



POLITECHNIKA ŚLĄSKA

WYDZIAŁ AUTOMATYKI, ELEKTRONIKI I INFORMATYKI

KIERUNEK INFORMATYKA

Praca dyplomowa magisterska

Projekt i realizacja stanowiska laboratoryjnego do badania zależności
czasowych w sieci EtherCAT

Autor: inż. Damian Karbowski

Kierujący pracą: dr inż. Jacek Stój

Gliwice, 16 września 2013

Streszczenie

Głównym celem pracy było zaprojektowanie i zrealizowanie stanowiska laboratoryjnego do zbadania zależności czasowych w protokole EtherCAT. Należało wykorzystać dostępny na uczelni w laboratorium sprzęt i przedstawić na nim obrazowo, że w protokole tym występują opóźnienia. Podstawowym zadaniem było stworzenie konfiguracji, oprogramowania sterownika swobodnie programowalnego oraz wizualizacji wykorzystując dostępne silniki i pozostałe elementy stanowiska do pokazania zależności czasowych w protokole EtherCAT.

W ramach pracy oprócz projektu i wykonania stanowiska, wykonano badania dotyczące opóźnień w protokole EtherCAT i zaproponowano rodzaje eksperymentów badawczych możliwych do zrealizowania na stanowisku. Po zakończeniu projektu posłuży on jako podstawa do stworzenia i przeprowadzenia ćwiczeń laboratoryjnych prowadzonych w ramach działalności dydaktycznej pracowników Zespołu Przemysłowych Zastosowań Informatyki.

Określono i objaśniono dziedzinę, z której wywodzi się temat. Praca zdaniem autora stanowi dobrą podstawę do poznania i zrozumienia zasad działania protokołu EtherCAT. Opisane zostały różne metody realizacji urządzeń pracujących w tym standardzie. Przeprowadzone badania udowodniły, że protokół jest bardzo nowoczesny i spełnia wszystkie wymagania stawiane protokołom przemysłowym.

W końcowej części dokument zawiera podsumowanie oraz wnioski wyciągnięte z przeprowadzonych badań. Podczas realizacji zostały rozwiązane problemy charakterystyczne dla produktów firmy Beckhoff, z których najważniejszym zdaniem autora było skonfigurowanie i poprawne uruchomienie silników. Przedstawione zostały również liczne możliwe do zrealizowania i przeprowadzenia w przyszłości eksperymenty. Pozwolą one na pogłębienie wiedzy o protokole, jak również sprawdzenie opóźnień czasowych występujących w protokole w inny sposób.

Spis treści

1	Wstęp	3
1.1	Geneza	3
1.2	Przemysłowe systemy czasu rzeczywitego	4
1.3	Stanowisko laboratoryjne	10
1.3.1	Sterownik PLC	12
1.3.2	Komputer	13
1.4	Plan pracy	14
2	Analiza tematu	15
2.1	EtherCAT	15
2.1.1	Przetwarzanie „w locie”	16
2.1.2	Telegram EtherCAT	18
2.1.3	Topologia	21
2.1.4	Warstwa fizyczna	22
2.1.5	Synchronizacja	22
2.1.6	Realizacja węzłów EtherCAT	23
2.1.7	EtherCAT Technology Group	25
2.1.8	Maszyna stanów EtherCAT	25
3	Stanowiska badawcze	28
3.1	Podstawowe stanowiska	28
3.2	Rozbudowane stanowisko	32
4	Badania	35
4.1	Czas stabilizacji sieci po odłączeniu i podłączeniu dowolnego urządzenia	35
4.1.1	Pojedyncze urządzenie	36
4.1.2	Wyspa z modułami I/O	37
4.1.3	Wszystkie węzły sieci	38
4.1.4	Pełna stabilizacja urządzenia	39
4.1.5	Problemy	42
4.2	Opóźnienie transmisji	43
4.2.1	Badanie różnic czasu	43
4.2.2	Badanie różnic odległości	45

