Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską

Politechnika Świętokrzyska
Wydział Elektrotechniki Automatyki i Informatyki

Kierunek studiów:
Elektrotechnika

Paweł Strączyński

Materiały dydaktyczne do przedmiotu

PRZEMYSŁOWE SIECI KOMPUTEROWE

opracowane w ramach realizacji Projektu
„**Dostosowanie kształcenia w Politechnice Świętokrzyskiej do potrzeb współczesnej gospodarki**”
FERS.01.05-IP.08-0234/23

Kielce, 2025

Spis treści

*Projekt „Dostosowanie kształcenia w Politechnice Świętokrzyskiej do potrzeb współczesnej gospodarki”
nr FERS.01.05-IP.08-0234/23*

1. Wprowadzenie

Przemysłowe sieci komputerowe to specjalistyczne systemy komunikacyjne, które zostały zaprojektowane do przesyłania danych w środowiskach przemysłowych. Sieci przemysłowe cechuje wysoka niezawodność, deterministyczność oraz odporność na zakłócenia. Protokoły przemysłowe umożliwiają komunikację pomiędzy różnymi urządzeniami automatyki, takimi jak sterowniki PLC, czujniki, panele HMI, systemy SCADA czy komputery przemysłowe. Sieci przemysłowe cechuje również specjalna warstwa sprzętowa. W odróżnieniu od typowych sieci biurowych, sieci przemysłowe są dostosowane do trudnych warunków pracy (np. wysoka temperatura, wibracje, zakłócenia elektromagnetyczne) i wspierają komunikację w czasie rzeczywistym.

Cechy charakterystyczne sieci przemysłowych:

- Deterministyczność – przewidywalne i gwarantowane czasy odpowiedzi.
- Odporność na zakłócenia – mechanizmy zabezpieczające transmisję danych.
- Długowieczność – projektowane na wiele lat eksploatacji.
- Bezpieczeństwo – ochrona przed awariami i cyberatakami.
- Topologie odporne na błędy – np. pierścienie redundancji
- Zgodność z normami przemysłowymi – np. IEC 61158, IEC 61784, ISA-95.

Cel stosowania przemysłowych sieci komputerowych:

1. Zwiększenie efektywności produkcji:
   - Umożliwiają szybki przepływ danych między urządzeniami sterującymi a wykonawczymi.
   - Skracają czas reakcji na zmiany w procesie technologicznym.
2. Integracja systemów automatyki:
   - Łączą różne poziomy zarządzania produkcją – od czujników po systemy MES/ERP.
   - Ułatwiają zarządzanie danymi i ich analizę w czasie rzeczywistym.
3. Zdalne sterowanie i monitoring:
   - Pozwalają na kontrolę procesów z poziomu SCADA lub przez Internet (IIoT).
   - Wspierają predykcyjne utrzymanie ruchu (predictive maintenance).
4. Obniżenie kosztów eksploatacji:
   - Redukują potrzebę ręcznego odczytu i sterowania.
   - Umożliwiają szybką diagnostykę i minimalizują przestoje.
5. Standaryzacja komunikacji:
   - Dzięki zastosowaniu otwartych protokołów (np. OPC UA, Modbus TCP) różne urządzenia mogą ze sobą współpracować.

2. Model OSI

2.1. Wprowadzenie do modelu OSI

Model OSI (ang. Open Systems Interconnection) to koncepcyjny, siedmiowarstwowy model stworzony przez Międzynarodową Organizację Normalizacyjną(ISO). Model ten opisuje sposób komunikacji w sieciach komputerowych. Choć często nie jest bezpośrednio implementowany w całości, stanowi uniwersalną ramę odniesienia dla projektowania i analizy protokołów sieciowych.

Warstwy modelu OSI:

- Warstwa fizyczna (Physical) – transmisja sygnału w medium fizycznym (kable, fale radiowe, złącza).
- Warstwa łącza danych (Data Link) – zapewnia niezawodny przesył ramek danych (adresacja MAC, wykrywanie błędów).
- Warstwa sieciowa (Network) – odpowiada za trasowanie pakietów (adresacja IP).
- Warstwa transportowa (Transport) – zapewnia niezawodność transmisji (TCP/UDP, kontrola błędów).
- Warstwa sesji (Session) – zarządzanie sesjami komunikacyjnymi (synchronizacja, inicjacja/kończenie sesji).
- Warstwa prezentacji (Presentation) – tłumaczenie danych (kodowanie, szyfrowanie, kompresja).
- Warstwa aplikacji (Application) – interakcja z aplikacjami użytkownika (np. SCADA, HMI, OPC UA).

2.2. Znaczenie modelu OSI w sieciach przemysłowych

W kontekście przemysłowym, model OSI pomaga w:

- analizie i doborze protokołów dla konkretnych zastosowań automatyki,
- zrozumieniu struktury komunikacji między urządzeniami sterującymi, wykonawczymi i nadzorczymi,
- diagnostyce problemów sieciowych – pozwala na precyzyjne zlokalizowanie błędu (np. w warstwie fizycznej lub aplikacyjnej).

Politechnika Świętokrzyska Kielce University of Technology
Projekt „Dostosowanie kształcenia w Politechnice Świętokrzyskiej do potrzeb współczesnej gospodarki” nr FERS.01.05-IP.08-0234/23

2.3. Protokoły przemysłowe a model OSI

Przemysłowe sieci nie zawsze wykorzystują wszystkie warstwy OSI – często są to uproszczone modele, dopasowane do konkretnych potrzeb (np. systemy real-time). W przypadku sieci przemysłowych model OSI jest stosowany przede wszystkim jako narzędzie koncepcyjne, które ułatwia projektowanie i analizę systemów komunikacyjnych. Pozwala on uporządkować poszczególne funkcje komunikacji w logiczne warstwy, dzięki czemu inżynierowie mogą lepiej dobierać technologie oraz diagnozować ewentualne problemy. W praktyce protokoły przemysłowe często łączą funkcje kilku warstw w jednym rozwiązaniu lub całkowicie pomijają niektóre z nich. Dzieje się tak głównie w celu uproszczenia komunikacji, zmniejszenia opóźnień i zwiększenia niezawodności działania w czasie rzeczywistym. Przykładowo, warstwy prezentacji i sesji są w wielu protokołach zintegrowane z warstwą aplikacji, a mechanizmy kontroli transmisji znajdują się w warstwach niższych.
Istotnym aspektem w środowiskach przemysłowych jest deterministyczny charakter transmisji. Oznacza to, że przesył danych musi odbywać się w ściśle określonym czasie, bez nieprzewidzianych opóźnień. Aby to zapewnić, w warstwie łącza danych często stosuje się specjalne mechanizmy priorytetyzacji ruchu lub rezerwacji pasma, a w warstwie fizycznej – media odporne na zakłócenia, takie jak skrętki ekranowane lub światłowody. Dzięki wykorzystaniu modelu OSI możliwe jest także łatwiejsze integrowanie urządzeń od różnych producentów. Standaryzacja interfejsów i formatów danych sprawia, że systemy automatyki mogą być rozbudowywane i modernizowane bez konieczności wymiany całej infrastruktury komunikacyjnej. To podejście nie tylko obniża koszty, ale również zwiększa elastyczność zakładu produkcyjnego, umożliwiając szybkie dostosowanie do zmieniających się wymagań procesów technologicznych. W tabeli poniżej przedstawiono opis jednego z najczęściej wykorzystywanych protokołów w przemyśle (Modbus TCP) za pomocą modelu OSI.

Tabela 1. Modbus TCP a OSI

Warstwa OSI	Modbus TCP
Aplikacji	Protokół Modbus
Prezentacji	-
Sesji	-
Transportu	TCP
Sieciowa	IP
Łącza danych	Ethernet (MAC)
Fizyczna	Medium: skrętka, światłowód

3. Architektury sprzętu sieciowego stosowane w przemyśle

3.1. Wprowadzenie

Architektura sprzętowa przemysłowej sieci komputerowej obejmuje zarówno fizyczną topologię połączeń, jak i urządzenia sieciowe, które umożliwiają wymianę danych między elementami systemu automatyki (np. PLC, HMI, SCADA, czujnikami i siłownikami).
W przeciwieństwie do sieci biurowych, sieci przemysłowe muszą być:

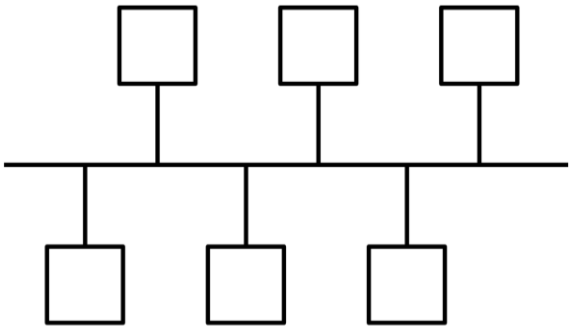
- deterministyczne (o przewidywalnych opóźnieniach),
- odporne na zakłócenia,
- często zbudowane redundantnie (dla niezawodności).

3.2. Typowe topologie sieci przemysłowych

3.2.1. Topologia magistrali (bus)

Opis: Wszystkie urządzenia podłączone do jednej linii transmisyjnej.
Zastosowanie: starsze systemy – Profibus, Modbus RTU.
Zalety: prostota, niskie koszty.
Wady: brak redundancji, awaria jednej linii = awaria całej sieci.

[Tekst alternatywny. Schemat sieci komputerowej o topologii magistrali. W centralnej części znajduje się pozioma linia (magistrala), do której pionowymi liniami podłączonych jest w sumie siedem urządzeń (przedstawionych jako kwadraty) – cztery nad magistralą i trzy pod nią. Linie pionowe symbolizują przewody łączące urządzenia z magistralą.]

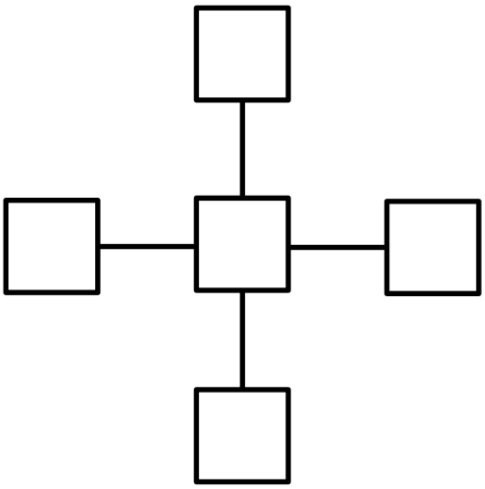


Rys. 1. Sieci o topologii magistrali

3.2.2. Topologia gwiazdy (star)

Opis: Wszystkie urządzenia są połączone do centralnego punktu (np. switcha)
Zastosowanie: Ethernet, PROFINET.
Zalety: łatwa diagnostyka, izolacja awarii.
Wady: awaria switcha = awaria całego segmentu.

[Tekst alternatywny Schemat sieci komputerowej o topologii gwiazdy. W centralnej części znajduje się kwadrat symbolizujący główny węzeł (np. koncentrator lub przełącznik). Od niego odchodzą cztery linie w kierunkach: góra, dół, lewo i prawo. Na końcach każdej linii znajdują się kwadraty symbolizujące urządzenia końcowe.]

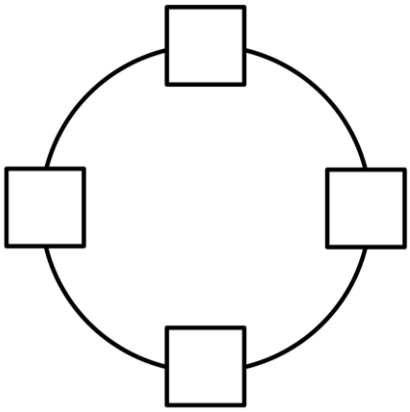


Rys. 2. Sieci o topologii gwiazdy

3.2.3. Topologia pierścienia (ring)

Opis: Urządzenia połączone w zamkniętą pętlę.
Zastosowanie: EtherNet/IP, , EtherCAT z redundancją.
Zalety: wysoka odporność na awarie (dzięki mechanizmom przełączania trasy).
Wady: konieczność obsługi mechanizmów redundancji.

[Tekst alternatywny. Schemat sieci komputerowej o topologii pierścienia. Cztery kwadraty symbolizujące urządzenia są połączone linią w kształcie zamkniętego okręgu. Każde urządzenie jest połączone z dwoma sąsiednimi, tworząc zamkniętą pętlę.]

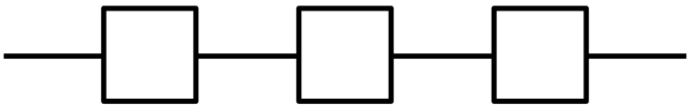


Rys. 3. Sieci o topologii pierścienia

3.2.4. Topologia liniowa (line/daisy chain)

Opis: Kolejne urządzenia połączone szeregowo.
Zastosowanie: EtherCAT, CANopen.
Zalety: minimalizacja okablowania.
Wady: awaria jednego połączenia odcina resztę sieci (bez redundancji).

[Tekst alternatywny. Schemat sieci komputerowej o topologii liniowej. Cztery kwadraty symbolizujące urządzenia są połączone pojedynczą poziomą linią. Urządzenia są ustawione w jednym rzędzie, a każdy z nich jest połączony z sąsiednim przewodem.]

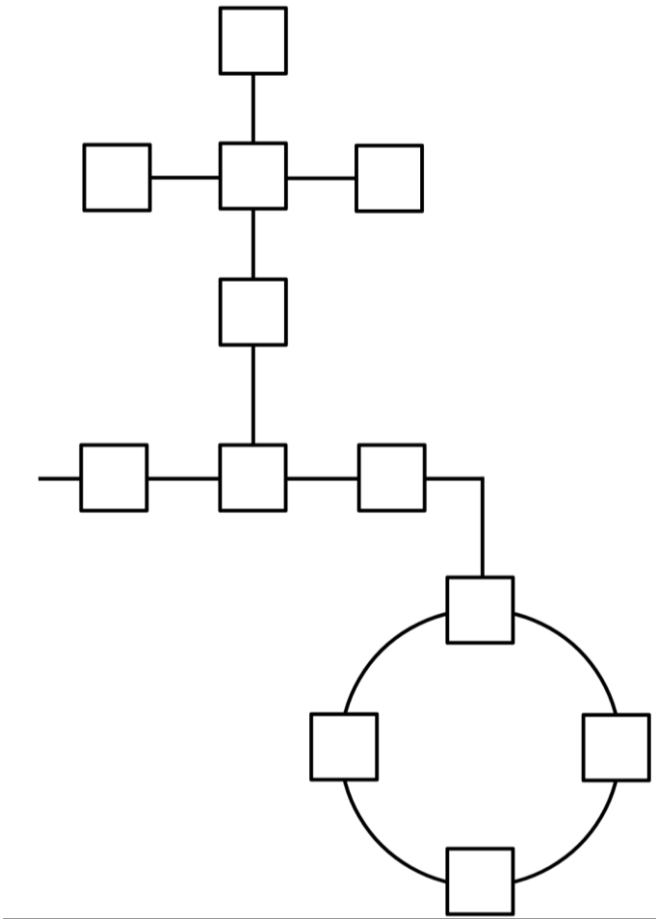


Rys. 4. Sieci o topologii liniowej

3.2.5. Topologia mieszana (hybrydowa)

Opis: Kombinacja powyższych – np. gwiazda z podpierścieniami.
Zastosowanie: rozbudowane systemy SCADA, rozproszone IO.

[Tekst alternatywny. Schemat sieci komputerowej łączącej różne topologie. W górnej części znajduje się układ o topologii gwiazdy, w którym centralny kwadrat łączy się z czterema urządzeniami. Od dolnego urządzenia tej gwiazdy odchodzi połączenie do kolejnego segmentu sieci o topologii liniowej, w której trzy urządzenia są połączone w szeregu. Od środkowego urządzenia w topologii liniowej prowadzi połączenie do układu o topologii pierścienia, w którym cztery urządzenia są połączone w zamkniętą pętlę.]



