



Politechnika Wrocławska

Sprawozdanie 4

Linia FESTO - Stanowisko 3 - Obrabianie produktu

Skład grupy:

Kateryna Fedoruk - 272609
Tomasz Piaseczny - 272516

Grupa zajęciowa: Czwartek 15:15

1 Konfiguracja sterownika

Struktura sieci i adresacja IP

Sterownik Siemens S7-1200 został skonfigurowany do pracy w sieci przemysłowej. Połączenie między sterownikami zostało wykonane przez interfejs **PN/IE**. Poniższy rysunek przedstawia topologię sieci.

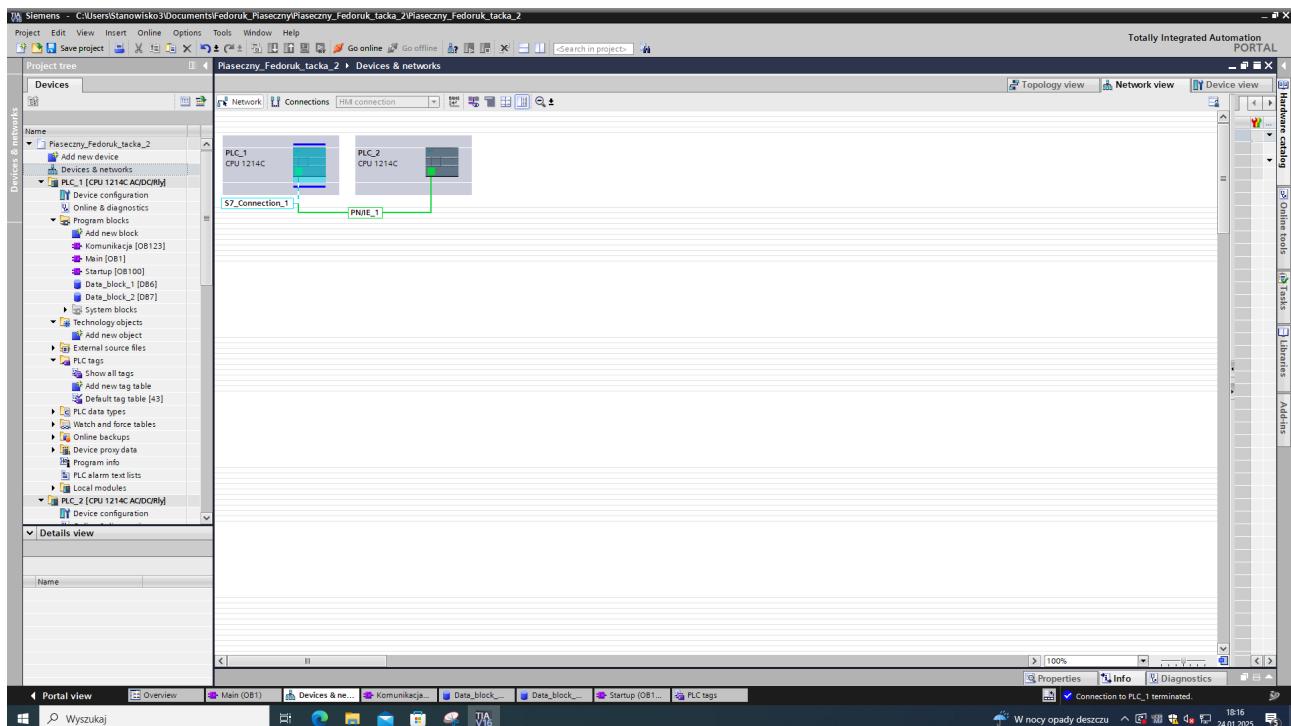


Figure 1: Konfiguracja sieciowa i połączenia sterownika Siemens S7-1200 w programie TIA Portal.

Sterowniki zostały przypisane do sieci lokalnej i otrzymały adresy IP:

- **PLC_1:** 192.168.22.147
- **PLC_2:** 192.168.22.146

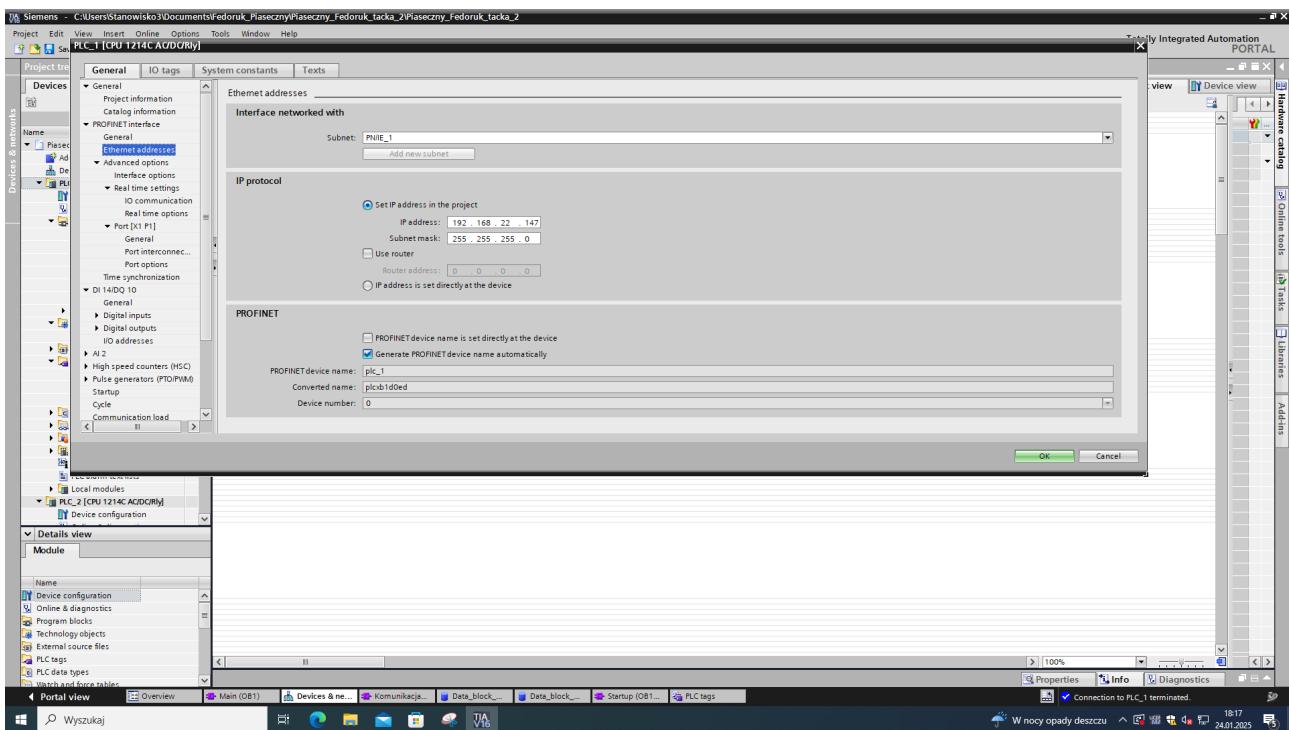


Figure 2: Konfiguracja IP dla PLC_1.