5	Uruchamianie i testowanie	46
5.1	Napotkane problemy	46
6	Wnioski	49
7	Bibliografia	53
8	Spis rysunków, tablic i kodów źródłowych	56
8.1	Spis rysunków	56
8.2	Spis tablic	57
8.3	Spis kodów źródłowych	57
9	Załączniki	58

1 Wstęp

Tematem, którego dotyczy ta praca jest: „Projekt i realizacja stanowiska laboratoryjnego do badania zależności czasowych w sieci EtherCAT”. Zagadnienia związane z tworzeniem oprogramowania dla sterowników przemysłowych są dla autora niezwykle interesujące, a zrealizowany projekt miał na celu dalsze pogłębienie jego wiedzy z tego zakresu. Wyboru tego konkretnego tematu autor dokonał, ponieważ protokół EtherCAT jest jeszcze nowością i według wielu źródeł stanowi przyszłość branży informatyki przemysłowej [11, 12], a praca nad tym tematem wydaje się być pomocna i wartościowa w przyszłej pracy zawodowej lub na ewentualnym dalszym etapie kształcenia.

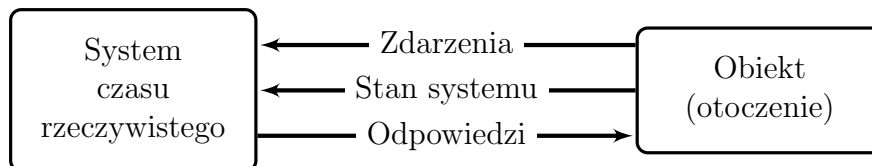
1.1 Geneza

Informatyka przemysłowa to dziedzina wiedzy z pogranicza nauk informatycznych oraz szeroko pojętych nauk o technologiach przemysłowych łącząca te dwa zakresy wiedzy. Informatyka przemysłowa to dział, który jest niezwykle pomocny, zwłaszcza przy analizach danych, ale również podczas tworzenia funkcjonalnych i wydajnych maszyn dla nowoczesnych gałęzi gospodarki. Specjaliści z zakresu informatyki przemysłowej zajmują się takimi zagadnieniami jak monitorowanie produkcji przemysłowej czy też analizowaniem wybranych danych, które później służą do porównań. Niezwykle ważne jest wykorzystanie tych danych, które dostarczają istotnych informacji dla każdego przemysłu, a także sterowanie produkcją przemysłową, która pomaga w dynamicznym rozwoju każdej gałęzi w gospodarce. Przemysł to szczególna gałąź gospodarki, która aby się rozwijać potrzebuje nowoczesnych rozwiązań. Jedną z nich jest informatyka, która pozwala na udoskonalanie wyprodukowanych już maszyn przemysłowych, a także tych, które dopiero są w trakcie rozwoju technologicznego. Informatyka to aspekt, który zwraca uwagę na nowoczesne zastosowanie komputerów w celach przemysłowych. Wielu informatyków współpracuje z firmami działającymi w przemyśle, gdyż współpraca opłaca się obydwu stronom.

Systemy komputerowe realizują bardzo odpowiedzialne zadania, a współczesna technologia stawia coraz poważniejsze wymagania związane przede wszystkim z gwarantowanym i nieprzekraczalnym czasem realizacji pojedynczego cyklu sterowania bądź regulacji, co wymaga wprowadzenia rozwiązań gwarantujących determinizm czasowy. Determinizm czasowy oznacza, że wyniki działania systemu komputerowego pojawiają się w określonym i skończonym czasie.

1.2 Przemysłowe systemy czasu rzeczywistego

System czasu rzeczywistego (ang. *Real-Time System*, w skrócie *RTS*) według jednej z definicji jest to system komputerowy, w którym obliczenia są prowadzone równolegle z przebiegiem zewnętrznego procesu w otoczeniu systemu (w sterowanym obiekcie). Mają one na celu nadzorowanie, sterowanie oraz terminowe reagowanie na zdarzenia zachodzące w tym procesie. Uproszczony schemat działania przedstawiono na Rysunku 1.