Rys. 5. Sieci o topologii mieszanej

3.3. Urządzenia w sieciach przemysłowych

W sieciach przemysłowych pracuje szereg urządzeń o różnych funkcjach, które wspólnie tworzą zintegrowany system automatyki. W przeciwieństwie do klasycznych sieci IT, w których główną rolę odgrywają komputery i serwery, w sieciach przemysłowych podstawą są urządzenia takie jak sterowniki PLC, moduły wejść/wyjść, panele operatorskie (HMI), serwonapędy, czujniki, systemy SCADA oraz urządzenia infrastruktury sieciowej, takie jak switche, routery i bramy komunikacyjne. Każde z tych urządzeń pełni określoną rolę: jedne odpowiadają za przetwarzanie i sterowanie (np. PLC), inne za komunikację i wizualizację danych (np. HMI, SCADA), a jeszcze inne za wykonanie operacji fizycznych (np. napędy, siłowniki). Łączy je wspólna cecha: są połączone w sieć komunikacyjną, która zapewnia wymianę danych w czasie rzeczywistym, deterministyczność działania oraz wysoką niezawodność — kluczową w aplikacjach przemysłowych.

Tabela 2. Typowe urządzenia wchodzące w skład współczesnej sieci przemysłowej

Kategoria	Przykładowe urządzenia
Sterujące	PLC, kontrolery motion
Wykonawcze	Serwonapędy, falowniki, zawory, przekaźniki
Interfejsy operatora	Panele HMI, terminale operatorskie
Nadrzędne	SCADA, komputer PC
Sieciowe	Switche, routery, bramy (gateways), access pointy
Pomiarowe	Moduły I/O, czujniki z komunikacją cyfrową

4. Komunikacja szeregowa w sieciach przemysłowych

4.1. Wprowadzenie

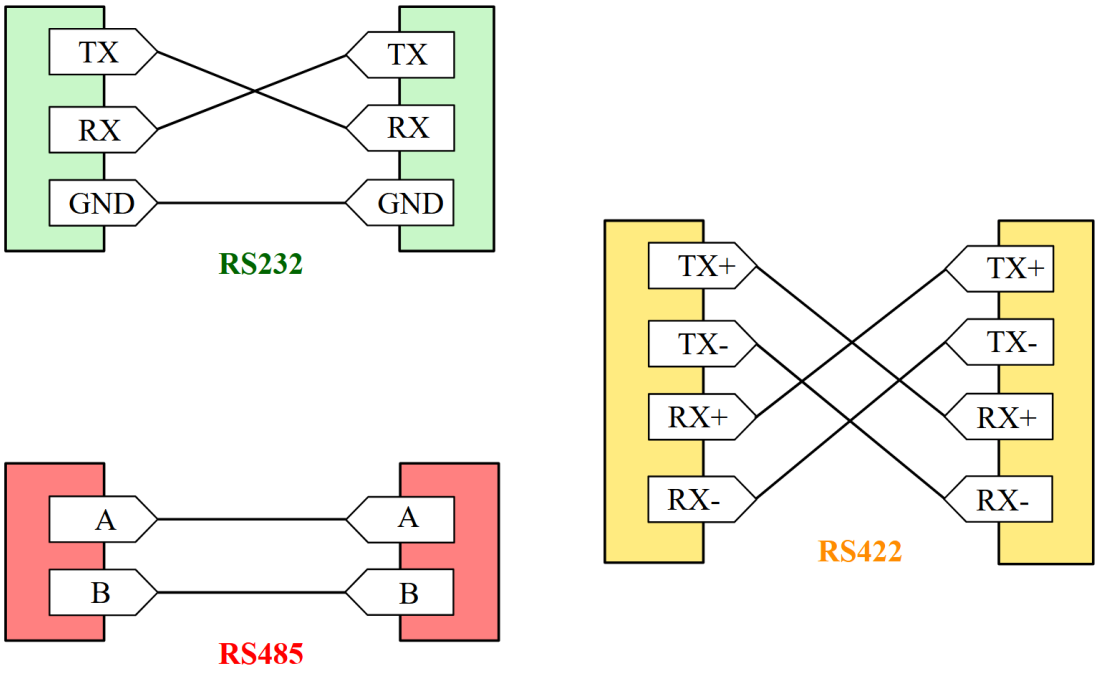
Komunikacja szeregowa to jedna z najstarszych, a jednocześnie wciąż szeroko stosowanych metod transmisji danych w automatyce przemysłowej. Polega na przesyłaniu informacji bit po bicie przez jeden kanał transmisyjny (przewód lub łącze bezprzewodowe). Mimo pojawienia się nowoczesnych sieci Ethernetu przemysłowego, interfejsy szeregowe są nadal używane ze względu na prostotę, niezawodność, niski koszt implementacji oraz odporność na zakłócenia.
Do najpopularniejszych standardów komunikacji szeregowej w przemyśle należą:

- **RS-232** – połączenia punkt–punkt, krótkie dystanse,
- **RS-422** – połączenia punkt–punkt o dużym zasięgu,
- **RS-485** – komunikacja wielopunktowa w topologii magistrali,
- **UART** – sprzętowy interfejs do obsługi transmisji szeregowej (często wewnątrz urządzeń).

Tabela 3. Standardy komunikacji szeregowej - parametry

Parametr	RS-232	RS-422	RS-485
Typ podłączenia	Jeden do jednego	Różnicowy	Różnicowy
Topologia	Punkt–punkt	1 do wielu	1 do wielu
Zasięg	do 15 m	do 1200 m	do 1200 m
Maksymalna prędkość	921,6 kbps	921,6 kbps	921,6 kbps
Tryb transmisji	Full-duplex	Full-duplex	Half-duplex
Stan wysoki H (1)	<-5V, -15V>	A - B < -0,2V	A - B < -0,2V
Stan niski L (0)	<5V, 15V>	A - B > +0,2V	A - B > +0,2V

[Tekst alternatywny. Ilustracja przedstawia schematy połączeń dla trzech standardów interfejsów szeregowych: RS232, RS485 i RS422. RS232 (zielone tło) – trzy przewody oznaczone jako TX, RX i GND; TX jest połączony z RX drugiego złącza i odwrotnie, GND połączone bezpośrednio. RS485 (czerwone tło) – dwa przewody oznaczone A i B; przewód A jest połączony z A drugiego złącza, a przewód B z B. RS422 (żółte tło) – cztery przewody oznaczone TX+, TX-, RX+, RX-; TX+ połączony z RX+ drugiego złącza, TX- z RX-, RX+ z TX+, RX- z TX-.]



Rys. 5. Schematy podłączeń urządzeń w sieciach szeregowych

4.2. Budowa klasycznego datagramu w transmisji szeregowej (asynchronicznej)

W komunikacji asynchronicznej dane przesyłane są w postaci kolejnych ramek (datagramów) zawierających ściśle określone elementy. Dzięki temu odbiornik jest w stanie rozpoznać początek i koniec transmisji, a także sprawdzić poprawność przesłanych danych.

4.2.1. Bitrate (prędkość transmisji)

Bitrate określa liczbę bitów przesyłanych w ciągu sekundy (*bps – bits per second).* Popularne wartości w przemyśle to np. 9600, 19200, 115200 bps.

4.2.2. Struktura ramki

Bit startu

- Wartość logiczna 0 (niski poziom napięcia).
- Informuje odbiornik o początku transmisji.
- Czas trwania bitu startu jest równy czasowi trwania pojedynczego bitu danych (1/bitrate).

Bity danych

- Najczęściej 7 lub 8 bitów (rzadziej 5, 6 lub 9 w systemach specjalnych).
- Wysyłane od najmłodszego bitu (LSB) do najstarszego bitu (MSB).

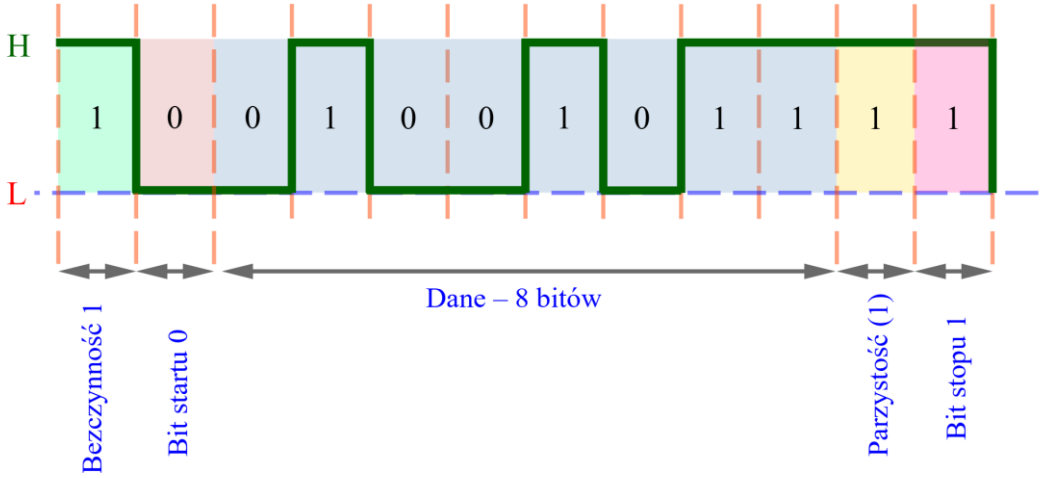
Bit parzystości (opcjonalny)

- Służy do prostej kontroli błędów.
- Można wyróżnić następujące typy:
  - Even (parzysta) – liczba jedynek w ramce (bity danych + parzystości) jest parzysta.
  - Odd (nieparzysta) – liczba jedynek jest nieparzysta.
  - None (brak) – brak kontroli parzystości.

Bity stop

- Wartość logiczna 1 (wysoki poziom napięcia),
- Najczęściej 1, 1,5 lub 2 bity stopu.
- Zapewniają przerwę przed rozpoczęciem kolejnej ramki.

[Tekst alternatywny. Ilustracja przedstawia przykład ramki transmisji asynchronicznej w systemie szeregowym. Oś pionowa oznacza poziomy sygnału: H (wysoki) oraz L (niski). Sygnał zaczyna się od stanu bezczynności (logiczna 1), następnie pojawia się bit startu (0). Kolejne 8 bitów danych to: 1, 0, 0, 1, 0, 0, 1, 1 (zaznaczone na niebiesko). Po nich występuje bit parzystości (1, zaznaczony na żółto), a na końcu bit stopu (1, zaznaczony na różowo). Granice bitów są wyznaczone pionowymi przerywanymi liniami.]



Rys. 5. Ramka komunikacji szeregowej

4.2.3. Parametry transmisji

Typowe ustawienia są określane w formacie:

[liczba bitów danych] – [parzystość] – [liczba bitów stopu]

Przykłady:

- 8-N-1 → 8 bitów danych, brak parzystości, 1 bit stopu
- 7-E-1 → 7 bitów danych, parzystość parzysta, 1 bit stopu
- 8-O-2 → 8 bitów danych, parzystość nieparzysta, 2 bity stopu

5. Protokół Modbus

5.1. Wprowadzenie i historia

Modbus to jeden z najstarszych i najbardziej rozpowszechnionych protokołów komunikacyjnych w automatyce przemysłowej. Został opracowany w 1979 roku przez firmę Modicon (obecnie Schneider Electric) jako prosty sposób komunikacji między sterownikiem PLC a urządzeniami zewnętrznymi. Dzięki otwartemu standardowi i łatwej implementacji, Modbus stał się szeroko stosowanym protokołem w systemach automatyki, m.in. do komunikacji z czujnikami, napędami, modułami I/O, licznikami i systemami SCADA.

Projekt „Dostosowanie kształcenia w Politechnice Świętokrzyskiej do potrzeb współczesnej gospodarki”
nr FERS.01.05-IP.08-0234/23

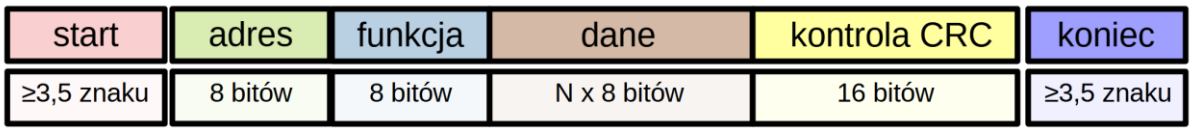
5.2. Warianty protokołu Modbus

Modbus występuje w kilku wersjach, które różnią się medium transmisyjnym i strukturą danych:

Modbus RTU (Real Time Unit)

- Działa na magistrali RS-485 (lub RS-232).
- Ramka zaczyna i kończy się „ciszą na linii" trwającą co najmniej tyle ile czas przesłania 3,5 znaku.
- Dane są przesyłane w formie binarnej.
- Komunikacja master-slave (jednostka nadrzędna pyta – podrzędne odpowiadają).
- Często wykorzystywany w prostych systemach lokalnych.
- Poprawność ramki zabezpieczona przez CRC.

[Tekst alternatywny. Schemat przedstawia strukturę ramki komunikacyjnej. Ramka składa się z sześciu pól: start (różowy) – co najmniej 3,5 znaku, adres (zielony) – 8 bitów, funkcja (niebieski) – 8 bitów, dane (brązowy) – N × 8 bitów, kontrola CRC (żółty) – 16 bitów, koniec (fioletowy) – co najmniej 3,5 znaku.]

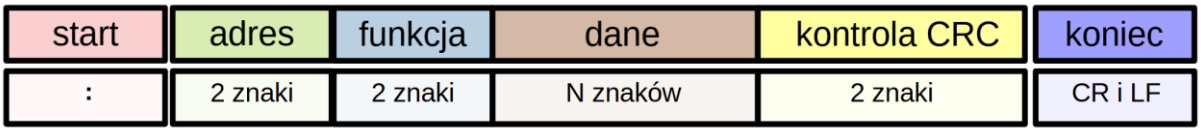


Rys. 5. Struktura ramki Modbus RTU

Modbus ASCII

- Działa również na magistrali RS-485/RS-232
- Dane są przesyłane jako ciągi znaków ASCII (czytelne dla człowieka).
- Ramka zaczyna się znakiem dwukropka : (0x3A) a kończy znakami powrotu karetki CR (0x0D) oraz nowej linii LF (0x0A).
- Wolniejszy od RTU, rzadziej stosowany.
- Także master-slave.
- Poprawność ramki zabezpieczona przez LRC.

[Tekst alternatywny] Schemat przedstawia strukturę ramki komunikacyjnej. Ramka składa się z sześciu pól: start (różowy) – znak „:", adres (zielony) – 2 znaki, funkcja (niebieski) – 2 znaki, dane (brązowy) – N znaków, kontrola CRC (żółty) – 2 znaki, koniec (fioletowy) – znaki CR i LF.]



Rys. 6. Struktura ramki Modbus ASCII

Modbus TCP/IP

- Wersja działająca w sieci Ethernet, oparta na protokole TCP/IP (port 502).
- Komunikacja klient-serwer (client-server), zamiast klasycznego master-slave.
- Umożliwia komunikację w sieciach lokalnych i zdalnych (np. SCADA ↔ PLC).

Tabela 4. Porównanie wariantów Modbus

Cecha	Modbus RTU	Modbus ASCII	Modbus TCP
Medium transmisji	RS-485/RS-232	RS-485/RS-232	Ethernet (TCP/IP)
Format danych	binarny	ASCII	binarny
Szybkość	wysoka	niska	wysoka
Tryb komunikacji	master-slave	master-slave	client-server
Kodowanie błędów	CRC16	LRC	CRC w warstwie TCP

5.3. Funkcje i typu pamięci

Protokół Modbus organizuje wymianę danych w oparciu o zdefiniowaną tablice pamięci urządzenia (np. modułu I/O, serwonapędu). Obszary pamięci są podzielone według rodzaju danych i możliwości dostępu (tylko odczyt lub odczyt/zapis) – tabela 3. Dostęp do każdego z obszaru pamięci możliwy jest za pomocą funkcji o określonym kodzie. Podstawowe kody funkcji protokołu przedstawiono w tabeli 4.