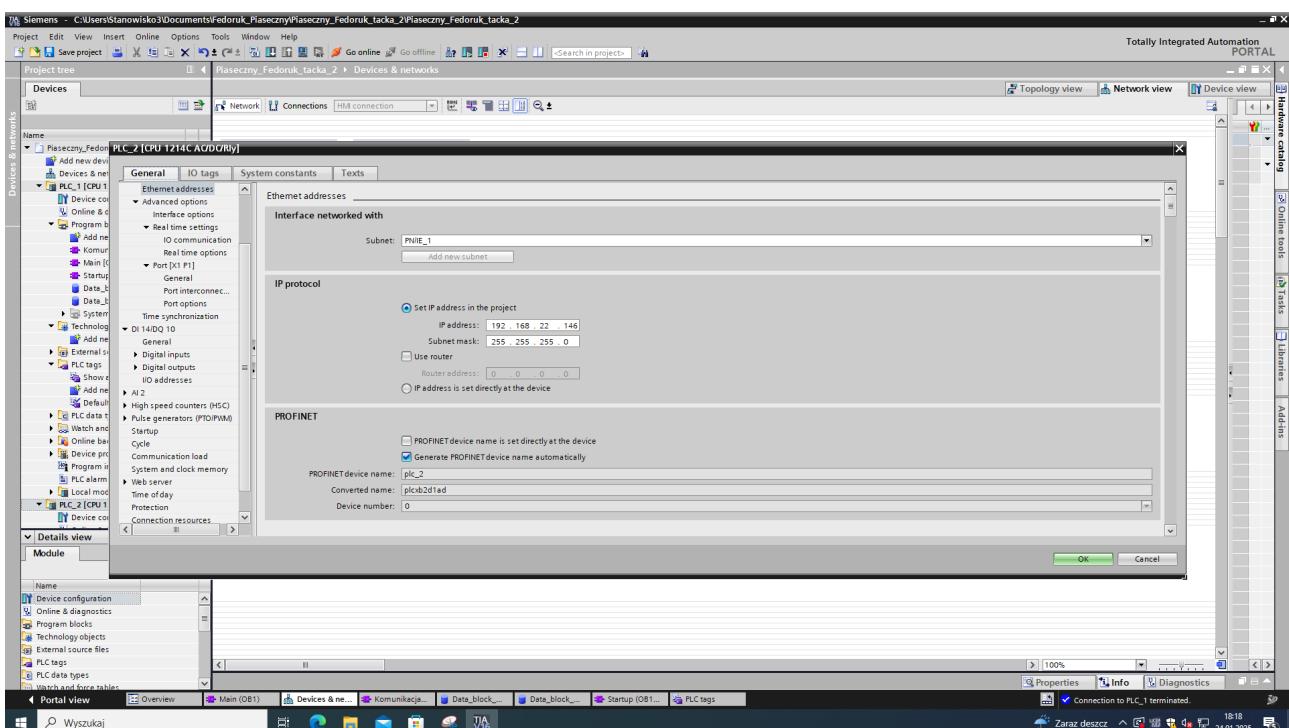
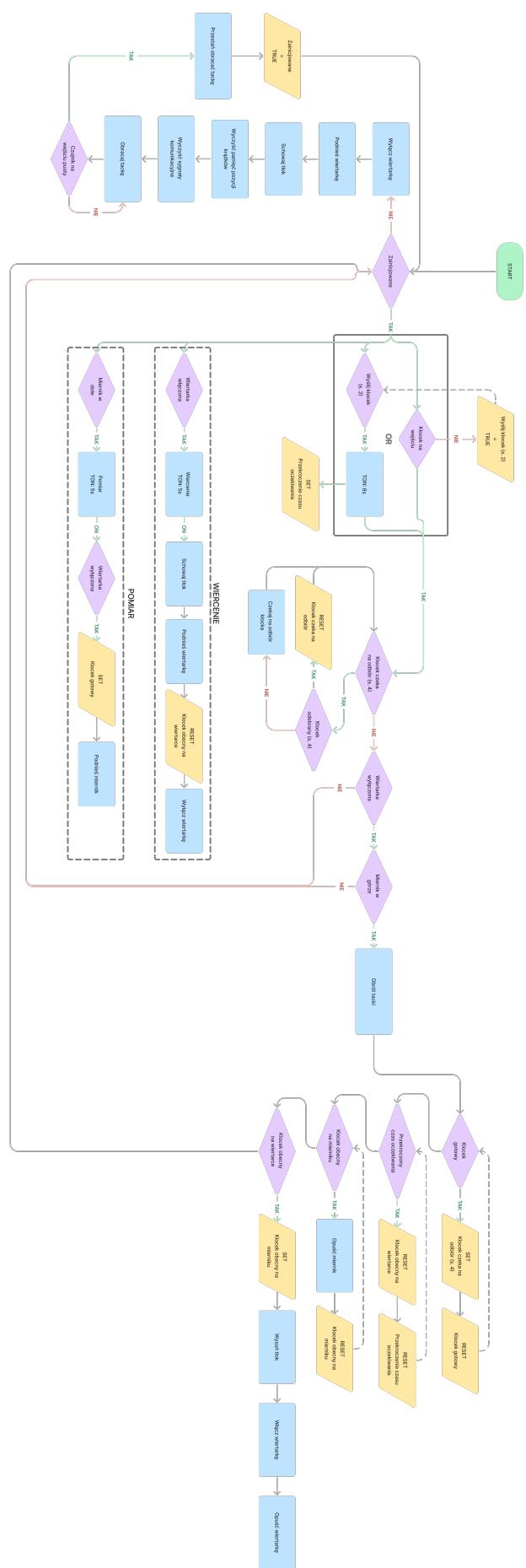


Figure 3: Konfiguracja IP dla PLC_2.

2 Schemat blokowy algorytmu sterowania

Schemat blokowy nie pokrywa się 1:1 z programem LAD, ze względu na niesekwencyjny charakter języka ladder, który ciężko jest przenieść w całości na schemat blokowy de facto logiki działania programu. Z tego też powodu niektóre procesy zostały na schemacie uproszczone, tak

aby przedstawać logiczny charakter aplikacji, niekoniecznie wchodząc w szczegóły implementacyjne. Prostokątne, niebieskie bloki odpowiadają działaniom lub procesom. Romby koloru fioletowego przedstawiają bloki decyzyjne (if). Z kolei romby koloru żółtego odpowiadają za przypisywanie zmiennych (ustawianie wartości zmiennych programowych). Dodatkowo sekcje odpowiadające za konkretne akcje składające się z kilku pod-akcji, takie jak wiercenie czy pomiar (czujnik grubości), zostały na schemacie oznaczone przerywanym obramowaniem. Obraz ze schematem ze względu na swój rozmiar musiał zostać obrócony i umieszczony w pozycji pionowej.