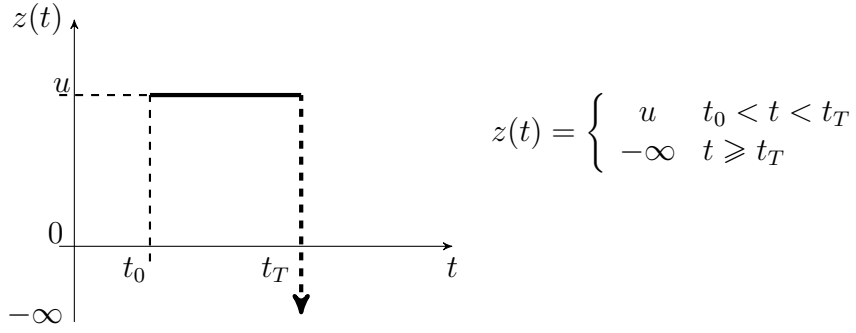
Rysunek 1: Schemat działania systemu czasu rzeczywistego.

System czasu rzeczywistego jest takim systemem, który do poprawnego działania musi spełniać następujące warunki:

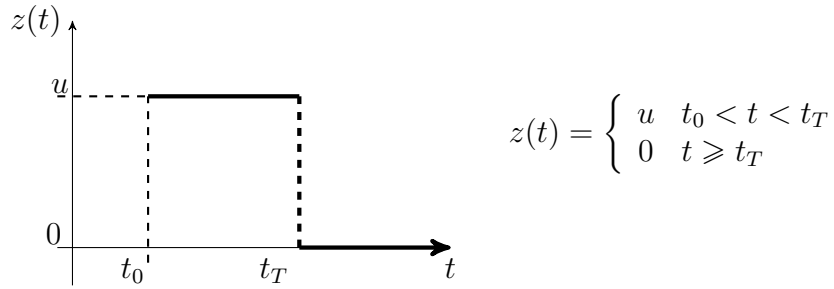
1. Warunki logiczne – odpowiedź systemu musi być zawsze prawidłowa przy uwzględnieniu stanu systemu, tzn. odpowiedź zależy od stanu otoczenia i występujących zdarzeń oraz jest zgodna z założeniami technologicznymi,
2. Warunki czasowe – odpowiedź systemu musi nadejść we właściwym czasie (przed przekroczeniem czasu granicznego dla danego systemu).

Jak wynika z przedstawionych na Rysunku 2 teoretycznych funkcji zysku, rozróżniamy trzy typy systemów czasu rzeczywistego:

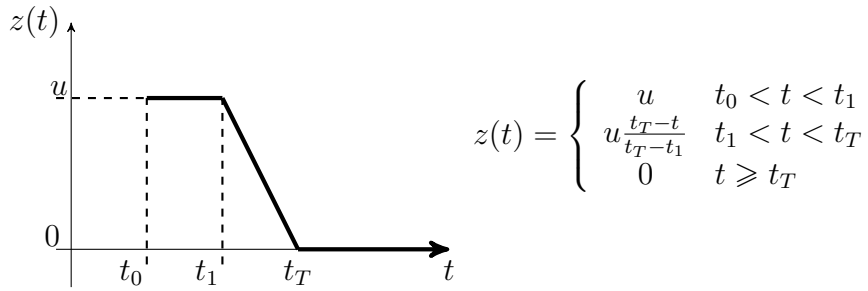
- systemy o ostrych ograniczeniach czasowych (ang. *hard real-time*) – gdy przekroczenie terminu powoduje poważne, a nawet katastrofalne skutki, jak np. zagrożenie życia lub zdrowia ludzi, uszkodzenie lub zniszczenie urządzeń, przy czym nie jest istotna wielkość przekroczenia terminu a jedynie sam fakt jego przekroczenia. Wynika z tego, że informacja wygenerowana przez system po przekroczeniu czasu granicznego t_T oznacza poważny błąd systemu,
- systemy o mocnych ograniczeniach czasowych (ang. *firm real-time*) – gdy fakt przekroczenia terminu powoduje całkowitą nieprzydatność wypracowanego przez system wyniku, jednakże nie oznacza to zagrożenia dla ludzi lub sprzętu; pojęcie to stosowane jest głównie w opisie teoretycznym baz danych czasu rzeczywistego,
- systemy o miękkich lub łagodnych ograniczeniach czasowych (ang. *soft real-time*) – gdy przekroczenie pewnego czasu powoduje negatywne skutki tym poważniejsze, im bardziej ten czas został przekroczony; w tym przypadku przez „negatywne skutki” rozumie się spadek funkcji zysku aż do osiągnięcia wartości zero.