Tabela 5. Typy pamięci w Modbus

Typ pamięci	Zakres adresów	Rodzaj dostępu	Typ danych
Coils	00001-09999	R/W	1 bit
Discrete Inputs	10001-19999	RO	1 bit
Input Registers	30001-39999	RO	16 bitów
Holding Registers	40001-49999	R/W	16 bitów

*Projekt „Dostosowanie kształcenia w Politechnice Świętokrzyskiej do potrzeb współczesnej gospodarki”
nr FERS.01.05-IP.08-0234/23*

Tabela 6. Funkcje w Modbus

Kod funkcji	Nazwa funkcji	Obszar pamięci
01	Read Coils	Coils
02	Read Discrete Inputs	Discrete Inputs
03	Read Holding Registers	Holding Registers
04	Read Input Registers	Input Registers
05	Write Single Coil	Coils
06	Write Single Register	Holding Registers
15	Write Multiple Coils	Coils
16	Write Multiple Registers	Holding Registers

5.4. Przykłady praktyczne

5.4.1. Modbus RTU – odczyt rejestrów i analiza ramek

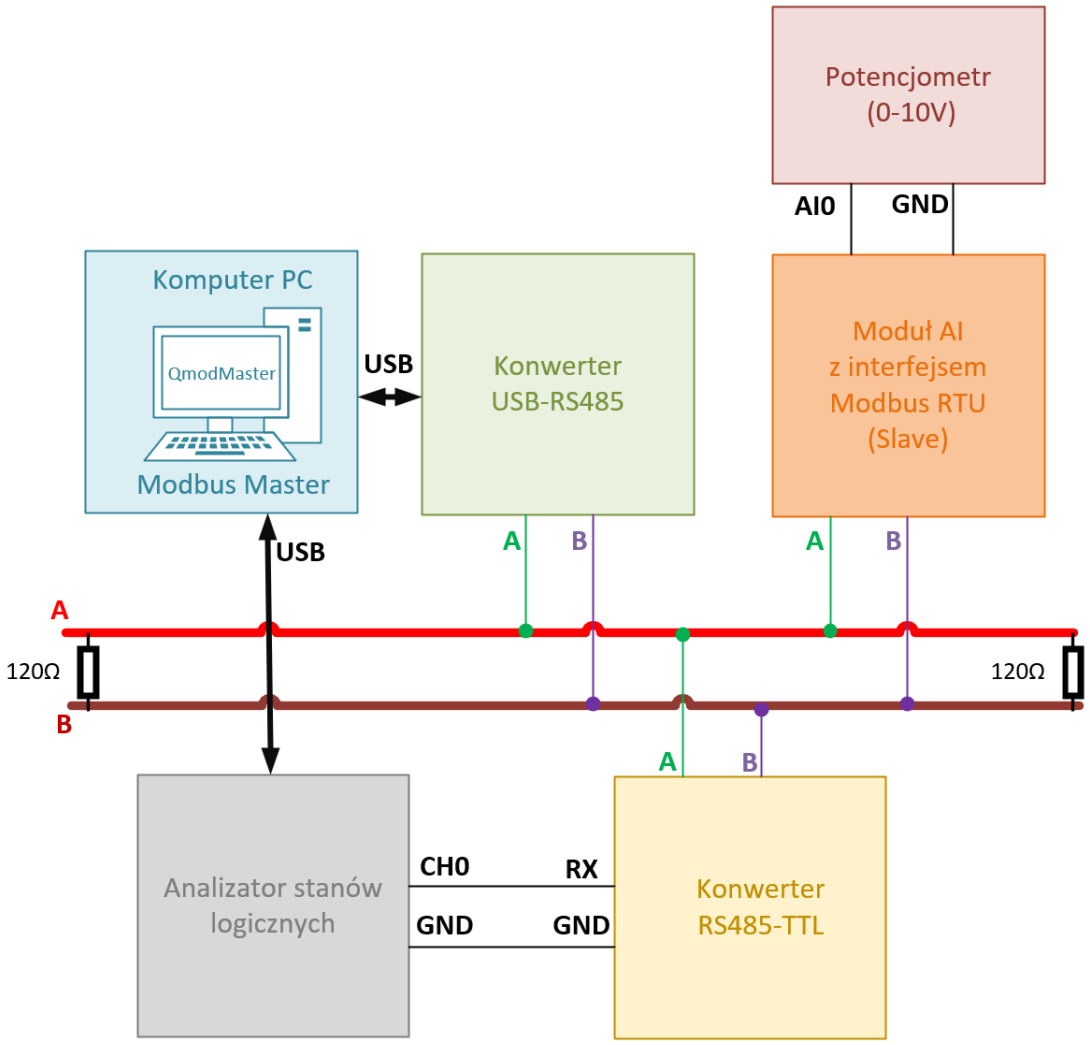
Zakres i cel ćwiczenia

Poznanie praktycznego działania protokołu Modbus RTU poprzez odczyt wartości z modułu wejść analogowych (przez RS485), konfigurację klienta w programie QModMaster oraz analizę ramek komunikacyjnych za pomocą analizatora Saleae Logic.

Stanowisko laboratoryjne

- Komputer PC z oprogramowaniem QModMaster.
- Moduł wejść analogowych (slave) z interfejsem RS-485.
- Konwerter USB–RS485.
- Analizator stanów logicznych (Saleae Logic).
- Przewody połączeniowe (z terminacją linii 120 Ω).

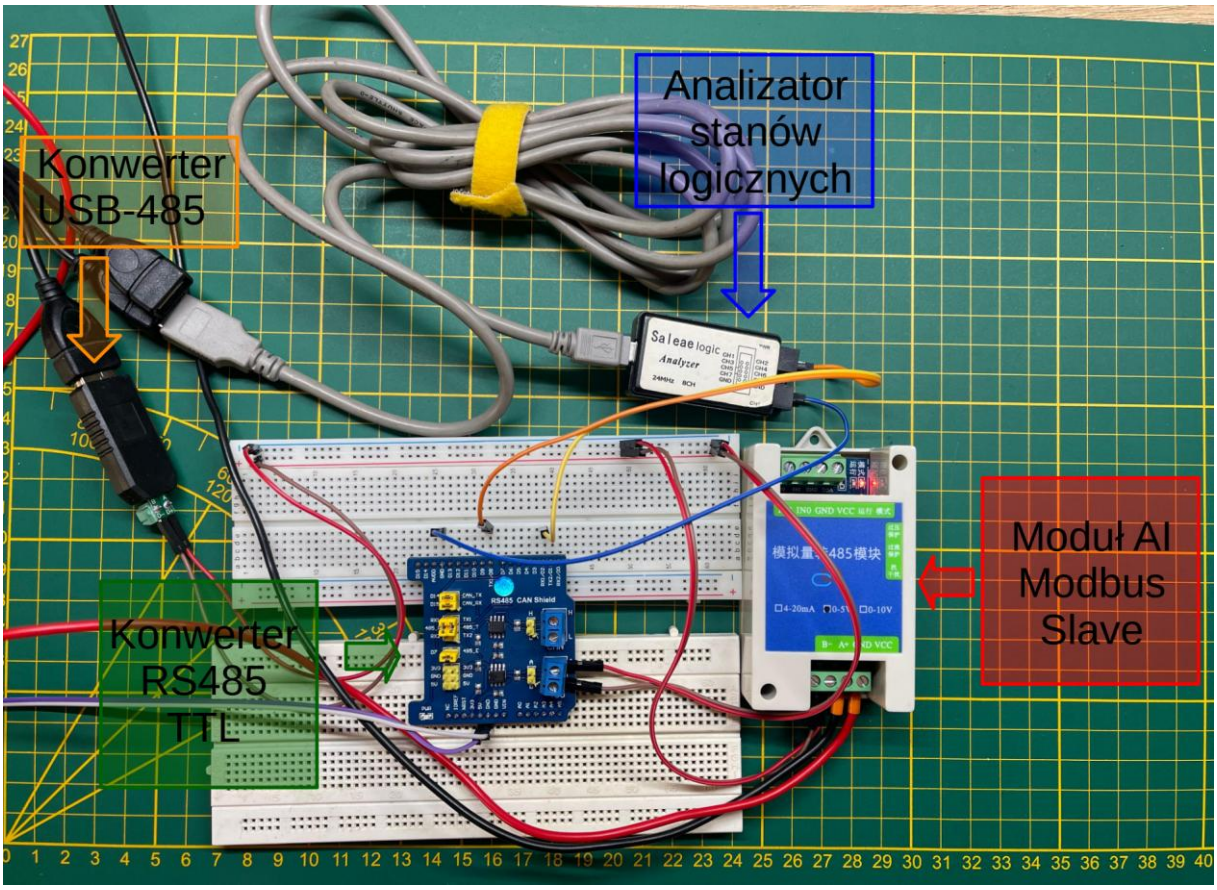
[Tekst alternatywny. Schemat przedstawia układ komunikacji Modbus RTU w oparciu o interfejs RS485. Po lewej stronie znajduje się komputer PC z oprogramowaniem QmodMaster, pełniący rolę Modbus Master. Jest on połączony kablem USB z konwerterem USB-RS485. Magistrala RS485 składa się z dwóch przewodów (oznaczonych A – czerwony i B – brązowy), zakończonych na obu końcach rezystorami terminującymi 120 Ω. Do magistrali dołączony jest moduł AI z interfejsem Modbus RTU (Slave), współpracujący z potencjometrem (0–10V), podłączonym do wejść AI0 i GND. Dodatkowo, do linii A i B dołączony jest konwerter RS485-TTL, którego sygnały RX oraz masa (GND) są podłączone do analizatora stanów logicznych (CH0 i GND).]



Rys. 7. Schemat stanowiska laboratoryjnego do komunikacji Modbus RTU z wykorzystaniem interfejsu RS485

[Tekst alternatywny Fotografia przedstawia stanowisko laboratoryjne do komunikacji Modbus RTU. Po lewej stronie widoczny jest konwerter USB-RS485 (czarny moduł z podłączonym przewodem USB). Niżej znajduje się konwerter RS485-TTL umieszczony na płytce stykowej - dodatkowy moduł elektroniczny (RS485 CAN Shield) z podłączonymi przewodami. Po prawej stronie umieszczono moduł wejść analogowych Modbus Slave, z podłączonymi przewodami sygnałowymi i zasilającymi. U góry znajduje się analizator stanów logicznych z podłączonym kablem USB. Całość ustawiona jest na zielonej macie antystatycznej z nadrukowaną siatką pomiarową.]

*Projekt „Dostosowanie kształcenia w Politechnice Świętokrzyskiej do potrzeb współczesnej gospodarki”
nr FERS.01.05-IP.08-0234/23*



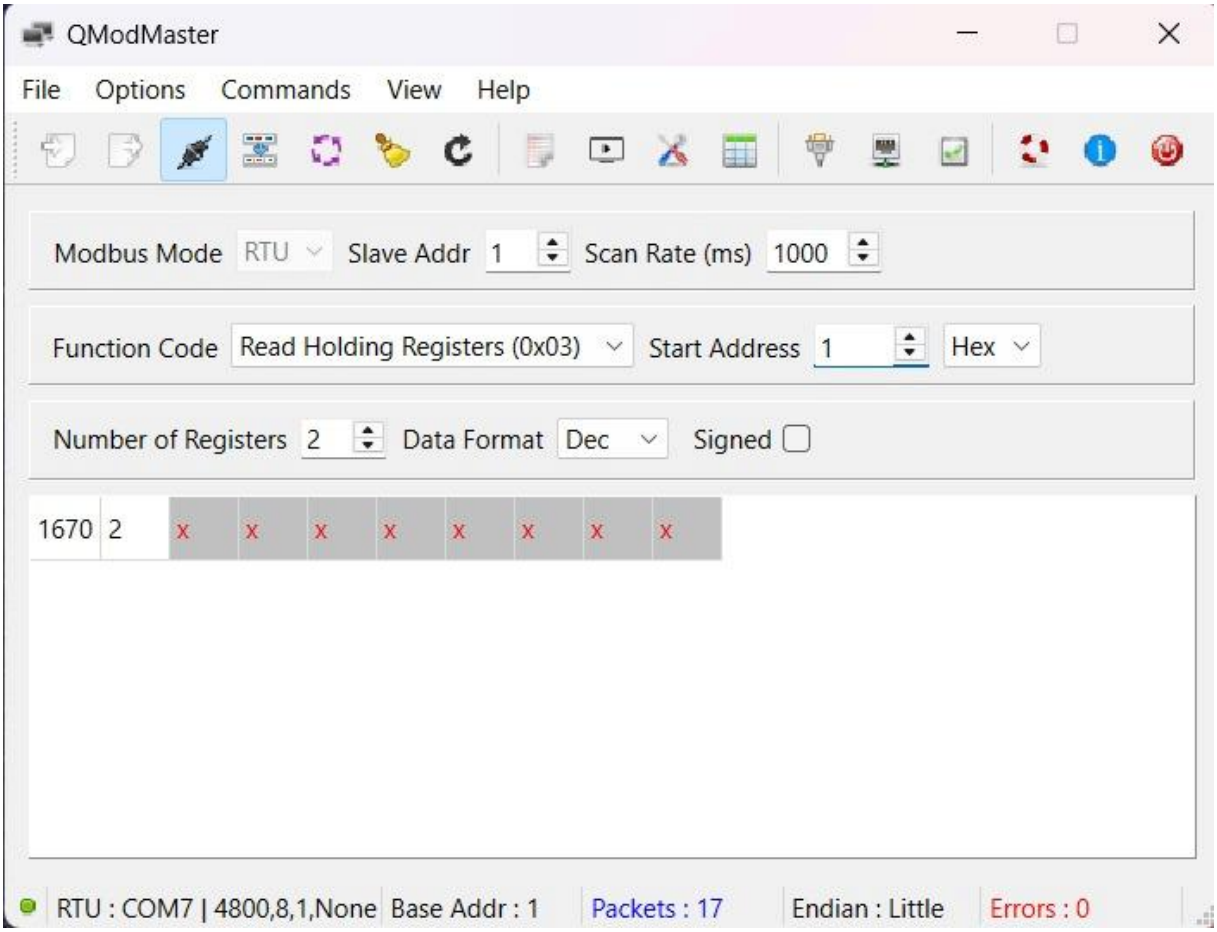
Rys. 8. Stanowisko laboratoryjne z konwerterami RS485, analizatorem stanów logicznych i modułem AI Modbus Slave

Konfiguracja w QmodMaster

Aby odczytać wartości z wejść analogowych modułu w programie QModMaster należy:

- Uruchomić QModMaster i wybrać odpowiedni port COM przypisany do konwertera USB–RS485.
- Ustawić parametry transmisji zgodnie z dokumentacją modułu: Baudrate: 4800 bps (domyślnie), Data bits: 8, Parity: None, Stop bits: 1
- W polu Mode wybrać tryb pracy RTU.
- W polu Slave ID wpisać adres urządzenia (domyślnie 1).
- W polu Function Code wybrać 03 – Read Holding Registers.
- W polu Address wpisać adres początkowego rejestru: 0 (czyli 0000H) dla odczytu kanału 1.
- W polu Quantity wpisać liczbę rejestrów do odczytu – 2 aby odczytać oba rejestry.
- Kliknąć Send – w oknie wyników pojawią się wartości rejestrów w formacie dziesiętnym i heksadecymalnym.

[Tekst alternatywny. Zrzut ekranu z programu QModMaster służącego do testowania komunikacji Modbus RTU. W górnej części okna ustawiony jest tryb pracy Modbus RTU, adres slave 1 oraz czas odświeżania 1000 ms. Wybrana funkcja to Read Holding Registers (0x03) z adresem początkowym 1 i formatem adresowania Hex. Liczba rejestrów do odczytu: 2, format danych: Dec. W tabeli wyników widoczny jest odczyt wartości 1670 oraz 2, a kolejne pola oznaczone są czerwonymi znakami „X” (brak danych). Na pasku stanu na dole okna wyświetlono: RTU: COM7 | 4800,8,1,None | Base Addr: 1 | Packets: 17 | Endian: Little | Errors: 0.]



Rys. 9. Interfejs programu QModMaster podczas testowania komunikacji Modbus RTU

Na rysunku 9 przedstawiono odczyt wartości rejestrów dla modułu analogowego. Moduł zgodnie z dokumentacją zwraca wartość w zakresie 0–4095 (12-bit). Dla wersji 0–5 V odczytana wartość 1670 będzie odpowiadać napięci 2.04V zgodnie z zależnością przedstawioną poniżej:

$$U = \left(\frac{1670}{4096}\right) \cdot 5V = 2{,}04V \quad (1)$$

[Tekst alternatywny Zrzut ekranu z analizatora stanów logicznych przedstawiający przebiegi sygnałów transmisji w protokole Modbus RTU.Na osi czasu oznaczono fragmenty ramki z danymi w postaci bajtów szesnastkowych. W pierwszej ramce widoczne są wartości: 0x01, 0x03, 0x00, 0x00, 0x00, 0x02, 0xC4, 0x0B. W drugiej ramce widoczne są wartości: 0x01, 0x03, 0x04, 0x0B, 0x26, 0x00, 0x02, 0x98, 0x1D. Przebiegi sygnałów przedstawione są w formie impulsów binarnych odpowiadających kolejnym bitom transmisji.]