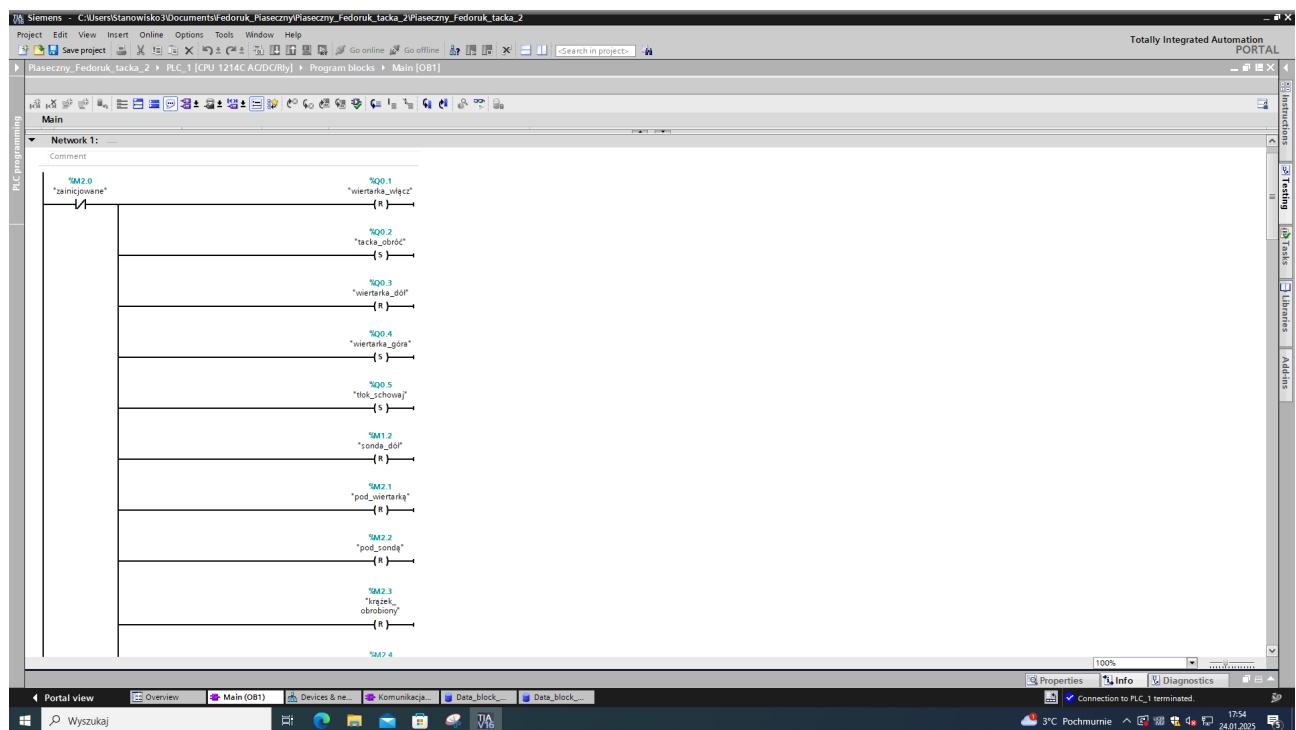


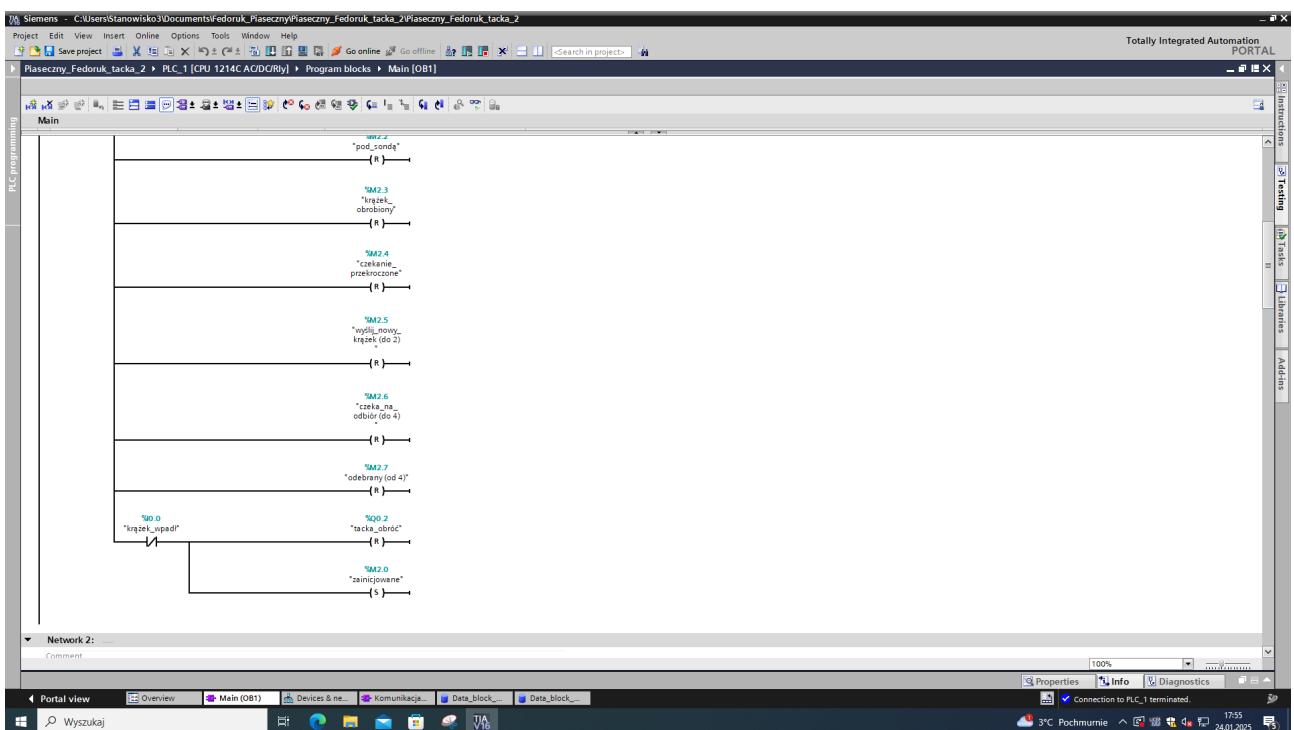
3 Program napisany w LD

Poniżej przedstawiono szczegółowy opis poszczególnych sieci sterowania w języku LD.

Network 1: Inicjacja systemu

Pierwszy network odpowiada za inicjalizację wszystkich urządzeń i ustawienie tacki w pozycji gotowości do pracy. Wyłącza on wiertarkę, podnosi ją, podobnie jak i miernik, chowa tłok, czyści adresy odpowiedzialne za przechowywanie informacji o pozycji klocków (czy są obecne przy poszczególnych urządzeniach) oraz przesuwa taczkę, póki czujnik wejściowy jest aktywny (wykrywamy, czy nad czujnikiem jest pusty otwór na klocek).

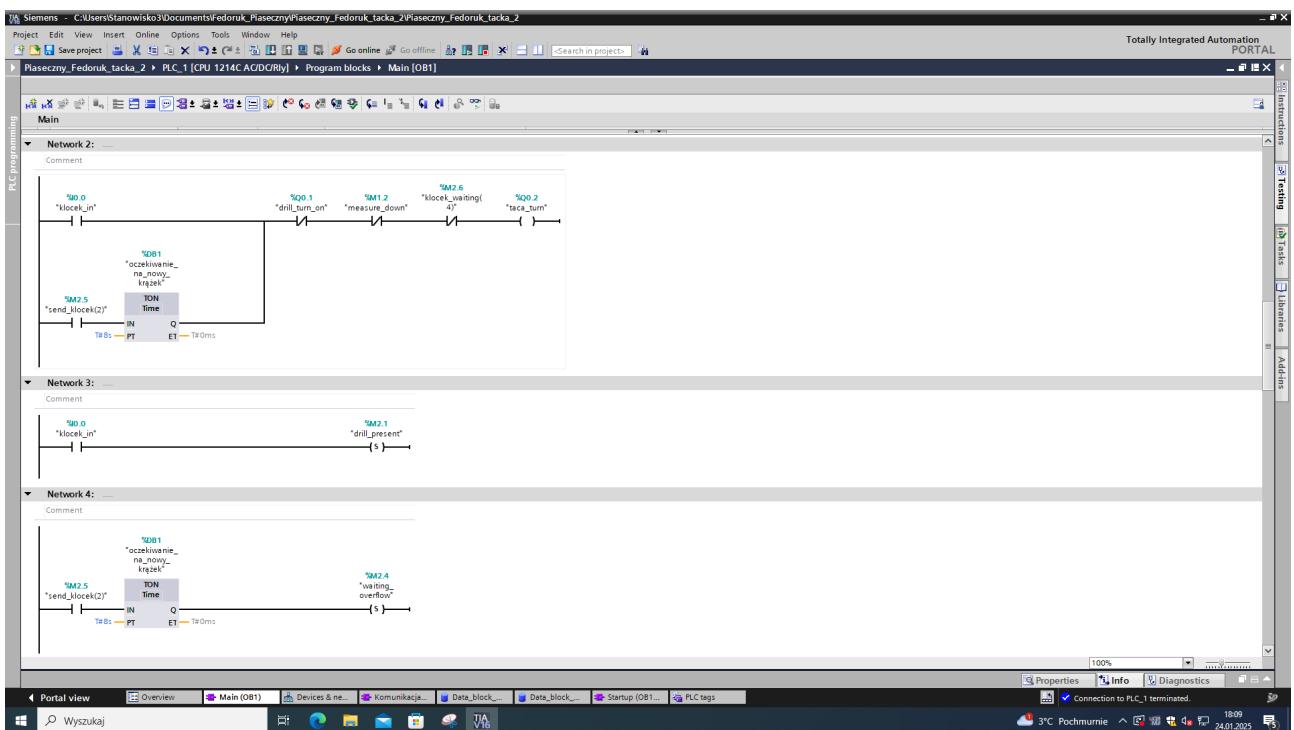




Network 2-4: Przygotowanie elementu do obróbki

Ta sekcja programu odpowiada za wprowadzenie elementu do systemu i jego przygotowanie do dalszej obróbki. Wykorzystuje następujące zmienne i bloki:

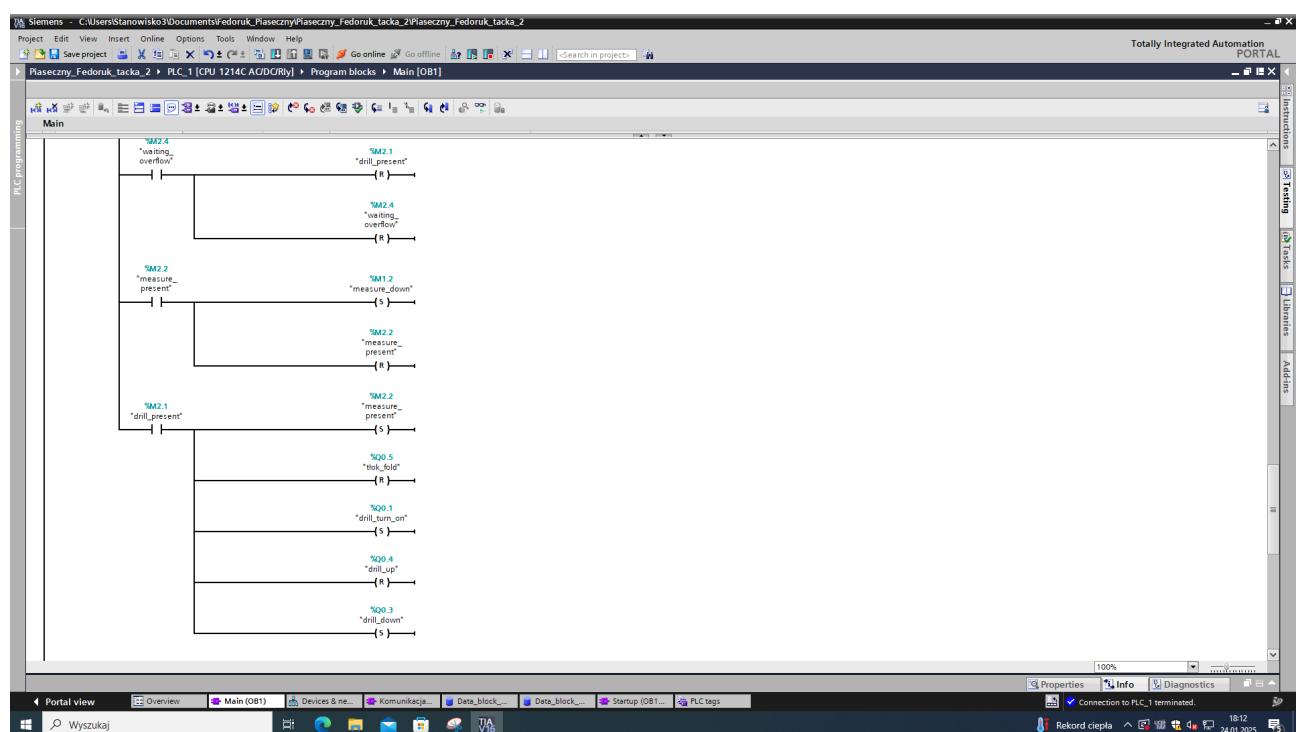
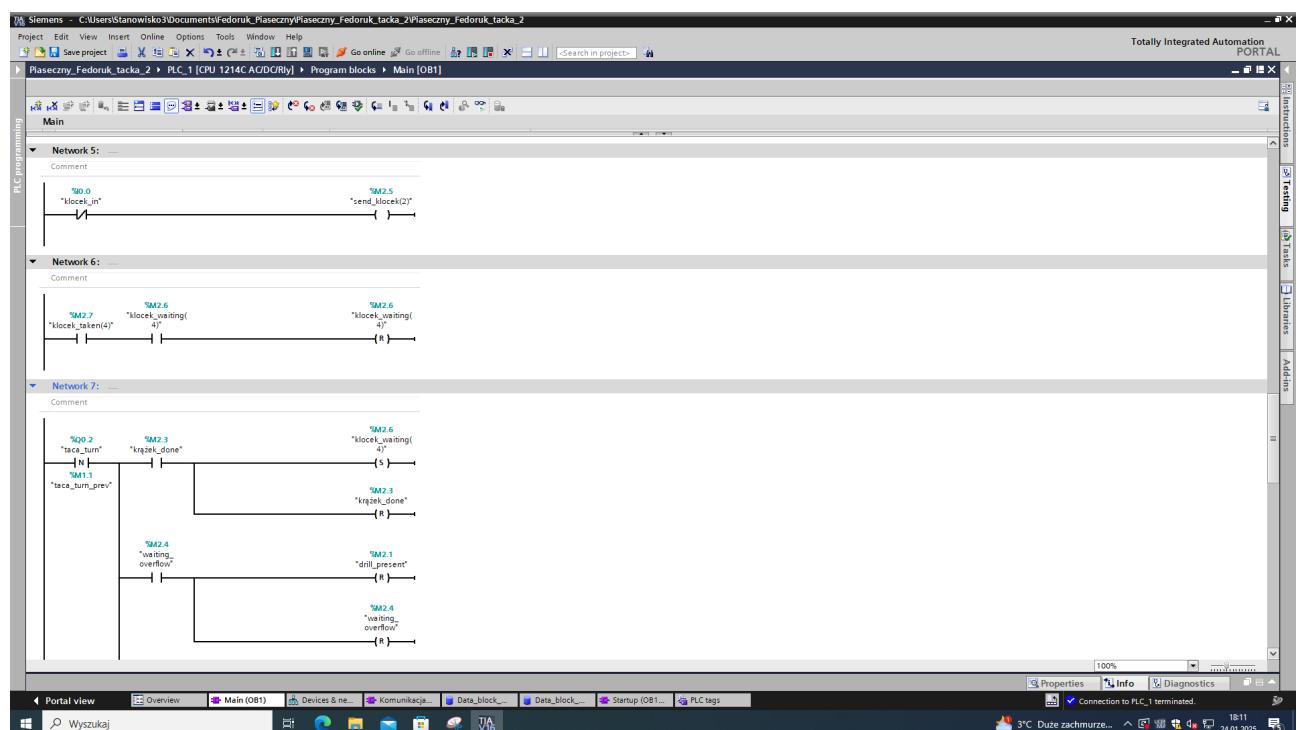
- Wejście:** %I0.0 – czujnik obecności elementu.
- Wyjście:** %Q0.1 – aktywacja mechanizmu transportującego element.
- Blok czasowy TON** – opóźnienie wprowadzenia elementu.



Network 5-7: Detekcja obecności i kontrola procesu

Sieć ta zarządza wykrywaniem obecności elementu oraz sterowaniem kolejnymi krokami obróbki. Kluczowe komponenty to:

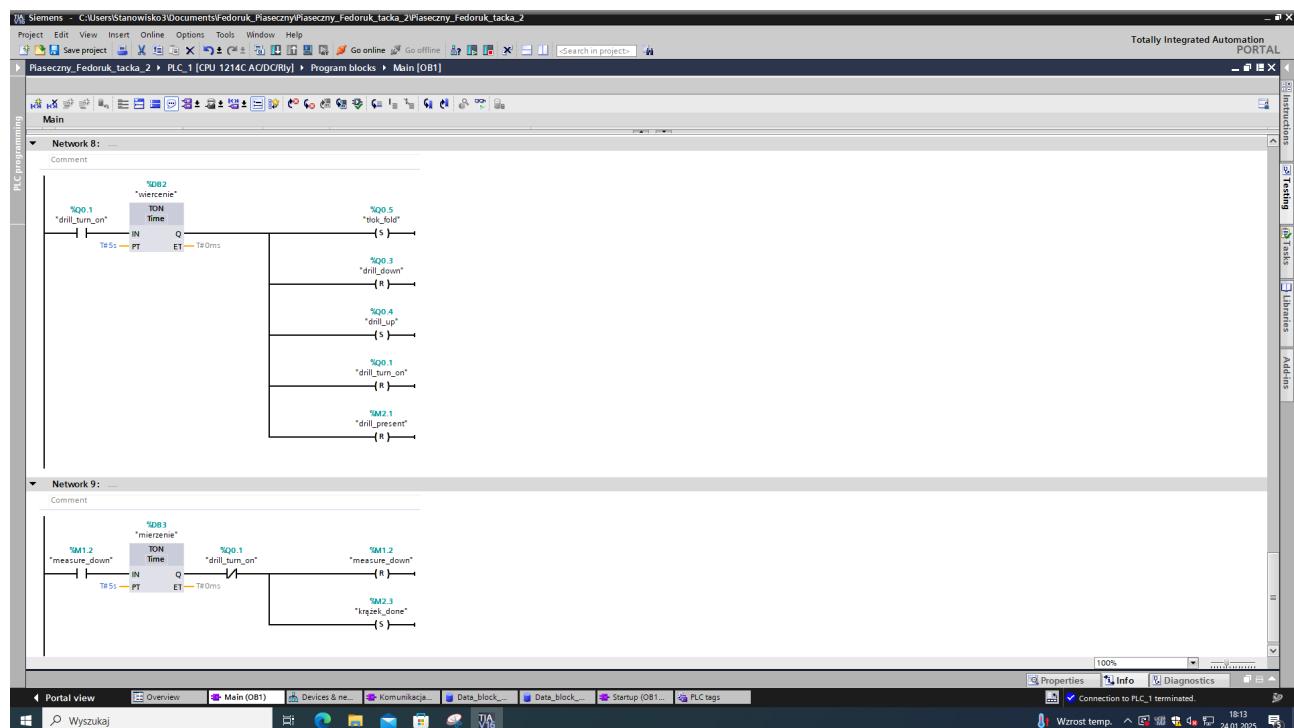
- Zmienne pamięciowe:** %M2.1 – stan gotowości do obróbki.
- Wyjście:** %Q0.2 – aktywacja wiertarki.
- Blok czasowy TON** – kontrola czasu wiercenia.