(a) System typu „hard”



(b) System typu „firm”



(c) System typu „soft”

Rysunek 2: Funkcja zysku dla systemów czasu rzeczywistego.

Dla określenia podziału systemów czasu rzeczywistego przydatne jest pojęcie funkcji zysku [3]. Jest to funkcja zależna głównie od czasu i dokonuje odwzorowania czasu wykonania zadania na korzyść wynikającą z jego wykonania. Wartość korzyści niekoniecznie jest wielkością wymiarowaną. Źródłem występujących ograniczeń czasowych są zazwyczaj zjawiska fizyczne zachodzące w świecie rzeczywistym kontrolowanym przez system. Zadanie jest uznawane za zrealizowane jako poprawne, jeśli w chwili jego zakończenia wartość funkcji zysku jest większa od zera.

Założenia dla wykresów funkcji zysku [3]:

- t_0 – jest to moment w którym zadanie zostało zlecone oraz rozpoczęło się jego wykonywanie,
- t_T – jest to graniczny moment w którym przetwarzanie może zostać zakończone,
- w przypadku jak na Rysunku 2c funkcja $z(t)$ jest funkcją ciągłą w punktach t_1 i t_T ,
- dla wartości $t \leq t_0$ funkcji zysku nie wyznacza się, gdyż jest to sytuacja niemożliwa, tzn. obliczany byłby zysk z zakończenia zadania przed jego rozpoczęciem, co nie może się zdarzyć i nie ma sensu logicznego.

W tego typu systemach przekształcanie danych wymienianych między nim, a środowiskiem zewnętrznym zachodzi w deterministycznie określonym czasie. W opisach stosuje się pojęcie terminu (ang. *deadline*), oznaczające najdłuższy dopuszczalny czas reakcji systemu na wystąpienie zdarzenia [3]. Szybkość działania systemu czasu rzeczywistego nie jest tak naprawdę ważna, istotne jest, aby spełnione były bezwarunkowo przyjęte ograniczenia czasowe.

Idealnymi przykładami praktycznymi z życia codziennego wyjaśniającymi wprowadzony podział oraz funkcję zysku są:

- W przypadku systemów typu „hard real-time” jest to reaktor jądrowy. Zakładając, że system nadzorujący pracę takiego urządzenia nie zdążyłby wystarczająco szybko zareagować na podnoszenie się temperatury rdzenia i doszłoby do jego przegrzania lub nawet do częściowego stopienia. Jak pokazał przykład awarii reaktora jądrowego w Czarnobylu tego typu zjawiska mają tragiczne i wieloletnie skutki. Jest to idealny przykład, że nie pojawienie się odpowiedzi systemu na czas może przynieść poważne konsekwencje i nie ma znaczenia o ile czas graniczny został przekroczony,
- W przypadku systemów typu „soft real-time” jest to bankomat. Jak wiadomo bankomat musi zareagować na żądania zgłaszane przez klienta w skończonym czasie. Jeżeli czas ten będzie zbyt długi, tzn. będzie przekraczał czas graniczny to negatywnym skutkiem będzie spadające zadowolenie klienta wynikające z jakości świadczonych usług. Wiadomo, że nie wolno ignorować tego faktu, bo niezadowolenie klienta może spowodować, że nie skorzysta on więcej z danej sieci bankomatów. Na tym przykładzie dobrze widać, że przekroczenie czasu granicznego nie jest poważne w skutkach, ale powoduje spadek wartości funkcji zysku, którą jest w tym przypadku zadowolenie klienta.