Rys. 10. Ramki komunikacyjne Modbus RTU zarejestrowane analizatorem stanów logicznych

Na rysunku 10 przedstawiono „podsłuchane” ramki komunikacji pomiędzy urządzeniem Master (aplikacja QmodMaster) i Slave (moduł wejść analogowych). Urządzenie master przesyła ramkę:

```
01 03 00 00 00 02 C4 0B
```

W której znaczenie kolejnych bajtów to:

- 01 – adres slave (ID=1)
- 03 – funkcja: Read Holding Registers
- 00 00 – adres startowy rejestru: 0000h (kanał 1)
- 00 02 – liczba rejestrów: 2 (kanał 1 i kanał 2)
- C4 0B – CRC16 (LE: low= C4, high= 0B) – poprawne dla bajtów 01 03 00 00 00 02.

Urządzenie slave odpowiada ramką:

```
01 03 04 0B 26 00 02 98 1D
```

Znaczenie poszczególnych danych w ramce odpowiedzi jest następujące:

- 01 – adres slave
- 03 – echo funkcji
- 04 – liczba bajtów danych (2 rejestry × 2 bajty = 4)
- 0B 26 – rejestr 0000h (kanał 1) = 0x0B26 = 2854
- 00 02 – rejestr 0001h (kanał 2) = 0x0002 = 2

- 98 1D – CRC16 (LE: low= 98, high= 1D) – poprawne dla bajtów 01 03 04 0B 26 00 02.

5.4.2. Modbus TCP - odczyt danych z serwera Modbus TCP na Siemens S7-1200 w Pythonie

Zakres i cel ćwiczenia

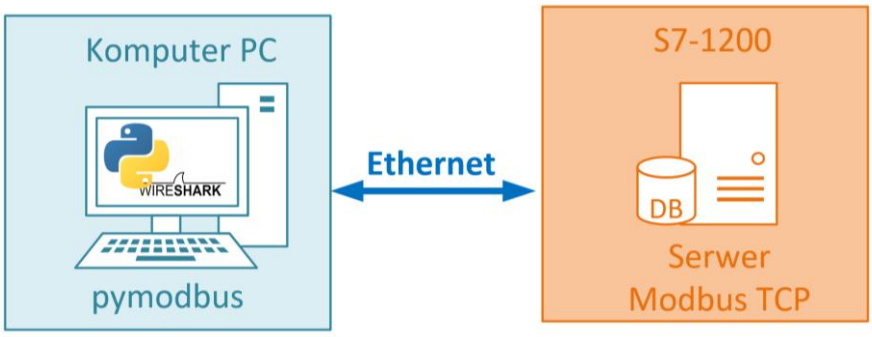
Poznanie praktycznego działania protokołu Modbus TCP poprzez:

- konfigurację sterownika Siemens S7-1200 jako serwera Modbus TCP,
- mapowanie obszarów pamięci PLC na rejestry Modbus,
- odczyt wartości rejestrów w języku Python z wykorzystaniem biblioteki pymodbus,
- dekodowanie danych 16-bitowych, 32-bitowych oraz wartości zmiennoprzecinkowych,
- podstawową analizę ramek Modbus TCP w programie Wireshark.

Sprzęt i oprogramowanie

- CPU S7-1200 (np. 1212C/1214C) + TIA Portal (dowolna współczesna wersja).
- PC w tej samej sieci (Ethernet).
- Python 3.9+ z pakietem pymodbus (np. pip install pymodbus).

[Tekst alternatywny Schemat przedstawia komunikację w protokole Modbus TCP. Po lewej stronie znajduje się komputer PC, na którym działa biblioteka pymodbus oraz program do analizy pakietów Wireshark. Po prawej stronie umieszczony jest sterownik S7-1200, pełniący rolę serwera Modbus TCP, z zaznaczoną bazą danych (DB). Komputer PC i sterownik S7-1200 są połączone siecią Ethernet (oznaczoną strzałką dwukierunkową).]



Rys. 8. Schemat komunikacji Modbus TCP pomiędzy komputerem PC a sterownikiem S7-1200

Na rysunku 9 przedstawiono konfigurację serwera Modbus TCP oraz organizację tablicy rejestrów dla sterownika S7-1200 w oprogramowaniu TIA Portal. Aby poprawnie skonfigurować PLC należy wykonać następujące kroki:

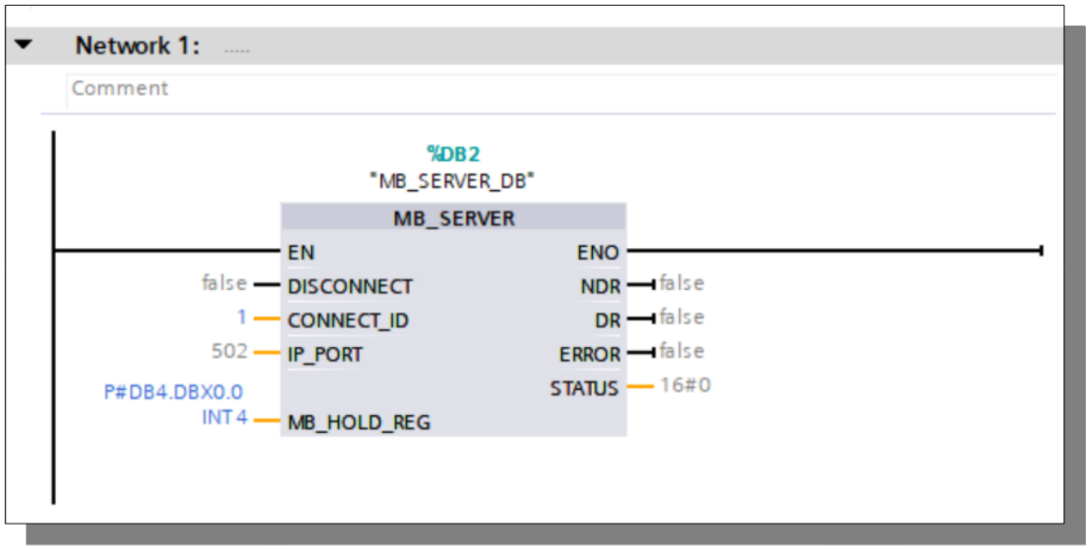
1. Konfiguracja sprzętowa i sieciowa
   - W TIA Portal utworzyć nowy projekt i dodać sterownik S7-1200 (w przedstawionym przykładzie użyto CPU1212C AC/DC/RLY).
   - W zakładce Properties → Ethernet addresses nadać CPU adres IP (np. 192.168.0.10) zgodny z podsiecią komputera.
   - Upewnić się, że komputer i sterownik są w tej samej sieci (test: ping).
2. Dodanie bloku MB_SERVER
   - Otworzyć OB1 i z biblioteki instrukcji komunikacyjnych przeciągnąć blok MB_SERVER.
   - Utworzyć dla niego DB instancyjny który będzie przechowywał parametry i status komunikacji.
   - W parametrach bloku ustawić:
     - EN = TRUE – włączenie serwera,
     - CONNECT_ID = 1 – identyfikator połączenia,
     - IP_PORT = 502 – standardowy port Modbus TCP,
     - MB_HOLD_REG = wskaźnik do tablicy rejestrów Holding (dla przedstawionego przykładu jest to P#DB4.DBX0.0),
     - INT = liczba rejestrów Holding (w przykładzie 4 ponieważ udostępniane są 4 rejestry).
3. Utworzenie bloku danych dla rejestrów
   - W projekcie utworzyć nowy Data Block (w przykładzie Data_block_1).
   - W nim zdefiniować zmienne typu Int, które będą reprezentowały kolejne rejestry Modbus Holding.
   - W przykładzie zdefiniowano cztery rejestry:
     - MB_HOLDING1 (offset 0.0, wartość początkowa 0),
     - MB_HOLDING2 (offset 2.0, wartość początkowa 0),
     - MB_HOLDING3 (offset 4.0, wartość początkowa 0),
     - MB_HOLDING4 (offset 6.0, wartość początkowa 0).
   - Każda zmienna zajmuje 2 bajty, co odpowiada jednemu rejestrowi Modbus (16 bitów).
4. Powiązanie bloku danych z MB_SERVER

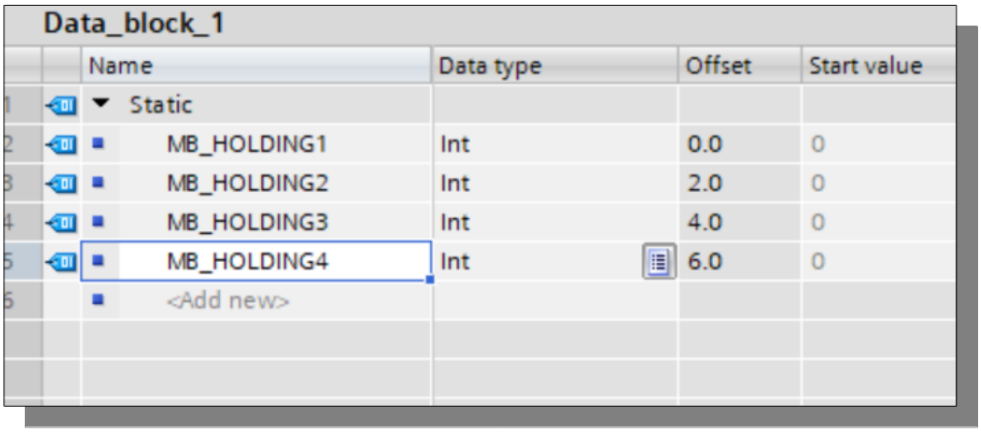
- W parametrze MB_HOLD_REG bloku MB_SERVER wskazać początek utworzonego Data Blocka (P#DB4.DBX0.0).
- W parametrze INT podać liczbę zdefiniowanych rejestrów (w przykładzie 4).
- Dzięki temu klient Modbus TCP, łącząc się ze sterownikiem, będzie mógł odczytywać i zapisywać wartości pod adresami 40001–40004.

5. Wgrywanie i test

6. Wgrać projekt do sterownika

[Tekst alternatywny] Zrzut ekranu z oprogramowania do programowania sterowników Siemens S7, przedstawiający konfigurację serwera Modbus TCP. Wyżej znajduje się fragment programu w języku blokowym z użyciem bloku MB_SERVER, którego parametry obejmują: CONNECT_ID = 1, IP_PORT = 502, MB_HOLD_REG = P#DB4.DBX0.0, INT = 4. Poniżej widoczna jest zawartość bloku danych Data_block_1, w którym zdefiniowane są cztery rejestry typu Int: MB_HOLDING1 (offset 0.0, wartość początkowa 0), MB_HOLDING2 (offset 2.0, wartość początkowa 0), MB_HOLDING3 (offset 4.0, wartość początkowa 0), MB_HOLDING4 (offset 6.0, wartość początkowa 0).]





Rys. 9. Konfiguracja bloku MB_SERVER oraz tablicy rejestrów w sterowniku Siemens S7

Tabela 6. Tabela rejestrów Modbus TCP – S7-1200

Adres Modbus (HR)	Adres startowy (offset w DB)	Nazwa zmiennej	Typ danych
40001	DB4.DBW0 (offset 0.0)	MB_HOLDING1	INT (16 bit)
40002	DB4.DBW2 (offset 2.0)	MB_HOLDING2	INT (16 bit)
40003	DB4.DBW4 (offset 4.0)	MB_HOLDING3	INT (16 bit)
40004	DB4.DBW6 (offset 6.0)	MB_HOLDING4	INT (16 bit)

Odczyt w Pythonie (pymodbus)

Na rysunku 10 przedstawiono przykładowy kod w Pythonie do odczytu danych z serwera Modbus TCP.

[Tekst alternatywny. Zrzut ekranu przedstawiający kod w języku Python z wykorzystaniem biblioteki snap7 do komunikacji ze sterownikiem Siemens S7-1200. Importowane są moduły: snap7, get_int z snap7.util oraz struct. Utworzony jest klient S7 (client = snap7.client.Client()). Nawiązywane jest połączenie z PLC o adresie IP 192.168.0.10, z parametrami rack=0 i slot=1. Z bazy danych DB4 odczytywanych jest 8 bajtów od offsetu 0 (client.db_read(4, 0, 8)). Dane są dekodowane jako 4 liczby typu UINT16 w porządku big-endian (struct.unpack('>4H', data)). Wyniki są wypisywane w konsoli (print(vals)). Na końcu wykonywane jest rozłączenie klienta (client.disconnect()).]

```python
import snap7
from snap7.util import get_int
import struct

client = snap7.client.Client()

client.connect('192.168.0.10', 0, 1)

data = client.db_read(4, 0, 8)

# 4 x UINT16 (big-endian)
vals = list(struct.unpack('>4H', data))
print(vals)

client.disconnect()
```

Rys. 10. Przykładowy kod w Pythonie do odczytu rejestrów ze sterownika PLC przy użyciu protokołu Modbus TCP

6. Sieć EtherCAT

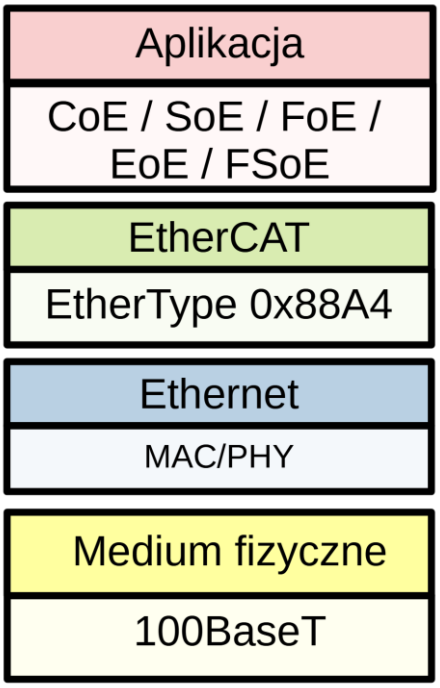
6.1. Wprowadzenie

EtherCAT (Ethernet for Control Automation Technology) jest deterministyczną siecią czasu rzeczywistego. Ramki Ethernet są przetwarzane on-the-fly przez kolejne węzły (slave), co minimalizuje opóźnienia. EtherCAT przewiduje kilka „protokołów usługowych", które działają na warstwie aplikacji i pozwalają obsługiwać różne typy urządzeń oraz zadań konfiguracyjnych:

- CoE (CANopen over EtherCAT): Najpopularniejszy mechanizm aplikacyjny. Pozwala przenieść strukturę słownika obiektów CANopen do EtherCAT-u.
  - PDO (Process Data Objects) → szybka wymiana danych procesowych w czasie rzeczywistym.
  - SDO (Service Data Objects) → wolniejsza, acykliczna komunikacja do konfiguracji parametrów.

  Dzięki CoE możliwe jest używanie istniejących profili urządzeń CANopen (np. dla napędów, czujników) bez ich modyfikacji.
- SoE (Sercos over EtherCAT): Przeznaczone głównie dla napędów. Implementuje strukturę i profile znane z sieci SERCOS. Pozwala na parametryzację serwonapędów i wymianę danych zgodnie z wymaganiami aplikacji ruchowych. Typowe w branży motion control.
- FoE (File access over EtherCAT): Protokół do przesyłania plików binarnych w trybie punkt–punkt. Używany głównie do aktualizacji firmware urządzeń EtherCAT (slave). Działa podobnie do TFTP, ale osadzony w ramach EtherCAT.
- EoE (Ethernet over EtherCAT): Mechanizm tunelowania ramek Ethernetu „przez" magistralę EtherCAT. Dzięki temu każde urządzenie slave może być osiągalne jako host w sieci TCP/IP (np. w celu diagnostyki przez HTTP, SNMP, telnet, Modbus/TCP). Nie wpływa na deterministyczny ruch czasu rzeczywistego.