Network 8-9: Uruchomienie procesu i pomiar parametrów

Końcowy etap operacji obejmuje kontrolę wykonania procesu oraz pomiar parametrów.

- Zmienne:** %M2.3 – zakończenie obróbki.
- Wyjście:** %Q0.3 – dezaktywacja wiertarki.
- Blok czasowy TON** – zapisanie stanu operacji.



Program obsługujący komunikację między sterownikami

W celu zapewnienia poprawnej wymiany danych między sterownikami wykorzystano bloki danych oraz polecenia GET/PUT.

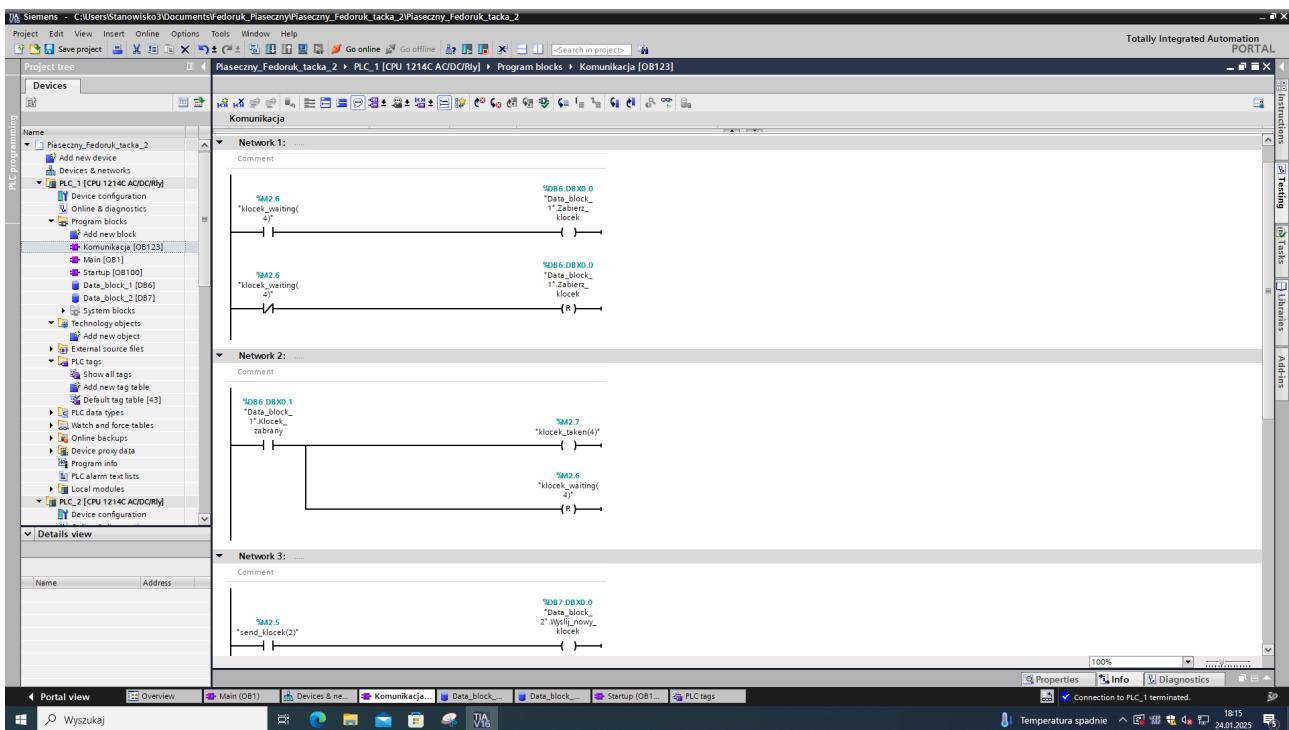


Figure 4: Program obsługujący komunikację - Networks 1-2.

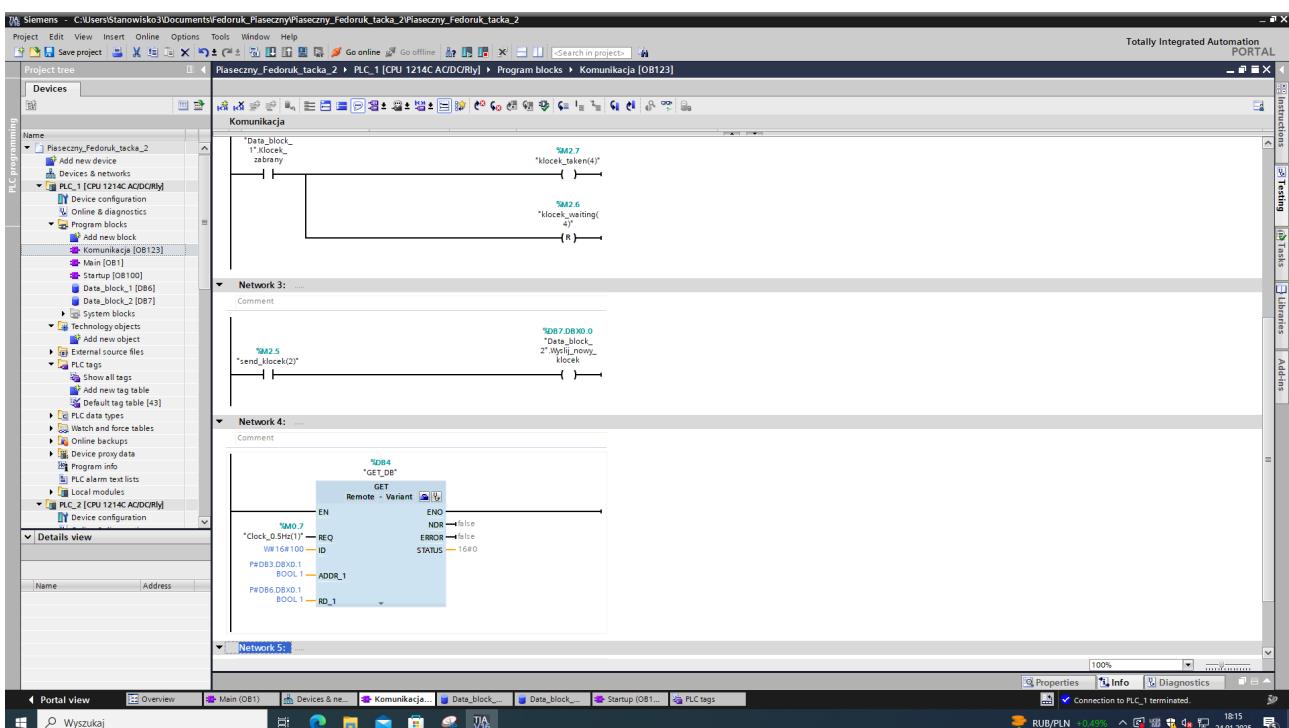


Figure 5: Program obsługujący komunikację - Networks 3-4.

Tabela zmiennych

Poniżej przedstawiono szczegółowy opis zmiennych używanych w programie.

