mb_di96 – mb_di159	zapis/odczyt	wejścia użytkownika

Rejestry systemowe:

Adresy	Dane	Jednostka
0–1	J1	stopnie
2–3	J2	stopnie
4–5	J3	stopnie
6–7	J4	stopnie
8–9	J5	stopnie
10–11	J6	stopnie
24–25	TCP X	mm
26–27	TCP Y	mm
28–29	TCP Z	mm
30–31	TCP Rz	stopnie
32–33	TCP Ry	stopnie
34–35	TCP Rx	stopnie
36–37	TCP Speed	mm/s

Rejestry użytkownika

Zakres	Dostęp	Zastosowanie
50–149	odczyt	dane wysyłane z robota do klienta
150–299	zapis/odczyt	dane zapisywane przez klienta i odczytywane przez robota