[Tekst alternatywny. Schemat warstw protokołu EtherCAT w modelu komunikacyjnym. Najwyższa warstwa to Aplikacja. Poniżej znajdują się protokoły aplikacyjne: CoE (CANopen over EtherCAT), SoE (Servo over EtherCAT), FoE (File over EtherCAT), EoE (Ethernet over EtherCAT), FSoE (Safety over EtherCAT). Kolejna warstwa to EtherCAT, oznaczona identyfikatorem EtherType 0x88A4. Niżej znajduje się warstwa Ethernet (MAC/PHY). Na samym dole widoczna jest warstwa Medium fizyczne – 100BaseT.]



Rys. 11. Model warstwowy protokołu EtherCAT w architekturze sieci przemysłowej

6.2. Ramka EtherCAT i proces wymiany danych (PDO/SDO)

6.2.1. Struktura ramki EtherCAT

Ramka EtherCAT jest zwykłą ramką Ethernet, której EtherType = 0x88A4. W polu danych Ethernetu mogą znajdować się jeden lub więcej „telegramów EtherCAT". Każdy telegram zawiera nagłówek z adresem logicznym/offsetem i fragment danych przypisany do konkretnego slave'a.

[Tekst alternatywny. Schemat przedstawia strukturę ramki EtherCAT. Ramka zawiera następujące pola: Dest (różowy) – adres MAC odbiorcy, Src (zielony) – adres MAC nadawcy, Ethertype (niebieski) – wartość 0x88A4 identyfikująca EtherCAT, Telegram 1 (brązowy) – nagłówek i dane PDO, Telegram 2 (żółty) – nagłówek i dane PDO, FCS (fioletowy) – suma kontrolna CRC.]

Dest	Src	Ethertype	Telegram 1	Telegram 2	FCS
MAC	MAC	0x88A4	Nagłówek + PDO	Nagłówek + PDO	CRC

Rys 12. Struktura ramki EtherCAT z polami adresów, typem EtherType, telegramami i sumą kontrolną FCS

6.2.2. Mechanizm przekazywania ramki on-the-fly

Kluczową cechą protokołu EtherCAT jest sposób, w jaki ramka przemieszcza się przez sieć. W przeciwieństwie do klasycznego Ethernetu, gdzie ramka kierowana jest do konkretnego adresata, w EtherCAT ramka jest przesyłana do wszystkich urządzeń z użyciem adresu docelowego typu broadcast (FF:FF:FF:FF:FF:FF).

Dzięki temu każde urządzenie (slave):

- odczytuje ze strumienia swoje dane wejściowe (wejścia procesowe),
- wpisuje do ramki swoje dane wyjściowe,
- przekazuje tę samą ramkę dalej, do kolejnego urządzenia w linii.

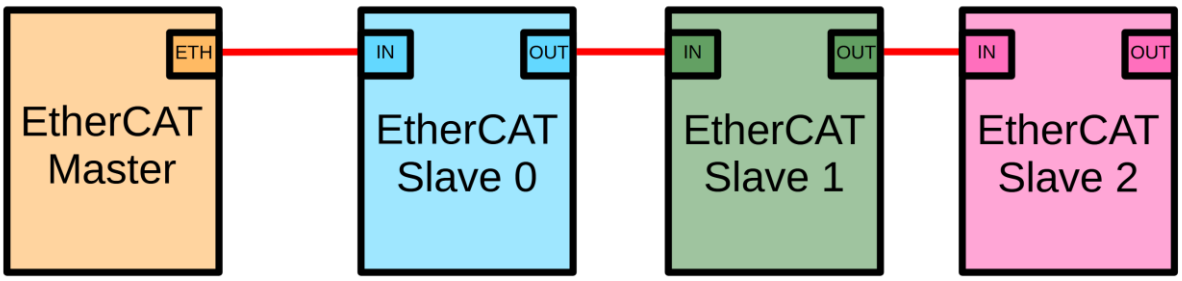
Cały ten proces odbywa się on-the-fly, czyli w locie, podczas przechodzenia ramki przez urządzenie, bez konieczności jej buforowania czy zatrzymywania.

W efekcie:

- jedna ramka może obsłużyć nawet kilkadziesiąt węzłów w czasie zaledwie kilkudziesięciu mikrosekund,
- komunikacja jest niezwykle deterministyczna i bardzo szybka,
- topologia oparta na tej zasadzie pozwala na synchronizację wielu urządzeń w czasie rzeczywistym.

[Tekst alternatywny. Schemat przedstawia przykładową topologię sieci EtherCAT.

Po lewej stronie znajduje się blok oznaczony jako EtherCAT Master, wyposażony w port Ethernet (ETH). Do niego kaskadowo podłączone są trzy urządzenia: EtherCAT Slave 0, EtherCAT Slave 1 i EtherCAT Slave 2. Każde urządzenie posiada dwa porty: IN (wejściowy) i OUT (wyjściowy), przez które przesyłana jest ramka. Czerwone linie symbolizują połączenia sieciowe Ethernet.]



Rys. 13. Topologia sieci EtherCAT z masterem i trzema urządzeniami slave połączonymi szeregowo

6.2.3. Proces cykliczny i acykliczny

EtherCAT rozróżnia dwa typy wymiany danych:

- Dane cykliczne → realizowane poprzez PDO (Process Data Objects): zawierają bieżące wartości I/O (np. stan wejścia, wartość czujnika, sygnał sterujący wyjścia), przesyłane są co cykl komunikacyjny (typowo 250 µs – 1 ms), mają gwarantowaną deterministkę i małe jittery.
- Dane acykliczne → realizowane poprzez SDO (Service Data Objects): służą do konfiguracji i parametryzacji urządzeń, wykorzystują słownik obiektów CANopen (indeks/subindeks), nie są krytyczne czasowo (np. ustawienie zakresu pomiarowego, prędkości komunikacji, offsetów).

PDO (Process Data Objects) niosą dane krytyczne czasowo i są transmitowane co cykl (typowo 0,25–1 ms):

- TxPDO – „to master”: wartości pomiarowe, stany DI.
- RxPDO – „from master”: wartości zadane, stany DO.

Minimalizujemy narzut: pola to surowe bity/bajty bez nagłówków aplikacyjnych.

Mapowanie PDO (CoE)

W CoE (CANopen over EtherCAT) zawartość PDO definiuje się przez obiekty słownika (OD):

- 0x1600–0x17FF – mapowania RxPDO (do slave’a).
- 0x1A00–0x1BFF – mapowania TxPDO (ze slave’a).
- 0x1C12 – przypisanie listy RxPDO do urządzenia.
- 0x1C13 – przypisanie listy TxPDO do urządzenia.

Każda pozycja mapowania to 32-bitowa wartość:

```
index(16 b) | subindex(8 b) | długość(8 b w bitach)
```

Przykład (TxPDO czujnika):

- 0x1A00:
  - Sub 01 = 0x6064:00:32 (pozycja DINT 32 b)
  - Sub 02 = 0x6041:00:16 (status WORD 16 b)

6.2.4. Maszyna stanów EtherCAT

Każde urządzenie EtherCAT (slave) pracuje zgodnie z określoną maszyną stanów, która definiuje poziom gotowości urządzenia do komunikacji i wymiany danych procesowych.

Podstawowe stany pracy:

- INIT – stan początkowy po włączeniu zasilania. Urządzenie jest zainicjalizowane, ale nie komunikuje się jeszcze z masterem.
- PREOP (Pre-Operational) – urządzenie jest rozpoznane przez mastera, możliwy jest odczyt i zapis parametrów konfiguracyjnych (SDO), ale nie odbywa się jeszcze wymiana danych procesowych PDO.
- SAFEOP (Safe-Operational) – urządzenie rozpoczęło konfigurację PDO i może wysyłać dane wejściowe (Input), ale wyjścia (Output) są jeszcze nieaktywne. Ten stan pozwala na sprawdzenie poprawności konfiguracji bez ryzyka sterowania wyjściami.
- OP (Operational) – pełny stan operacyjny, w którym wymiana PDO jest aktywna w obu kierunkach (Input/Output). Slave przekazuje i odbiera dane cyklicznie zgodnie z konfiguracją mastera.

Przejścia między stanami kontrolowane są przez mastera za pomocą specjalnych ramek EtherCAT. Zwykle po uruchomieniu systemu sekwencja wygląda następująco:

```
INIT → PREOP → SAFEOP → OP
```

Maszyna stanów EtherCAT pełni kluczową rolę w zapewnieniu bezpiecznej i deterministycznej pracy całej sieci. Dzięki kolejnym poziomom (INIT, PREOP, SAFEOP, OP) możliwe jest stopniowe uruchamianie urządzeń – od samej inicjalizacji sprzętu, poprzez konfigurację i test wymiany danych, aż do pełnego działania w trybie operacyjnym.

Takie podejście pozwala:

- uniknąć niekontrolowanego załączenia wyjść natychmiast po starcie systemu,
- weryfikować poprawność konfiguracji urządzeń zanim rozpoczną pracę w cyklu,
- w sposób deterministyczny i przewidywalny synchronizować dużą liczbę węzłów,
- zapewnić możliwość diagnostyki oraz bezpiecznego wyłączania poszczególnych części systemu.

6.3. Komunikacja Ethercat komputera PC (Linux) z modułem IO Beckhoff EK1818 z użyciem biblioteki IgH Mater (Etherlab)

Zakres i cel ćwiczenia

Poznanie możliwość uruchomienia komunikacji przez sieć EtherCAT na komputerze PC z wykorzystaniem klasycznego sprzętu sieciowego (karta sieciowa) poprzez:

- Uruchomienie mastera EtherCAT (IgH/EtherLab) na Linuksie i wykrycie węzłów.
- Konfigurację interfejsu sieciowego do pracy w trybie „raw Ethernet”.
- Identyfikacja i diagnostyka slave’a Beckhoff EK1818 (wejścia/wyjścia binarne).
- Mapowanie danych procesowych PDO oraz cykliczny odczyt/zapis w programie użytkownika.
- Test funkcjonalny: odbicie DI→DO i rejestracja stanów.
- Podstawy diagnostyki: ethercat slaves/pdos, statystyki mastera

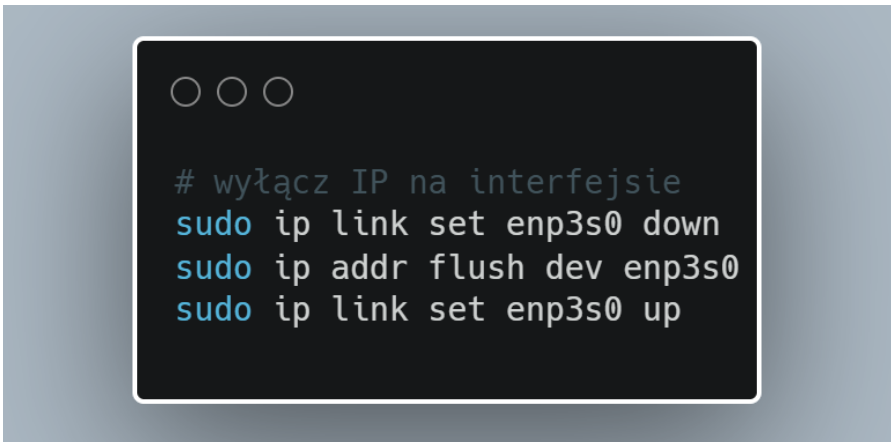
Sprzęt i oprogramowanie

- PC z kartą Ethernet (zalecane uruchomienie bez wirtualizacji; VM typu VirtualBox filtruje EtherCAT i nie działa bez PCI passthrough).
- Beckhoff EK1818 (wejścia binarne 8×DI, wyjścia 4×DO) + zasilacz 24 V DC
- Okablowanie: skrętka RJ45; połączenie PC → EtherCAT IN (EK1818).
- Oprogramowanie (Linux): Dystrybucja: Ubuntu/Debian/openSUSE (kernel zgodny z modułami IgH). Oraz zainstalowany IgH EtherCAT Master.

Konfiguracja interfejsu

EtherCAT działa na surowym Ethernecie, więc karta nie może mieć IP. W związku z tym konieczne jest odpowiednie skonfigurowanie interfejsu sieciowego. Na rysunku poniżej przedstawiono konfigurację dla interfejsu enp3s0.

[Tekst alternatywny. Zrzut ekranu przedstawia okno terminala systemu Linux z przykładowymi poleceniami do wyłączenia i ponownego uruchomienia interfejsu sieciowego. Komentarz: „# wyłącz IP na interfejsie”. Polecenia: sudo ip link set enp3s0 down – wyłączenie interfejsu sieciowego enp3s0, sudo ip addr flush dev enp3s0 – usunięcie przypisanego adresu IP, sudo ip link set enp3s0 up – ponowne włączenie interfejsu sieciowego.]



Rys. 13. Polecenia systemu Linux do usunięcia adresu IP z interfejsu sieciowego enp3s0

Następne konieczne jest wpisanie danych interfejsu sieciowego (który będzie używany dla sieci EtherCAT) do pliku konfiguracyjnego mastera EtherCAT.

[Tekst alternatywny. Zrzut ekranu przedstawia fragment pliku konfiguracyjnego systemu Linux dla sterownika EtherCAT Master. Parametr MASTER0_DEVICE="enp3s0" określa kartę sieciową, która będzie używana jako interfejs główny EtherCAT Master. Parametr DEVICE_MODULES="generic" definiuje użycie ogólnego sterownika (modułu) dla obsługi urządzeń EtherCAT.]

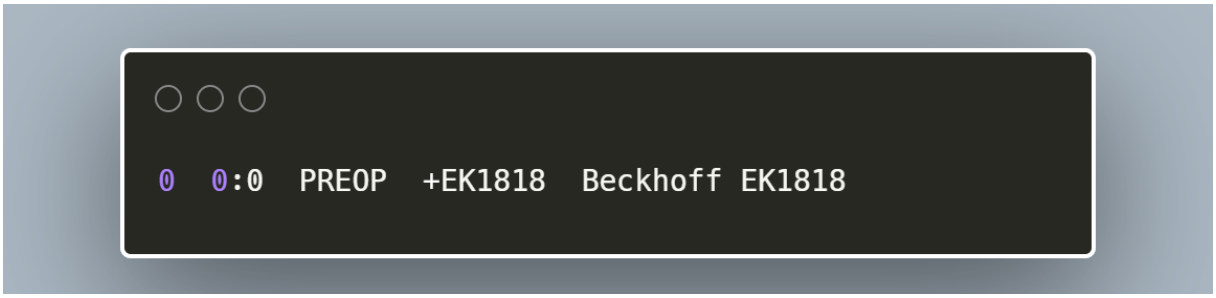
Rys. 14. Fragment pliku konfiguracyjnego definiującego urządzenie sieciowe i moduł sterownika dla EtherCAT Master

Po dokonaniu odpowiedniej konfiguracji można zrestartować usługę mastera sieci EtherCAT i wywołać polecenie skanujące magistralę w poszukiwaniu podłączonych do komputera slave'ów.

[Tekst alternatywny. Zrzut ekranu przedstawia okno terminala systemu Linux z poleceniami obsługi EtherCAT Master. Polecenia to: sudo /etc/init.d/ethercat restart – ponowne uruchomienie usługi EtherCAT Master. oraz sudo ethercat slaves – polecenie wyświetlające listę podłączonych urządzeń slave w sieci EtherCAT.]

Rys. 15. Polecenia systemu Linux do restartu usługi EtherCAT Master i wyświetlenia listy urządzeń slave

[Tekst alternatywny] Zrzut ekranu przedstawia wynik działania polecenia ethercat slaves w terminalu systemu Linux. Wyświetlone są informacje o jednym podłączonym urządzeniu EtherCAT: identyfikator: 0 0:0, stan: PREOP (tryb pre-operational), typ modułu: EK1818, producent: Beckhoff.]