Nazwa	Adres	Typ	Opis
klocek_in	%I0.0	Bool	Czujnik zbliżeniowy wykrywający obecność klocka (lub tacki przy obrocie) na wejściu do systemu.
drill_turn_on	%Q0.1	Bool	Wyjście aktywujące wiercenie.
taca_turn	%Q0.2	Bool	Sterowanie obrotom tacy.
drill_down	%Q0.3	Bool	Obniżenie wiertarki.
drill_up	%Q0.4	Bool	Podniesienie wiertarki.
tłok_fold	%Q0.5	Bool	Schowanie tłoku.
drill_present	%M2.1	Bool	Flaga informująca, że klocek jest na stanowisku wiercenia.
measure_present	%M2.2	Bool	Flaga informująca, że klocek jest na stanowisku pomiarowym.
krążek_done	%M2.3	Bool	Informacja, że klocek jest w pełni obrabiony.
waiting_overflow	%M2.4	Bool	Zmienna informująca o przekroczeniu czasu oczekiwania na nowy klocek.
send_klocek(2)	%M2.5	Bool	Informacja o gotowości na przyjęcie klocka (stanowisko 2).
klocek_waiting(4)	%M2.6	Bool	Informacja o klocku oczekującym na zabranie.
klocek_taken(4)	%M2.7	Bool	Informacja o zabraniu klocka z tacy.
taca_turn_prev	%M1.1	Bool	Zmienna pamiętająca poprzedni stan obrotu tacy.

Table 1: Tabela zmiennych sterownika S7-1200.

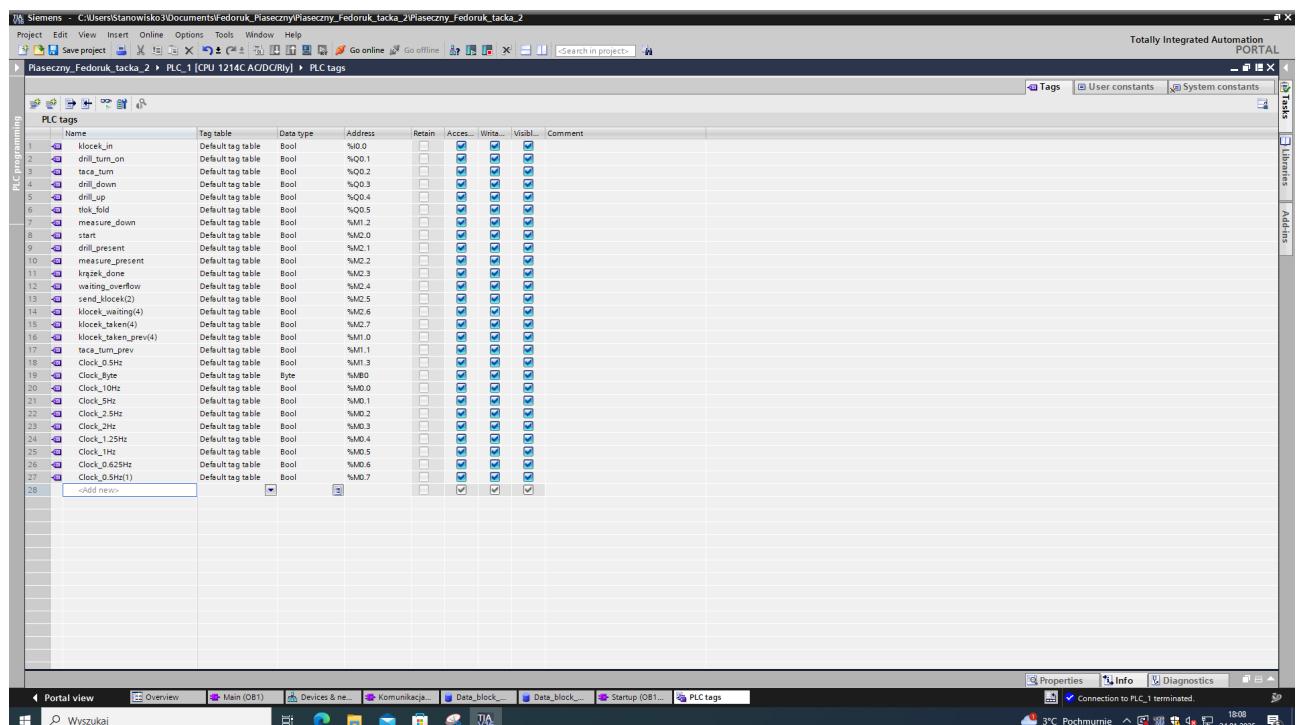


Figure 6: Widok tabeli zmiennych w programie TIA Portal.