Systemy czasu rzeczywistego najczęściej pracują pod kontrolą specjalnych systemów operacyjnych spełniających reżimy czasu rzeczywistego. System operacyjny czasu rzeczywistego (ang. *Real-Time Operating System*, w skrócie *RTOS*) jest to system operacyjny, który został opracowany tak, aby spełniać wymagania narzucone na czas wykonania zadanych operacji. Podstawowym wymaganiem dla systemu typu RTOS jest określenie najdłuższego możliwego czasu, po którym urządzenie wypracuje odpowiedź na wystąpienie zdarzenia (określenie przypadku pesymistycznego).

Zaawansowanym problemem występującym w systemach operacyjnych tego typu jest dobór algorytmu szeregowania oraz przydziału czasu. W systemie czasu rzeczywistego ze względu na ograniczone zasoby trzeba wybrać, który z procesów będzie miał w danym momencie przydzielony procesor oraz jak długo przydział taki powinien trwać, aby wszystkie wykonywane procesy spełniły określone dla nich ograniczenia czasowe.

Kolejnym niezbędnym elementem, oprócz układów wejściowych i wyjściowych oraz kontrolowanych urządzeń (np. robotów) są deterministyczne sieci transmisyjne. W sieciach przemysłowych dane są często przesyłane, ale są one krótkie (mają niewielki rozmiar), w przeciwieństwie do sieci ogólnego przeznaczenia, gdzie przesyła się dane rzadko i w dużej ilości. Dane mają charakter np. wartości pomiaru, rozkazów start/stop, alarmu, przekazania wartości zadanej do zrealizowania przez regulator itd. Bardzo często informacja przesyłana w tych sieciach ma rozmiar pojedynczego bitu lub słowa.

W sieciach, w których zagwarantowany jest determinizm czasowy transmisji, okres od wysłania pakietu danych od nadawcy do jego odbioru po stronie odbiorcy jest z góry znany i pozostaje niezmienny. Wówczas komunikacja między węzłami sieci może być realizowana w czasie rzeczywistym, łatwiejsza jest też ich synchronizacja. Obie te kwestie są kluczowe dla sprawnego działania systemów sterowania. Dodatkowo sieci takie muszą zapewniać wszelkiego rodzaju mechanizmy kontroli poprawności przesyłu danych oraz zabezpieczać przez ewentualną ich utratą. Kolejnym wymaganiem stawianym sieciom tego typu jest odporność na zakłócenia zewnętrzne, w tym szczególnie zakłócenia elektromagnetyczne. Rola komunikacji jest krytyczna z punktu widzenia technologii, ekonomii i bezpieczeństwa prowadzenia procesu.

Aby sieć mogła być deterministyczna musi stosować specyficznym sposób transmisji danych. Istnieją trzy modele transmisji gwarantujące spełnienie wymogów determinizmu czasowego [1, 2]:

- Master–Slave,
- Token,
- Producent-Dystrybutor-Konsument, w skrócie PDK.

Model Master–Slave bazuje na wymianie informacji pomiędzy węzłem nadrzędnym (ang. *Master*), a węzłami podrzędnymi (ang. *Slave*). Stacja master jest odpowiedzialna za kontrolowanie ruchu sieciowego. Wymiany odbywają się cyklicznie zgodnie ze scenariuszem. Wymiany zawsze odbywają się pomiędzy węzłami Master i Slave, w przypadku wymiany danych między węzłami podrzędnymi, ruch odbywa się pośrednio przez węzeł nadrzędny.