Rys. 16. Wynik działania polecenia ethercat slaves pokazujący wykryty moduł Beckhoff EK1818 w stanie PREOP

[Tekst alternatywny. Zrzut ekranu przedstawia zestaw podstawowych poleceń konsolowych służących do obsługi EtherLab (IgH EtherCAT Master) w systemie Linux. Pokazano polecenia umożliwiające uruchamianie, zatrzymywanie i

restartowanie usługi EtherCAT, a także sprawdzanie jej statusu. W dalszej części znajdują się komendy pozwalające uzyskać informacje o masterze, ponownie skanować magistralę EtherCAT i wyświetlać listę podłączonych urządzeń slave wraz ze szczegółowymi danymi. Kolejna grupa dotyczy danych procesowych, w tym map PDO i listy obiektów SDO. Zamieszczono również przykłady odczytu i zapisu obiektów SDO, polecenia do zmiany stanów urządzeń slave oraz zestaw komend diagnostycznych, obejmujących statystyki ramek, odczyt rejestrów, pobieranie plików XML (ESI) oraz generowanie struktur C dla PDO.]

```
# 🔹 Podstawowe polecenia EtherLab (IgH EtherCAT Master)

# --- Sterowanie usługą EtherCAT ---
sudo /etc/init.d/ethercat start      # uruchomienie mastera EtherCAT
sudo /etc/init.d/ethercat stop       # zatrzymanie mastera
sudo /etc/init.d/ethercat restart    # restart usługi
sudo systemctl status ethercat       # sprawdzenie statusu (systemd)

# --- Informacje o masterze ---
ethercat masters                     # lista dostępnych masterów (np. /dev/EtherCAT0)
ethercat master                      # szczegóły mastera (stan, interfejs, statystyki)

# --- Wykrywanie i zarządzanie slave'ami ---
sudo ethercat rescan                 # ponowne skanowanie magistrali
ethercat slaves                      # lista podłączonych slave'ów
ethercat slaves -v                   # szczegółowe info (Vendor ID, Product Code, Serial)

# --- Dane procesowe ---
ethercat pdos                        # mapa PDO wszystkich slave'ów
ethercat pdos -p0                    # mapa PDO konkretnego slave'a (port 0)
ethercat sdos -p0                    # lista obiektów SDO slave'a (słownik obiektów)

# Odczyt i zapis SDO:
sudo ethercat upload -p0 0x1018 1    # odczytaj obiekt SDO (np. Product Code)
sudo ethercat download -p0 0x1600 1 0x000F   # zapisz wartość do SDO

# --- Stany urządzeń ---
ethercat states                      # pokaż stany wszystkich slave'ów (INIT/PREOP/SAFEOP/OP)
sudo ethercat states -p0 OP          # ustaw slave 0 w tryb Operation (OP)

# --- Diagnostyka ---
ethercat master -p                   # statystyki ramek (Tx/Rx, błędy)
ethercat reg -p0 0x0100              # odczyt rejestru slave'a (adres, hex)
ethercat xml -p0                     # pobierz XML (ESI) z urządzenia
ethercat cstruct                     # wygeneruj C-struktury dla PDO (do użycia w
programie)
```

Rys. 17. Przykładowe polecenia EtherLab (IgH EtherCAT Master) w systemie Linux do zarządzania siecią EtherCAT, slave'ami i diagnostyki

Projekt „Dostosowanie kształcenia w Politechnice Świętokrzyskiej do potrzeb współczesnej gospodarki” nr FERS.01.05-IP.08-0234/23

Dotychczas zaprezentowane zostały podstawowe narzędzia powłoki systemowej umożliwiające konfigurację, diagnozowanie i sterowanie siecią EtherCAT. Narzędzia te są bardzo przydatne w pracy inżyniera podczas uruchamiania i testowania sieci, jednak w zastosowaniach praktycznych sterowanie urządzeniami odbywa się najczęściej z poziomu aplikacji użytkownika.

IgH EtherCAT Master udostępnia odpowiednią bibliotekę programistyczną (libethercat), która pozwala na tworzenie własnych programów w językach C oraz C++. Dzięki temu możliwa jest nie tylko konfiguracja sieci i zmiana stanów urządzeń, ale przede wszystkim cykliczna wymiana danych procesowych (PDO) w czasie rzeczywistym.

W kolejnej części zostanie przedstawiony przykład programu w języku C++, który ilustruje proces:

- otwarcia połączenia z masterem,
- utworzenia domeny wymiany danych,
- konfiguracji slave'a Beckhoff EK1818,
- oraz cyklicznego odczytu stanu wejść binarnych.

Takie podejście umożliwia integrację systemów EtherCAT z własnym oprogramowaniem sterującym i pomiarowym, a także stanowi podstawę do dalszej pracy projektowej.

[Tekst alternatywny. Zrzut ekranu przedstawia przykładowy program w języku C++ z wykorzystaniem biblioteki IgH EtherCAT Master. Kod demonstruje procedurę otwarcia mastera EtherCAT, utworzenia domeny do wymiany danych PDO, konfiguracji urządzenia slave Beckhoff EK1818 na podstawie identyfikatora producenta i produktu, a następnie mapowania PDO. Program aktywuje mastera, pobiera wskaźnik do danych domeny i wchodzi w główną pętlę, w której cyklicznie odbiera i przetwarza dane procesowe. W przykładzie wykonywany jest odczyt 16-bitowych wejść cyfrowych, których wartość wypisywana jest na standardowe wyjście w formacie szesnastkowym. Po każdym cyklu dane są umieszczane w kolejce i wysyłane do urządzeń slave, a pętla powtarzana jest co 10 ms. Na końcu program zwalnia mastera i kończy działanie.]

```cpp
#include <iostream>
#include <ecrt.h>

int main() {
    // Otwórz mastera
    ec_master_t* master = ecrt_request_master(0);
    if (!master) {
        std::cerr << "Nie można otworzyć EtherCAT mastera!\n";
        return -1;
    }

    // Utwórz domenę do wymiany PDO
    ec_domain_t* domain = ecrt_master_create_domain(master);
    if (!domain) {
        std::cerr << "Nie można stworzyć domeny!\n";
        return -1;
    }

    // Dane modułu EK1818 (VID: 0x00000002, PID: 0x07123052 – przykładowe)
    uint32_t vendor_id = 0x00000002;
    uint32_t product_id = 0x07123052;

    // Konfiguracja slave'a
    ec_slave_config_t* sc = ecrt_master_slave_config(master, 0, 0, vendor_id, product_id);
    if (!sc) {
        std::cerr << "Nie znaleziono EK1818!\n";
        return -1;
    }

    // Mapowanie PDO
    static ec_pdo_entry_reg_t domain_regs[] = {
        {0, 0, vendor_id, product_id, 0x6000, 1, NULL}, // wejścia cyfrowe
        {}
    };
    ecrt_domain_reg_pdo_entry_list(domain, domain_regs);

    // Włączenie mastera
    ecrt_master_activate(master);
    uint8_t* domain_pd = ecrt_domain_data(domain);

    // Pętla główna
    while (true) {
        ecrt_master_receive(master);
        ecrt_domain_process(domain);

        // Odczyt wejść (przykład: 16 bitów)
        uint16_t inputs = EC_READ_U16(domain_pd);
        std::cout << "Wejścia: 0x" << std::hex << inputs << std::dec << std::endl;

        ecrt_domain_queue(domain);
        ecrt_master_send(master);

        usleep(10000); // 10 ms
    }

    ecrt_release_master(master);
    return 0;
}
```

Rys. 18. Przykładowy program w języku C++ wykorzystujący IgH EtherCAT Master do konfiguracji slave'a i odczytu danych procesowych

Projekt „Dostosowanie kształcenia w Politechnice Świętokrzyskiej do potrzeb współczesnej gospodarki" nr FERS.01.05-IP.08-0234/23

Aby skompilować program należy użyć kompilatora g++ z linkowaniem odpowiednich bibliotek.

[Tekst alternatywny. Zrzut ekranu przedstawia przykład kompilacji i uruchomienia programu w języku C++ wykorzystującego bibliotekę EtherCAT. Do kompilacji użyto polecenia g++ ek1818_test.cpp -o ek1818_test -I/usr/local/include -L/usr/local/lib -lethercat, w którym wskazano plik źródłowy, nazwę pliku wynikowego oraz ścieżki do katalogów nagłówków i bibliotek, a także dołączono bibliotekę EtherCAT. Następnie program został uruchomiony z uprawnieniami administratora za pomocą polecenia sudo ./ek1818_test.]

Rys. 19. Kompilacja i uruchomienie programu w języku C++ z użyciem biblioteki EtherCAT

7. Protokół PROFINET

7.1. Wprowadzenie

PROFINET (Process Field Net) to otwarty standard komunikacji przemysłowej opracowany i rozwijany przez organizację PI – PROFIBUS & PROFINET International. Stanowi on naturalne rozwinięcie technologii PROFIBUS, wprowadzając komunikację opartą na sieciach Ethernet. Dzięki temu umożliwia integrację urządzeń przemysłowych z klasycznymi sieciami IT oraz zapewnia obsługę wymagających aplikacji czasu rzeczywistego.

Podstawowe cechy PROFINET:

- oparty na standardowym Ethernet IEEE 802.3,
- zapewnia transmisję danych cyklicznych i acyklicznych,
- wspiera komunikację deterministyczną w czasie rzeczywistym,
- umożliwia łatwą integrację urządzeń różnych producentów dzięki profilom aplikacyjnym,
- pozwala na stosowanie redundantnych topologii i mechanizmów diagnostycznych.

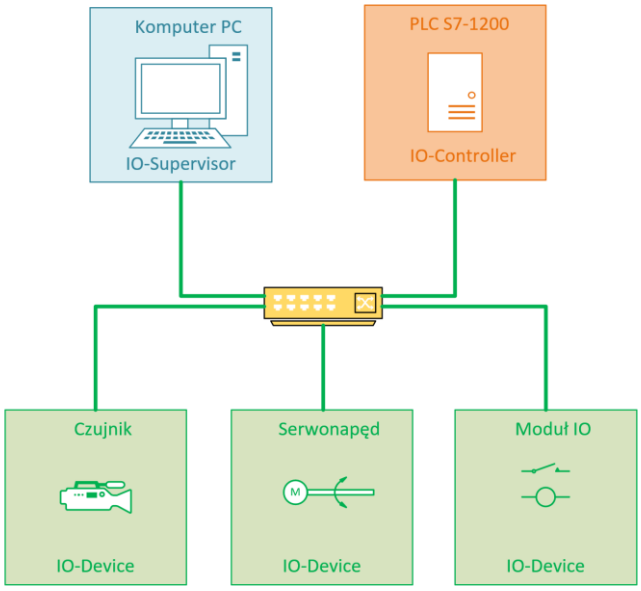
7.2. Architektura i warstwy

Architektura systemu PROFINET oparta jest na trzech typach urządzeń:

- IO-Controller – centralny kontroler systemu (najczęściej sterownik PLC), zarządza konfiguracją i cykliczną wymianą danych,
- IO-Device – urządzenia polowe, np. czujniki, moduły I/O, napędy, które wymieniają dane procesowe z kontrolerem,
- IO-Supervisor – stacja inżynierska lub narzędzie diagnostyczne, służące do uruchamiania i monitorowania systemu.

W ujęciu warstwowym PROFINET wykorzystuje standardowy Ethernet (warstwa fizyczna i MAC), a w wyższych warstwach definiuje własne rozszerzenia dla obsługi danych procesowych i diagnostycznych. Topologia sieci PROFINET jest elastyczna: może przyjmować formę gwiazdy, linii lub pierścienia. W przypadku pierścienia stosowany jest protokół MRP (Media Redundancy Protocol), zapewniający bardzo szybkie przełączenie w razie awarii jednego z połączeń.

[Tekst alternatywny. Schemat przedstawia przykładową architekturę sieci Profinet. W jej skład wchodzi komputer PC pełniący rolę IO-Supervisora oraz sterownik PLC S7-1200 pełniący rolę IO-Controllera. Oba urządzenia są połączone ze sobą i z przełącznikiem sieciowym. Do tego samego przełącznika podłączone są trzy urządzenia typu IO-Device: czujnik, serwonapęd oraz moduł wejść/wyjść. Wszystkie elementy komunikują się ze sobą za pomocą wspólnej sieci Ethernet.]



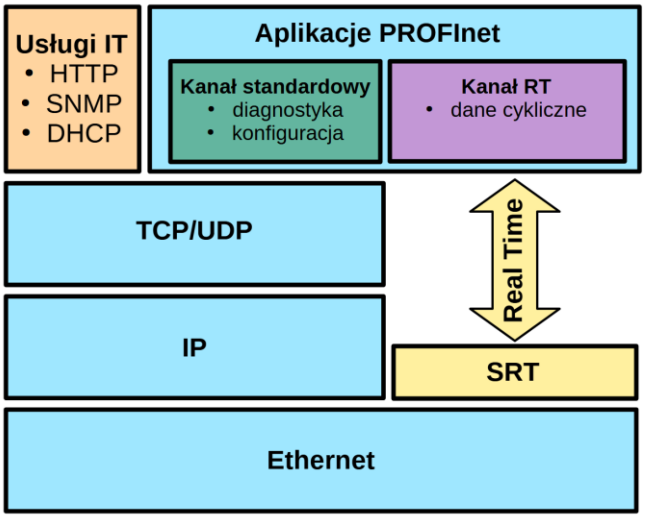
Rys. 20. Architektura sieci Profinet

7.3. Klasy komunikacji PROFINET

Standard PROFINET definiuje kilka klas komunikacji w zależności od wymagań czasowych aplikacji:

- PN-IO – podstawowa wymiana danych wejść/wyjść, służąca do komunikacji cyklicznej i acyklicznej pomiędzy IO-Controller a IO-Device.
- RT (Real-Time) – tryb czasu rzeczywistego, w którym dane procesowe przesyłane są w cyklu deterministycznym (typowo 1–10 ms). Zapewnia to płynną obsługę standardowych procesów automatyki.
- IRT (Isochronous Real-Time) – tryb izochroniczny, umożliwiający precyzyjną synchronizację urządzeń w sieci (np. w sterowaniu ruchem). Dane przesyłane są w ramach ściśle rezerwowanych przedziałów czasowych, co pozwala osiągać czasy cyklu poniżej 1 ms.

[Tekst alternatywny. Schemat przedstawia model komunikacji w sieci Profinet, pokazujący współistnienie standardowych usług IT oraz komunikacji czasu rzeczywistego. Po lewej stronie znajdują się aplikacje IT, takie jak HTTP, SNMP czy DHCP, które wykorzystują tradycyjne protokoły TCP/UDP i IP działające na warstwie Ethernet. Po prawej stronie zaznaczono aplikacje Profinet, które korzystają z dwóch kanałów komunikacyjnych: standardowego oraz czasu rzeczywistego. Kanał standardowy opiera się na klasycznym stosie TCP/IP, natomiast kanał czasu rzeczywistego (Real Time Channel) realizuje wymianę danych omijając warstwę TCP/UDP i IP, co oznaczono jako SRT (Send Real Time). Dzięki temu możliwe jest równoczesne działanie aplikacji IT i komunikacji deterministycznej w ramach jednej infrastruktury sieci Ethernet.]



Rys. 21. Model komunikacji w sieci Profinet z rozdzieleniem kanału standardowego i kanału czasu rzeczywistego – w oparciu o []

Klasy konformacji

Klasy konformacji PROFINET (Conformance Classes, CC-A…CC-D) to kategorie zgodności urządzeń i systemów z wymaganiami standardu. Określają one zakres funkcjonalności i obsługiwane mechanizmy komunikacji, jakie dane urządzenie musi spełniać, aby być uznanym za zgodne z PROFINET.

Tabela 7. Klasy konformacji (CC-A…CC-D)

Klasa	Funkcjonalność
CC-A	RT, cykliczne IO, parametry, alarmy.
CC-B	diagnostyka (SNMP), LLDP, wykrywanie topologii
CC-C	rezerwacja pasma (IRT), synchronizacja, redundancja mediów.
CC-D	TSN (Time-Sensitive Networking), pełna komunikacja na warstwie 2, RSI.

W praktyce klasy konformacji PROFINET określają, jakie funkcje komunikacyjne i diagnostyczne obsługuje dane urządzenie. Producent deklaruje klasę CC w dokumentacji, a użytkownik dobiera sprzęt tak, aby spełniał wymagania konkretnej aplikacji. Dzięki temu proste moduły IO mogą działać w CC-A, a zaawansowane systemy sterowania ruchem wymagają urządzeń zgodnych z CC-C lub wyższą.