Bloki danych używane w programie

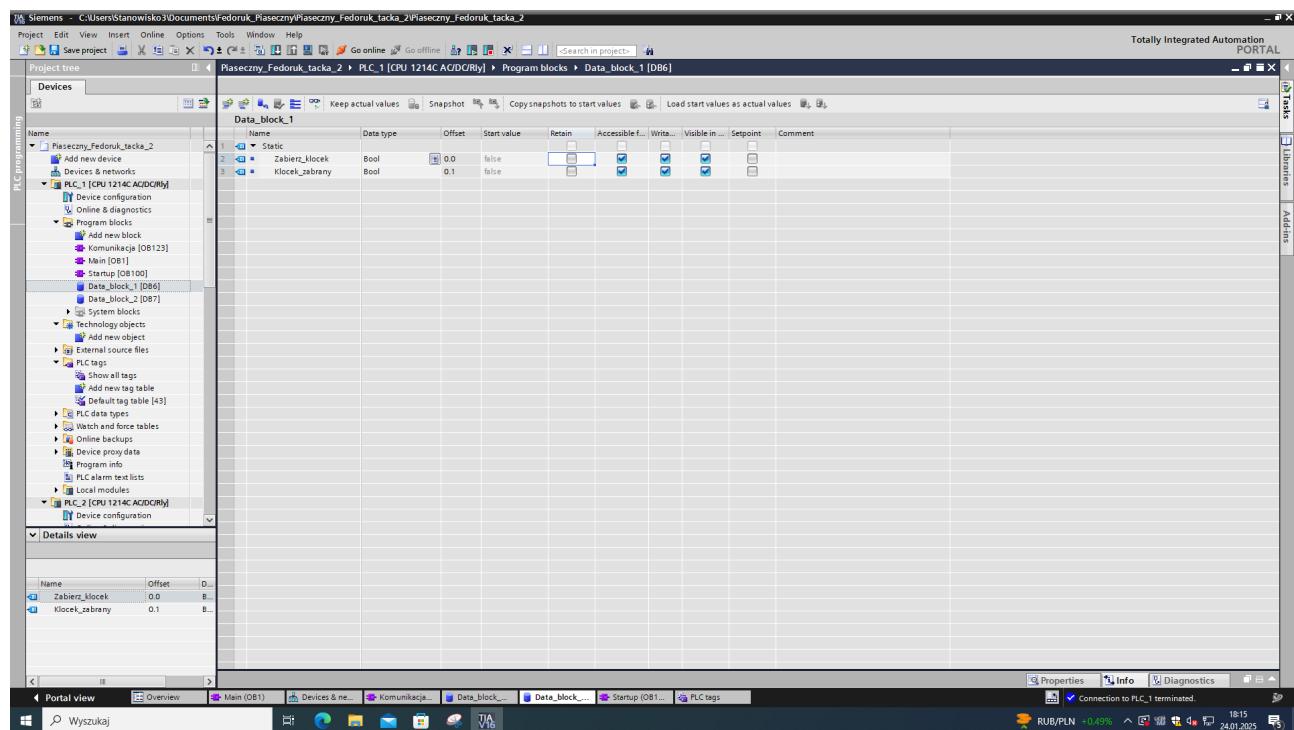


Figure 7: Blok danych DB1

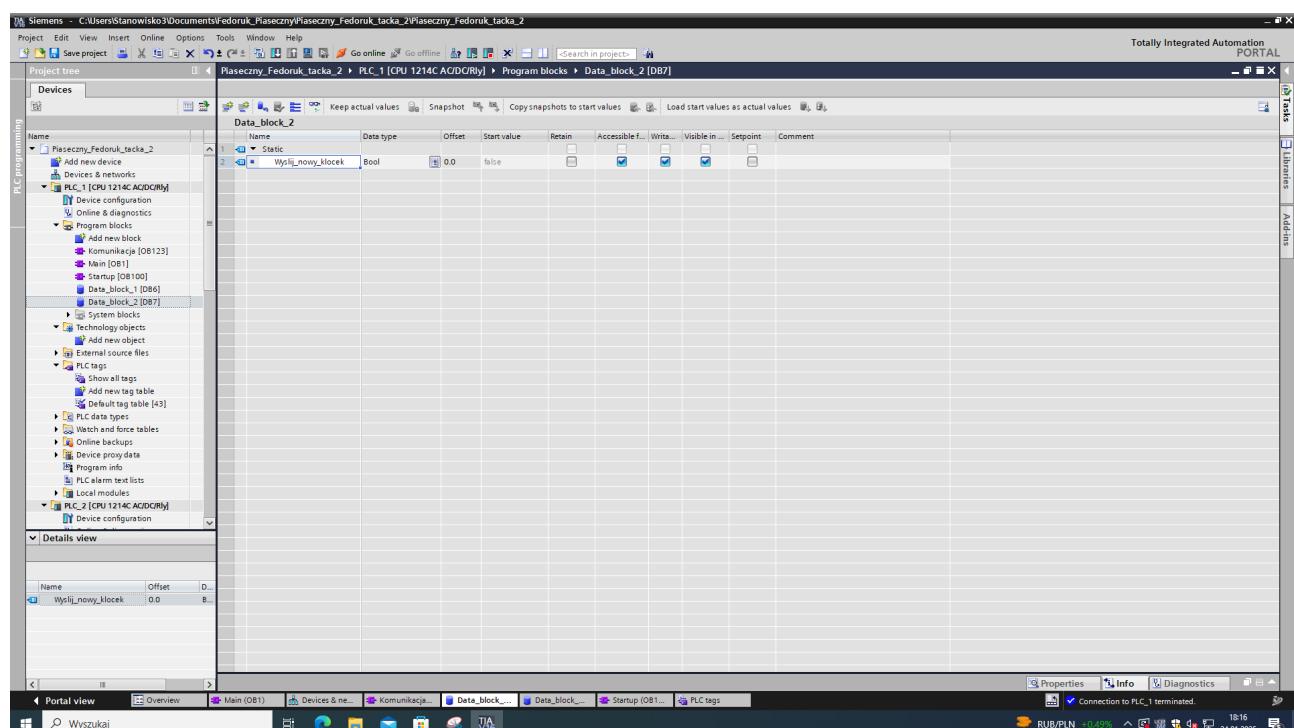


Figure 8: Blok danych DB2

4 Umiejscowienie czujników i urządzeń w rzeczywistym systemie

Położenie wiertarki, miernika grubości, czujnika zbliżeniowego [%I0.0] oraz taśmy podającej (stanowisko 2)