Model Token – wszystkie stacje w sieci są równorzędne. Prawo do nadawania ma w danym momencie tylko jeden abonent posiadający żeton (ang. *Token*). Żeton jest wymieniany między kolejnymi węzłami (ang. *token passing*). Stacja, która otrzyma żeton ma prawo nadawać przez określony czas, po czym bezwzględnie musi przekazać go dalej. Sieć oparta o ten model może być zbudowana w topologii pierścienia (ang. *Token Ring*) lub (ang. *Token Bus*)

Model PDK – istnieją w nim trzy rodzaje stacji. Zgodnie z nazwą są to:

- Producent – jest odpowiedzialny na poziomie aplikacji za „produkcję” danych, które mogą być periodyczne lub nie, synchroniczne lub nie, z innymi aplikacjami lub „produkcjami” danych.
- Dystrybutor – przechowuje scenariusz wymian i na jego podstawie określa, kiedy jaka informacji ma być obsłużona przez sieć. Dystrybutor w przeciwieństwie do stacji Master nie inicjuje samodzielnie transmisji danych użytecznych ani ich nie retransmituje. Określa jedynie, kiedy producent danej informacji ma zrealizować zapis rozgłoszeniowy odpowiednich danych w sieci.
- Konsument – jest aplikacją, która będąc wykonywaną, żąda danych. I znów, użycie danych może być periodyczne lub nie i synchroniczne bądź nie, z innymi aplikacjami.

Zdaniem autora EtherCAT jest modelem hybrydowym. Nie da się go w prosty sposób przypisać do żadnego z opisanych tutaj modeli. Rozwiązanie jest nowoczesne i innowacyjne dlatego nie zostało oparte na modelach wykorzystywanych przy tworzeniu starszych rozwiązań. Teoretycznie w opisach działania protokołu występują pojęcia master oraz slave, ale w topologii protokołu EtherCAT może występować wiele węzłów nadrzędnych, węzły podrzędne mogą żądać transmisji oraz nie ma sztywnego scenariusza wymian. Węzeł zarządzający może zatrzymać cały ruch w sieci, zawiesić działanie dowolnego węzła lub zażądać od niego danych. Porównując zasady działania do modelu z żetonem węzeł chcący nadać informacje musi oczekiwać, aż do momentu wykrycia fragmentu telegramu przeznaczonego dla niego, co zdaniem autora przypomina koncepcję żetonu. Niestety jednak z powodu mechanizmu organizacji wymiany danych zależnie od zapotrzebowania wszystkich węzłów, nie jest stały i dobrze znany moment uzyskania prawa do nadawania. Ostatni

z przedstawionych modeli najlepiej pasuje do rozwiązań zastosowanych w standardzie EtherCAT. Węzeł nadrzędny pełni tutaj rolę dystrybutora, a wszystkie pozostałe węzły wraz z nim pełnią rolę producenta i konsumenta lub tylko jedną z nich. Powyższa analiza udowadnia, że sieć EtherCAT posiada niektóre cechy każdego z przedstawionych modeli. Podsumowując zdaniem autora zasada zapewniania determinizmu czasowego w protokole EtherCAT jest najbliższa modelowi Producent-Dystrybutor-Konsument. Zdecydowanie najistotniejszą innowacją mającą na celu poprawę parametrów czasowych jest zastosowana metoda transmisji danych pozwalająca uzyskać wysokie wykorzystanie kanału transmisyjnego.

Podsumowanie najważniejszych wymagań stawianych przemysłowym sieciom informatycznym:

- Niezawodność,
- Przewidywalność procesu komunikacji:
 - Relacja pomiędzy węzłami w sieci,
 - Praca w czasie rzeczywistym,
- Efektywność w przekazywaniu krótkich wiadomości,
- Standaryzacja interfejsów, łatwość podłączania dużej liczby urządzeń,
- Możliwość podłączenia do zewnętrznych sieci,
- Łatwość lokalizowania usterek.

Aby system składający się z wielu komponentów był systemem czasu rzeczywistego, konieczne jest spełnianie wymogów przez każdy z elementów składowych. W przypadku systemów informatycznych oznacza to, że zarówno sprzęt, system operacyjny, jak i oprogramowanie aplikacyjne muszą gwarantować dotrzymanie zdefiniowanych ograniczeń czasowych.