7.4. Ramka PROFINET

Dane w PROFINET przesyłane są w standardowych ramach Ethernet, zarejestrowanych pod odpowiednimi identyfikatorami EtherType:

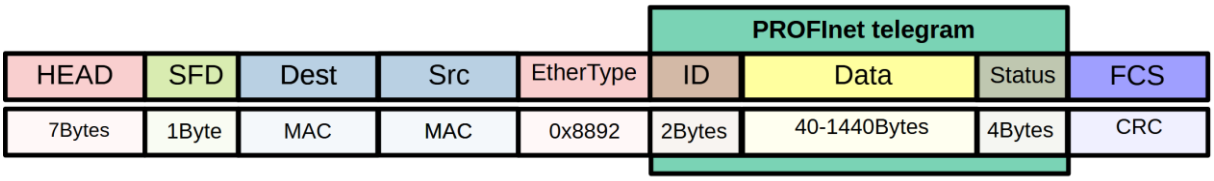
- 0x8892 – PROFINET RT (komunikacja czasu rzeczywistego),
- 0x88D8 – PROFINET IRT (komunikacja izochroniczna).

Struktura ramki PROFINET obejmuje:

- nagłówki Ethernet (adres MAC źródłowy i docelowy, EtherType),
- pole danych PROFINET z informacjami cyklicznymi lub acyklicznymi,
- sumę kontrolną FCS (Frame Check Sequence).

W przypadku komunikacji cyklicznej ramka zawiera wartości wejść/wyjść procesowych, aktualizowane w każdym cyklu wymiany. W komunikacji acyklicznej przesyłane są dane diagnostyczne, parametry konfiguracji i alarmy.

[Tekst alternatywny. Rysunek przedstawia strukturę ramki Profinet, która bazuje na standardowej ramce Ethernet. Na początku znajdują się pola HEAD o długości siedmiu bajtów oraz SFD o długości jednego bajtu. Następnie widoczne są pola adresów MAC odbiorcy i nadawcy, a po nich pole EtherType z wartością 0x8892 identyfikującą protokół Profinet. Kolejna część ramki obejmuje specyficzny telegram Profinet, który zawiera identyfikator, dane o zmiennej długości od czterdziestu do 1440 bajtów oraz pole statusu. Ramka kończy się sumą kontrolną FCS obliczaną metodą CRC.]



Rys. 22. Struktura ramki Profinet

7.5. Proces wymiany danych (cykliczny i acykliczny)

W PROFINET wyróżnia się dwa podstawowe mechanizmy wymiany danych:

- Dane cykliczne – wykorzystywane do przesyłania aktualnych wartości wejść i wyjść procesowych. Transmisja odbywa się z określonym czasem cyklu, co gwarantuje deterministyczną wymianę danych.
- Dane acykliczne – obejmują komunikację związaną z konfiguracją urządzeń, parametryzacją i diagnostyką. Przesyłane są z użyciem protokołów DCP i RPC (opartych o TCP/UDP).

Role urządzeń w procesie wymiany danych:

- IO-Controller (np. sterownik PLC) inicjuje komunikację i cyklicznie wymienia dane z urządzeniami polowymi,
- IO-Device (np. moduł wejść/wyjść, czujnik) dostarcza dane pomiarowe i przyjmuje sygnały sterujące,
- IO-Supervisor (np. komputer z oprogramowaniem inżynierskim) służy do uruchamiania, konfiguracji i diagnostyki.

7.6. Mechanizmy dodatkowe

PROFINET wprowadza szereg mechanizmów wspierających niezawodność i łatwość zarządzania siecią:

- DCP (Discovery and Configuration Protocol) – umożliwia nadawanie nazw i adresów IP urządzeniom bez konieczności stosowania dodatkowych narzędzi,
- MRP (Media Redundancy Protocol) – zapewnia odporność sieci na awarie poprzez wykorzystanie redundantnej topologii pierścieniowej,
- LLDP (Link Layer Discovery Protocol) – pozwala na automatyczne wykrywanie sąsiadujących urządzeń i topologii sieci,
- SNMP (Simple Network Management Protocol) – wspiera monitorowanie i zarządzanie siecią,

Alarmy i diagnostyka – urządzenia mogą zgłaszać zdarzenia i błędy do IO-Controllera, co pozwala na szybką reakcję w systemie.

7.7. Maszyna stanów PROFINET IO

Każde urządzenie PROFINET przed rozpoczęciem wymiany danych musi przejść określoną sekwencję stanów:

- Discovery – IO-Controller wykrywa urządzenie w sieci i przypisuje mu nazwę przy pomocy DCP,
- Configuration – przesyłana jest konfiguracja slotów i subslotów, definiująca strukturę danych IO,
- Parameterization – do urządzenia wysyłane są parametry konfiguracyjne,
- Application Ready – urządzenie rozpoczyna wymianę cyklicznych danych procesowych.

Mechanizm ten zapewnia, że każde urządzenie przed rozpoczęciem pracy zostaje poprawnie skonfigurowane i zainicjalizowane przez kontroler.

7.8. Pliki GSDML w PROFINET

Pliki GSDML (General Station Description Markup Language) pełnią w systemie PROFINET rolę opisu urządzeń. Każdy producent dostarcza dla swoich IO-Device taki plik w formacie XML, zgodny z definicją GSDML, który zawiera wszystkie niezbędne informacje o funkcjonalności urządzenia.

Podstawowe elementy pliku GSDML

- Dane identyfikacyjne urządzenia – producent, typ urządzenia, wersja, numer katalogowy,
- Struktura modułowa – dostępne sloty i subsloty, które mogą być konfigurowane,
- Opis interfejsu komunikacyjnego – obsługiwane tryby (RT, IRT), klasy konformacji,
- Dane diagnostyczne i alarmowe – jakie zdarzenia urządzenie może zgłaszać,
- Parametry konfiguracyjne – dostępne opcje ustawień (np. adresacja, czasy cykli, zakres pomiarowy).

Rola plików GSDML w praktyce

- Inżynier wgrywa plik GSDML do oprogramowania inżynierskiego (np. TIA Portal, Step7).
- Urządzenie staje się widoczne w katalogu sprzętowym jako gotowy blok, możliwy do użycia w projekcie.
- Na podstawie GSDML kontroler (IO-Controller) przesyła konfigurację slotów, parametrów i diagnostyki do IO-Device podczas inicjalizacji.

Znaczenie

Bez pliku GSDML kontroler nie wiedziałby, jakie dane procesowe i parametry obsługuje urządzenie. Plik stanowi więc swoisty „sterownik” komunikacyjny, zapewniający zgodność urządzeń różnych producentów.

[Tekst alternatywny Zrzut ekranu przedstawia fragment pliku konfiguracyjnego GSDML używanego w sieci Profinet. Plik zapisany w formacie XML zawiera informacje o urządzeniu typu IO-Device, między innymi identyfikator producenta i urządzenia, nazwę produktu, numer katalogowy, wersję sprzętu i oprogramowania. Określony jest punkt dostępu urządzenia DAP z przypisanym numerem modułu oraz przykładowy moduł wejść cyfrowych. W sekcji danych I/O opisano strukturę wymiany informacji, wskazując wejście o długości ośmiu bitów.]

```
<?xml version="1.0" encoding="UTF-8"?>
<GSDML xmlns="http://www.profibus.com/GSDML/2003/11/PROFINET">
  <ProfileHeader>
    <ProfileIdentification>PROFINET Device Profile</ProfileIdentification>
    <ProfileRevision>1.0</ProfileRevision>
    <DeviceIdentity VendorID="0x1234" DeviceID="0x5678"/>
    <InfoText>Example IO-Device - Digital Input Module</InfoText>
    <ProductName>Example DI8</ProductName>
    <OrderNumber>DI8-001</OrderNumber>
    <HardwareRelease>V1.0</HardwareRelease>
    <SoftwareRelease>V1.0</SoftwareRelease>
  </ProfileHeader>

  <DeviceAccessPointItem ID="DAP1" FixedInSlot="0" ModuleIdentNumber="0x00010001">
    <ModuleInfo>
      <Name>DAP Example Device</Name>
    </ModuleInfo>
  </DeviceAccessPointItem>

  <ModuleItem ID="Mod1" ModuleIdentNumber="0x00010002">
    <ModuleInfo>
      <Name>8 Digital Inputs</Name>
    </ModuleInfo>
    <IOData>
      <Input ByteOffset="0" BitLength="8"/>
    </IOData>
  </ModuleItem>
</GSDML>
```

Rys. 22. Fragment pliku GSDML opisującego urządzenie Profinet wraz z informacjami o module i strukturze danych wejściowych

7.9. Komunikacja S7-1200 (IO-Controller) z KEYENCE GT-2 + DL-PN1 (IO-Device)

Cel i zakres ćwiczenia

Uruchomienie komunikacji PROFINET między PLC S7-1200 a modułem komunikacyjnym DL-PN1 z czujnikiem przemieszczenia KEYENCE GT-2: instalacja GSDML, nadanie nazwy/IP (DCP), wstawienie urządzenia do projektu, mapowanie danych i weryfikacja on-line.

Sprzęt i połączenia

- PLC: Siemens S7-1200 (np. 1214C) – rola IO-Controller.
- IO-Device: DL-PN1 z podpiętym czujnikiem GT-2.
- Połączenie Ethernet: PLC ↔ (switch) ↔ DL-PN1, PC z TIA Portal jako IO-Supervisor.

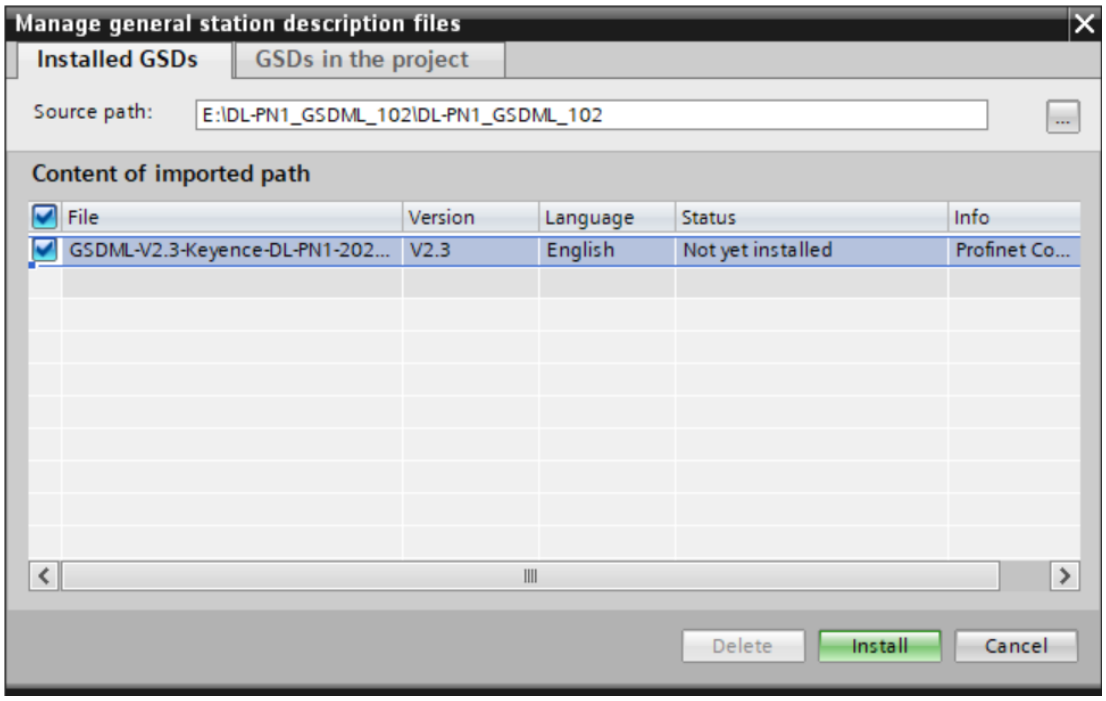
- Zasilanie czujnika i DL-PN1 zgodnie z kartą katalogową.

Instalacja pliku GSDML

Aby móc dodać urządzenie PROFInet do projektu TIA Portal należy:

1. Pobrać plik GSDML dostarczony przez producenta czujnika.
2. W TIA Portal wybrać Options → Manage general station description files (GSD).
3. Wskazać pobrany plik i wykonać instalację.
4. Po zakończeniu instalacji sprawdzić, czy urządzenie pojawiło się w katalogu sprzętowym.

[Tekst alternatywny. Zrzut ekranu przedstawia okno programu do zarządzania plikami GSDML w projekcie Profinet. W górnej części widoczna jest ścieżka źródłowa, z której importowany jest plik opisu urządzenia. W tabeli umieszczone są szczegóły dotyczące pliku, w tym jego nazwa, wersja oznaczona jako V2.3, język angielski oraz status wskazujący, że plik nie został jeszcze zainstalowany. W dolnej części okna znajdują się przyciski umożliwiające usunięcie pliku, zatwierdzenie instalacji lub anulowanie operacji.]

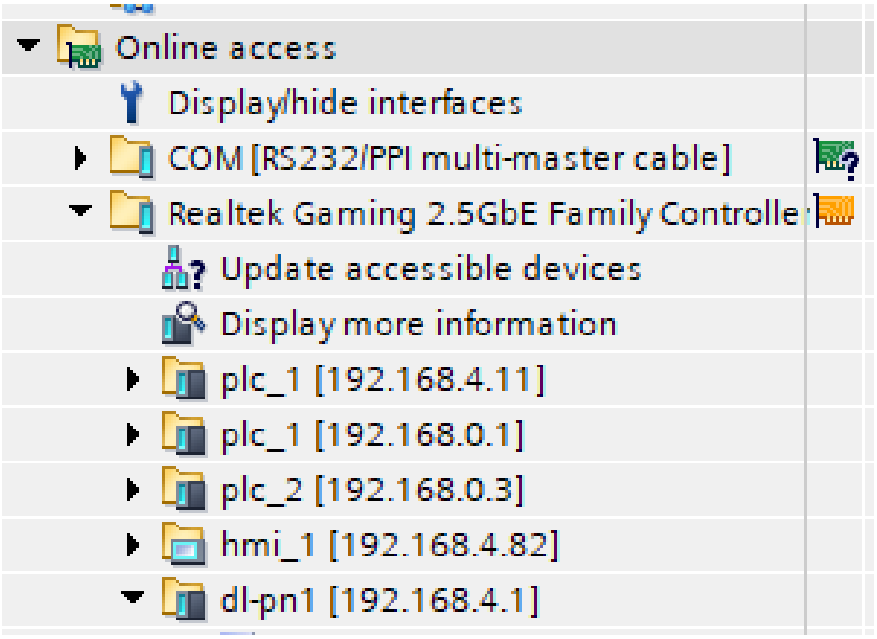


Rys 23. Okno instalacji plików GSDML w TIA Portal

Wykrycie urządzeń w sieci

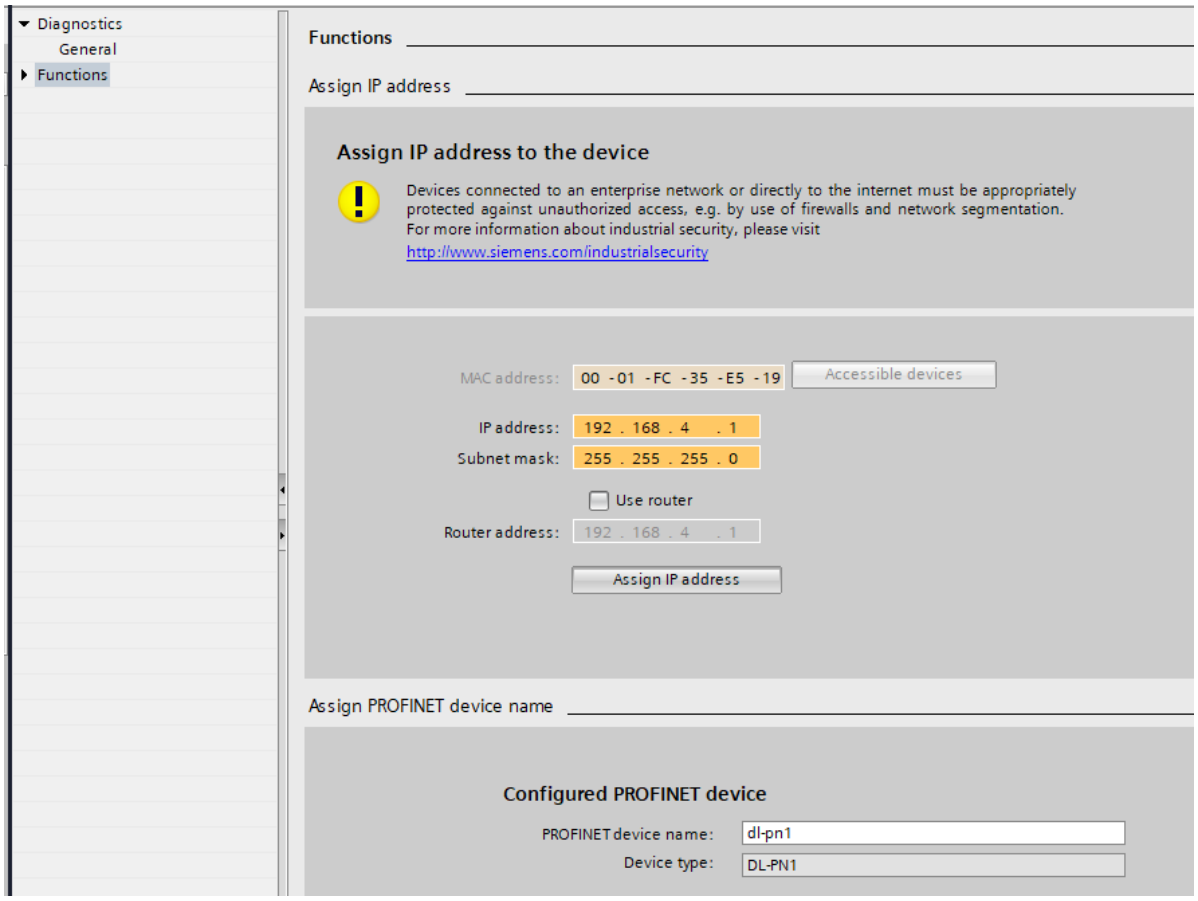
1. Otworzyć zakładkę Online access w TIA Portal.
2. Wykonać skanowanie sieci PROFINET.
3. Sprawdzić, czy moduł DL-PN1 został wykryty.
4. Nadać urządzeniu nazwę i adres IP przy użyciu DCP (funkcja „Assign device name / IP”).