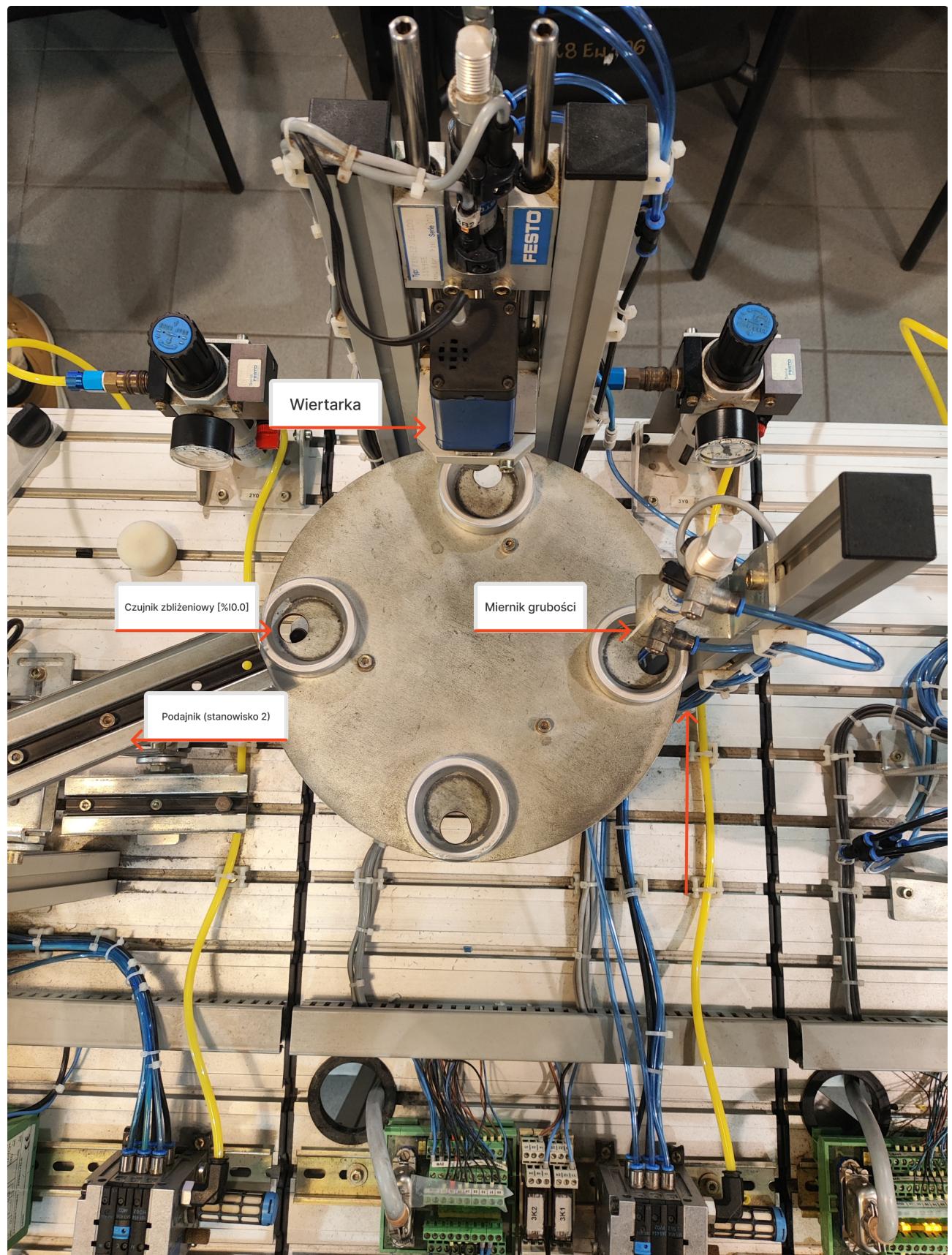


Figure 9: Widok stanowiska z góry

Wygląd i dokładne umiejscowienie czujników tacki

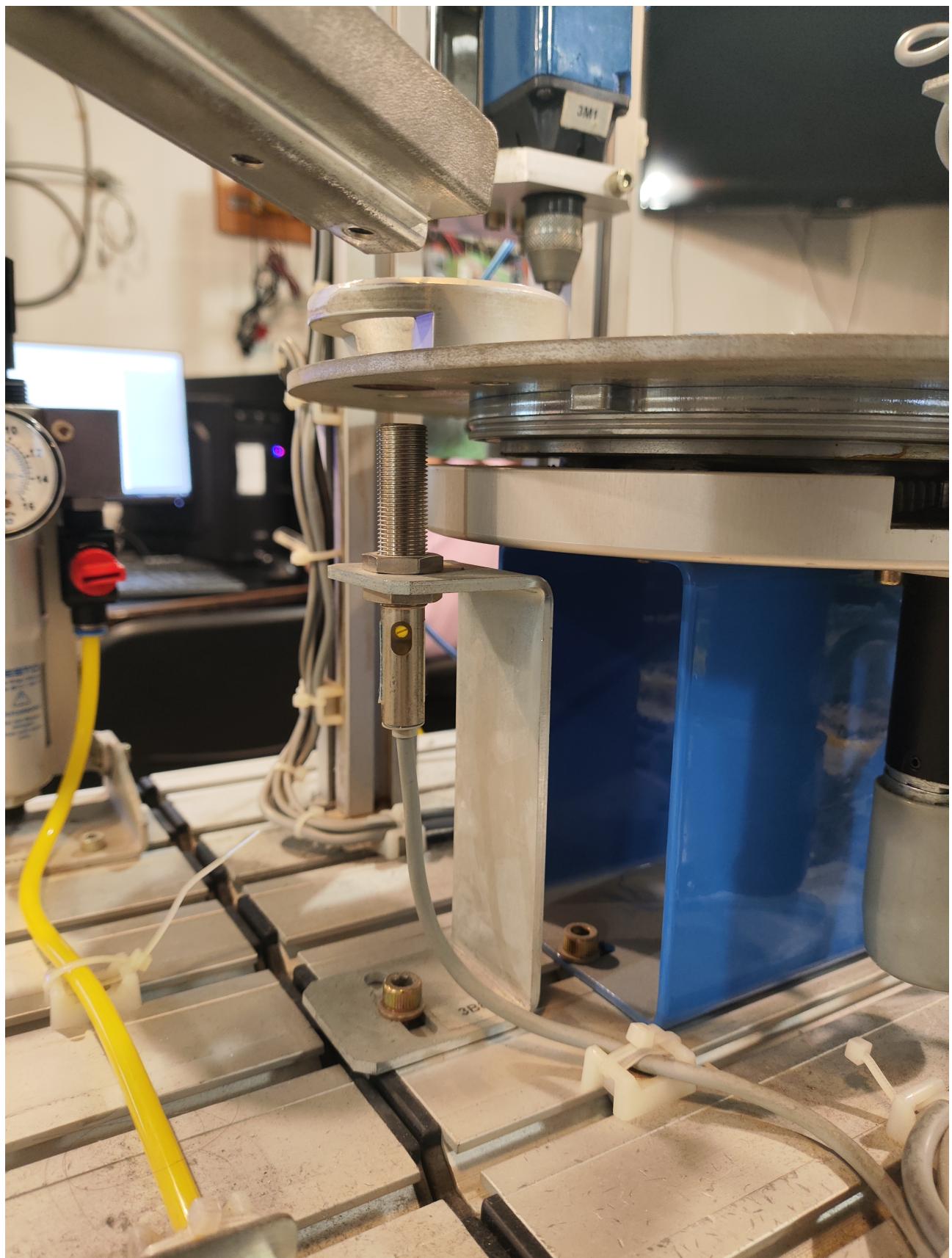


Figure 10: Czujnik zbliżeniowy na wejściu [%I0.0]

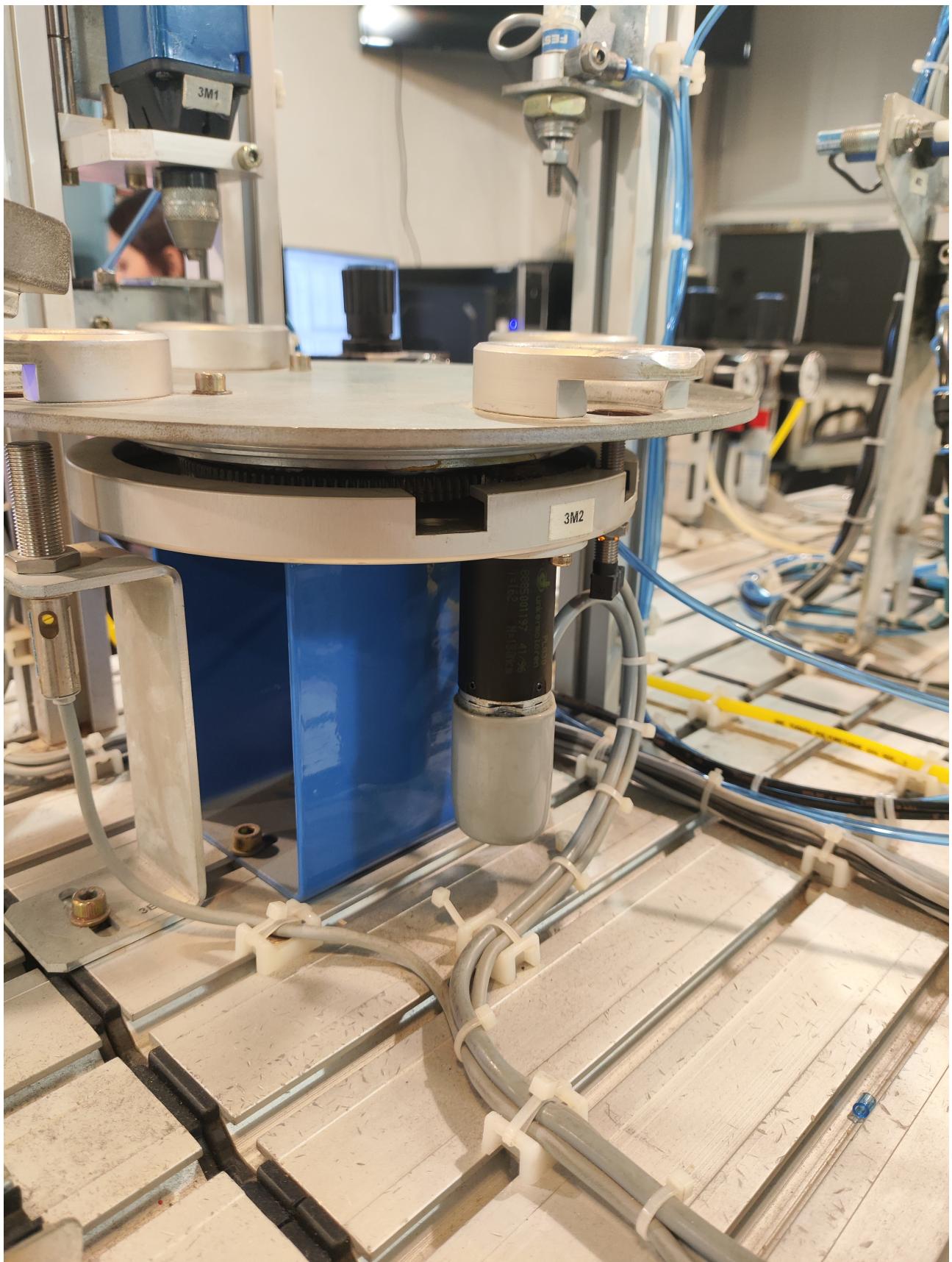


Figure 11: Czujnik zbliżeniowy wykrywający obrót tacki [%I0.1]

5 Komunikacja

Komunikacja między sterownikami została zrealizowana z użyciem bloków GET-PUT, co pozwoliło na prostą i łatwą w implementacji drogę na przesyłanie danych między sterownikami. Komunikacja ta polega na podmienianiu danych w tabelach zmiennych między sterownikami. Przy pomocy bloczku GET sterownik pobiera dane do swojej tabeli, natomiast blok PUT służy do wstawienia danych do sterownika partnerskiego. Aby komunikacja ta była możliwa, tabele na obu sterownikach muszą być takie same (typy, kolejność i adresy zmiennych).

Zalety komunikacji GET-PUT

Poza wspomnianą wyżej prostotą, zarówno w zrozumieniu, jak i implementacji, można także wspomnieć o możliwości przesyłu danych tylko w przypadku zaistnienia konkretnych warunków (np. po wykonaniu konkretnej operacji) co znacznie zmniejsza obciążenie sieci i sterowników. W naszym przypadku jednak nie miało to zastosowania, gdyż korzystaliśmy z cyklicznej wymiany danych na podstawie zegara systemowego o częstotliwości 0.5Hz.

Warto tutaj także powiedzieć o bardzo prostej konfiguracji połączenia do kilku sterowników na raz, gdzie wystarczy dodać sterownik partnerski w sieci i dodać do niego datablock z odpowiednimi polami, a nasz sterownik już będzie mógł zarówno wysyłać do niego dane, jak i je z niego pobierać. Możemy tak robić z wieloma datablockami i urządzeniami, stąd zarówno kilka sterowników może korzystać z danych udostępnianych przez nasz sterownik, jak i nasz sterownik z danych udostępnianych przez wiele sterowników w sieci.

Problemy komunikacji GET-PUT

Głównym problemem, jaki mógł wystąpić przy tego typu komunikacji, była dość niska częstotliwość przesyłu danych, co może prowadzić do niespójności między stanem na sterowniku a stanem faktycznym.

Schemat połączeń



Figure 12: Schemat połączenia sieciowego (komunikacja GET-PUT)