[Tekst alternatywny. Zrzut ekranu przedstawia widok drzewa urządzeń w oprogramowaniu TIA Portal. W zakładce Online access widoczne są dostępne interfejsy komunikacyjne, w tym port szeregowy COM oraz karta sieciowa. Po odświeżeniu listy wyświetlone zostały urządzenia dostępne w sieci, oznaczone jako plc_1, plc_2, hmi_1 oraz dl-pn1, z przypisanymi adresami IP. Wśród nich znajdują się zarówno sterowniki PLC, panel operatorski HMI jak i moduł DL-PN1 firmy Keyence.]



Rys. 24. Lista urządzeń dostępnych w sieci Profinet

[Tekst alternatywny. Zrzut ekranu przedstawia okno programu inżynierskiego TIA Portal służące do przypisywania adresu IP urządzeniu Profinet. W górnej części wyświetlone są dane interfejsu sieciowego, w tym adres MAC oraz pola umożliwiające ręczne wprowadzenie adresu IP, maski podsieci i opcjonalnie adresu routera. Na ilustracji ustawiono adres IP 192.168.4.1 oraz maskę 255.255.255.0. W dolnej części okna widoczna jest sekcja konfiguracji urządzenia Profinet, w której przypisano nazwę dl-pn1 do urządzenia typu DL-PN1.]

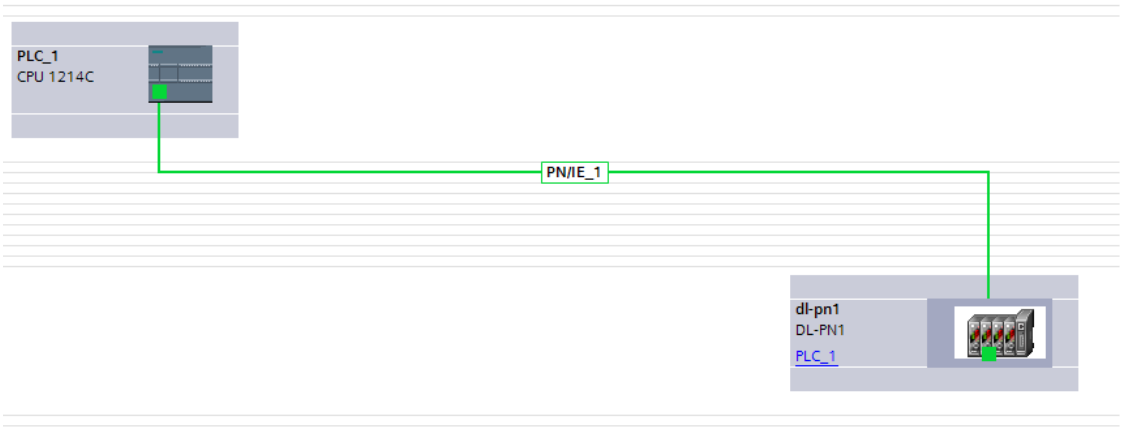


Rys. 25. Okno TIA Portal do przypisywania adresu IP i nazwy urządzenia w sieci Profinet

Konfiguracja urządzeń - Devices & Networks

1. Otworzyć edytor Devices & Networks.
2. Przeciągnąć z katalogu sprzętowego moduł DL-PN1 do obszaru projektu.
3. Połączyć interfejs PROFINET sterownika S7-1200 z modułem DL-PN1.
4. Przypisać nazwę i adres IP urządzeniu, zgodnie z konfiguracją z kroku 2.

[Tekst alternatywny. Zrzut ekranu przedstawia konfigurację sprzętową w środowisku TIA Portal, w której utworzono połączenie Profinet pomiędzy sterownikiem PLC S7-1200 oznaczonym jako PLC_1 z CPU 1214C a urządzeniem sieciowym dl-pn1 typu DL-PN1. Oba elementy połączone są zieloną linią komunikacyjną opisaną jako PN/IE_1, która symbolizuje magistralę Profinet i potwierdza poprawnie zestawioną komunikację pomiędzy kontrolerem a urządzeniem.]

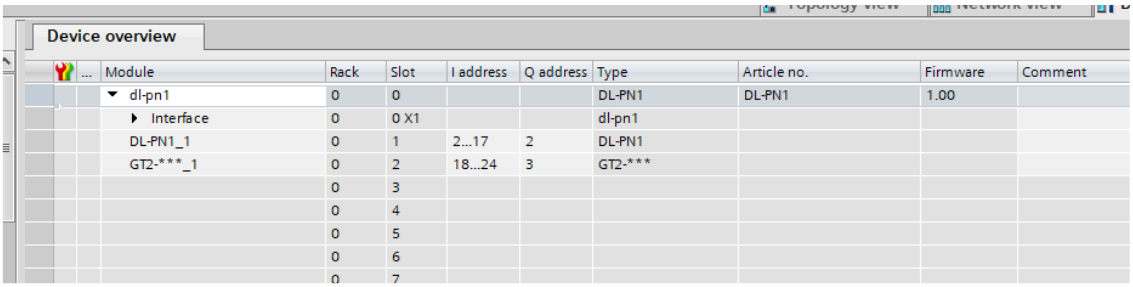


Rys 26. Konfiguracja sprzętowa w TIA Portal przedstawiająca połączenie Profinet pomiędzy sterownikiem PLC i urządzeniem IO-Device

Mapowanie wejść/wyjść

1. Otworzyć konfigurację DL-PN1 w projekcie.
2. Dodać odpowiednie moduły wejść z czujnika GT-2
3. Zanotować przypisane adresy procesowe (I-Area), aby móc je wykorzystać w tablicy PLC tags.

[Tekst alternatywny. Zrzut ekranu przedstawia zakładkę Device overview w środowisku TIA Portal, która zawiera zestawienie skonfigurowanych urządzeń i modułów Profinet. W tabeli widoczny jest moduł dl-pn1 z przypisanym interfejsem oraz kolejnymi jednostkami DL-PN1_1 i GT2-_1 umieszczonymi w poszczególnych slotach. Dla każdego modułu określono adresy wejść i wyjść procesowych. Przykładowo jednostka DL-PN1_1 zajmuje adresy wejściowe 2–17, a moduł GT2-_1 adresy 18–24. Widok ten umożliwia szybkie sprawdzenie konfiguracji i przypisania adresów w projekcie.]


Device overview

	Module	Rack	Slot	I address	Q address	Type	Article no.	Firmware	Comment
	dl-pn1	0	0			DL-PN1	DL-PN1	1.00	
	Interface	0	0 X1			dl-pn1			
	DL-PN1_1	0	1	2...17	2	DL-PN1			
	GT2-***_1	0	2	18...24	3	GT2-***			
		0	3						
		0	4						
		0	5						
		0	6						
		0	7						

Rys. 27. Zestawienie modułów i przypisanych adresów w oknie Device overview w TIA Portal

Wgrywanie projektu i obserwacja on-line

1. Wgrać konfigurację sprzętową i program do sterownika S7-1200.
2. Otworzyć tablicę PLC tags i włączyć tryb monitorowania.
3. Sprawdzić zmiany wartości surowego sygnału INT podczas mechanicznego przesuwania czujnika.
4. Zaobserwować zmiany bitów statusowych w zależności od stanu GT-2.

[Tekst alternatywny. Zrzut ekranu przedstawia zakładkę PLC tags w środowisku TIA Portal, w której zdefiniowane zostały znaczniki wykorzystywane w programie sterownika PLC. Każdy wpis zawiera nazwę znacznika, tabelę, typ danych, przypisany adres w pamięci sterownika oraz dodatkowe właściwości, takie jak dostęp do zapisu, widoczność czy możliwość monitorowania wartości online. Na ilustracji widoczne są znaczniki od Tag_1 do Tag_2(1), przypisane kolejno do bajtów pamięci od %IB18 do %IB24. W kolumnie Monitor value zaprezentowane są aktualne wartości poszczególnych znaczników, zapisane w formacie szesnastkowym.]

PLC tags

	Name	Tag table	Data type	Address	Retain	Acces...	Writa...	Visibl...	Monitor value
	Tag_1	Default tag table	Byte	%IB18	☐	☑	☑	☑	16#01
	Tag_2	Default tag table	Byte	%IB19	☐	☑	☑	☑	16#00
	Tag_3	Default tag table	Byte	%IB20	☐	☑	☑	☑	16#00
	Tag_4	Default tag table	Byte	%IB21	☐	☑	☑	☑	16#00
	Tag_5	Default tag table	Byte	%IB22	☐	☑	☑	☑	16#01
	Tag_6	Default tag table	Byte	%IB23	☐	☑	☑	☑	16#4F
	Tag_2(1)	Default tag table	Byte	%IB24	☐	☑	☑	☑	16#AD

Rys 28. Tabela znaczników PLC w TIA Portal z przypisanymi adresami i monitorowaniem wartości w czasie rzeczywistym

Kalibracja i skalowanie wartości

1. Odczytać wartości INT w położeniach granicznych. W prezentowanym przykładzie dla minimalnego położenia 0mm wartość odczytana to 0 a dla maksymalnego 12,5mm odczytana wartość to 506.

[Tekst alternatywny. Fotografia przedstawia moduł pomiarowy Keyence podłączony do sieci Profinet, widoczny w szafie sterowniczej. Urządzenie wyposażone jest w wyświetlacz LED, na którym prezentowane są wartości pomiarowe. Po lewej stronie zdjęcia odczyt wynosi 24,40 – jest to wartość odpowiadająca minimalnemu sygnałowi wejściowemu. Po prawej stronie zaprezentowano sytuację dla maksymalnego sygnału wejściowego, gdzie wyświetlacz pokazuje wartość 1976,9. Moduł komunikacyjny powyżej

wyświetlacza, z podłączonym przewodem Ethernet, zapewnia integrację z siecią Profinet.]

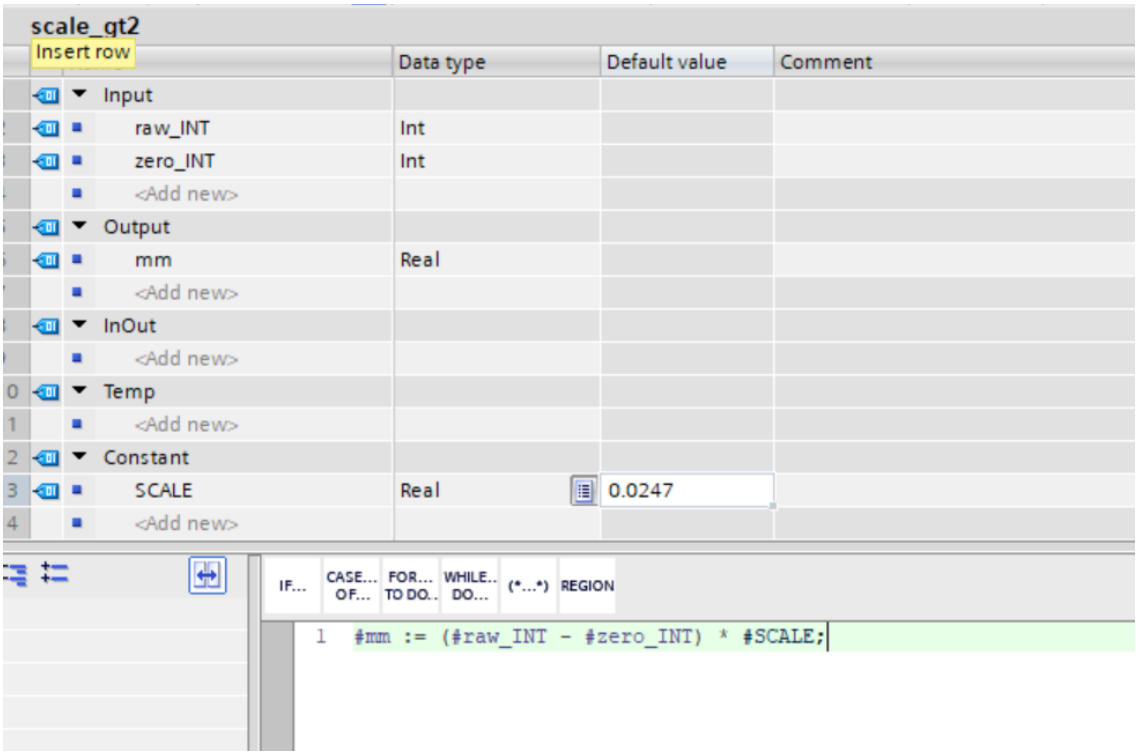
Rys. 29. Odczyty z modułu Keyence w sieci Profinet dla wartości minimalnej (po lewej) i maksymalnej (po prawej)

2. Obliczyć współczynnik skali:

$$SCALE = \left(\frac{12.5mm}{506}\right) \approx 0.0247mm \quad (1)$$

3. Zaimplementować w programie przeliczanie wartości.

[Tekst alternatywny. Zrzut ekranu przedstawia fragment programu w TIA Portal, w którym zdefiniowany został blok funkcyjny o nazwie scale_gt2. W bloku zadeklarowano zmienne wejściowe raw_INT i zero_INT typu Int, zmienną wyjściową mm typu Real oraz stałą SCALE o wartości 0,0247. W dolnej części okna widoczna jest instrukcja przypisania w języku SCL, w której wartość wyjściowa mm obliczana jest jako różnica pomiędzy raw_INT i zero_INT pomnożona przez współczynnik skalowania SCALE. Dzięki temu możliwa jest konwersja wartości surowych z czujnika na jednostki fizyczne w milimetrach.]



Rys. 30. Implementacja bloku skalującego wartości surowe z czujnika na jednostki fizyczne w TIA Portal

8. Literatura

1. Solnik W., Zajda Z. (2018). Sieci przemysłowe Profibus DP, ProfiNet, AS-i i EGD. Przykłady zastosowań. Wydawnictwo BTC, Warszawa.
2. Mystkowski A. (2012). Sieci przemysłowe PROFIBUS DP i PROFINET IO. Oficyna Wydawnicza Politechniki Białostockiej
3. Świszcz P. (red.), Dębowski K., Grabowski D. (2011). Laboratorium przemysłowych sieci komunikacyjnych. Część I. Wydawnictwo Politechniki Śląskiej.
4. Mielczarek W. (2016). Szeregowe interfejsy cyfrowe. Helion, Gliwice.
5. Koch R., Luftner R. (2019). Communication Networks in Automation. Publicis MCD Verlag
6. Modbus Organization (2012). Modbus Application Protocol Specification V1.1b3. Modbus-IDA, North Grafton, MA.
7. Siemens AG (2022). PROFINET w automatyce procesowej – podręcznik użytkownika. Siemens