Z powyższego opisu wynika, że systemy czasu rzeczywistego są bardzo ważnym elementem nie tylko w przemyśle, ale również w życiu codziennym. Przemysłowe sieci komputerowe są bardzo istotnym elementem tych systemów, dlatego właśnie uzasadnione jest, aby przebadać i poznać je dokładnie oraz ustalić jakie zależności czasowe w nich występują. Sieć EtherCAT będąca tematem niniejszej pracy spełnia większość wymagań stawianych nowoczesnym protokołom pracującym w warunkach przemysłowych, dlatego autor uważa, że warto przyjrzeć mu się bliżej i przeprowadzić zaplanowane badania oraz znaleźć ewentualne perspektywy na kolejne prace badawcze w przyszłości.

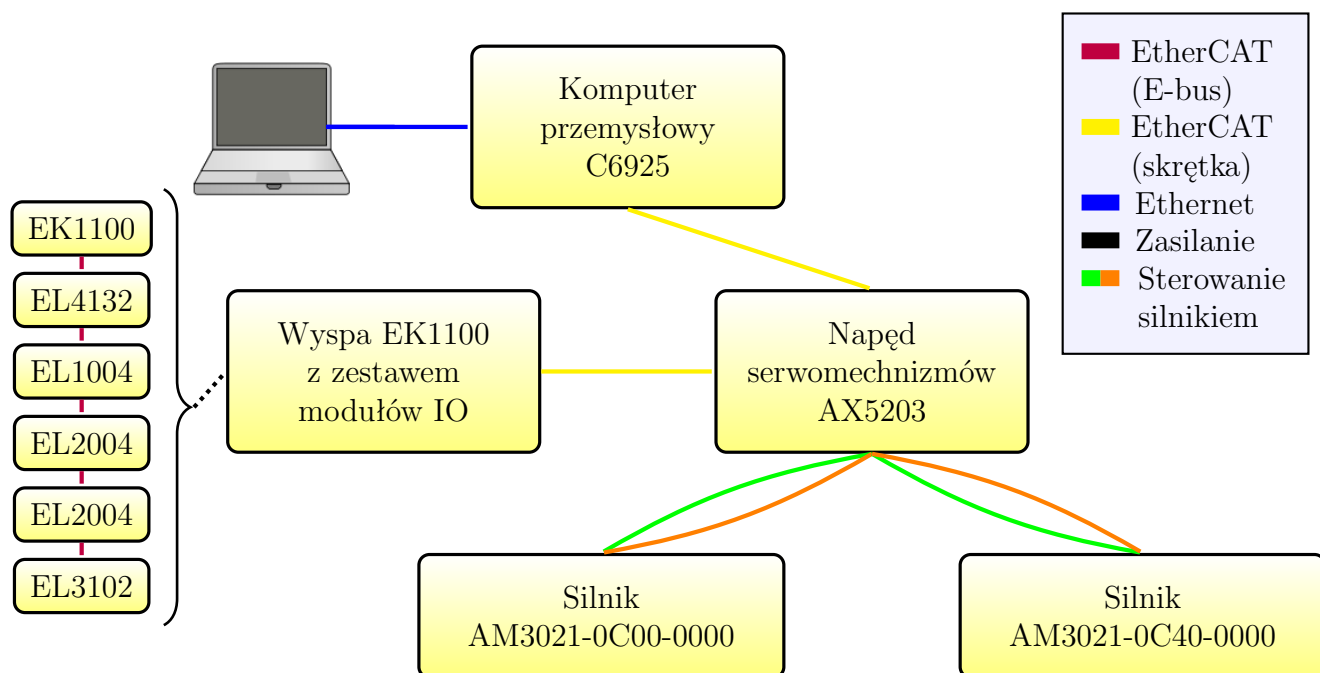
1.3 Stanowisko laboratoryjne

Dostępne stanowiska wyposażone w urządzenia firmy Beckhoff należało uruchomić i przystosować do pracy umożliwiającej zbadanie zależności czasowych w protokole EtherCAT. Dokładny opis wykonanych prac zostanie przedstawiony w Rozdziale 3. Na potrzeby realizacji projektu wykorzystano dwa różne istniejące stanowiska laboratoryjne, które składały się z elementów opisanych w Tablicy 1.

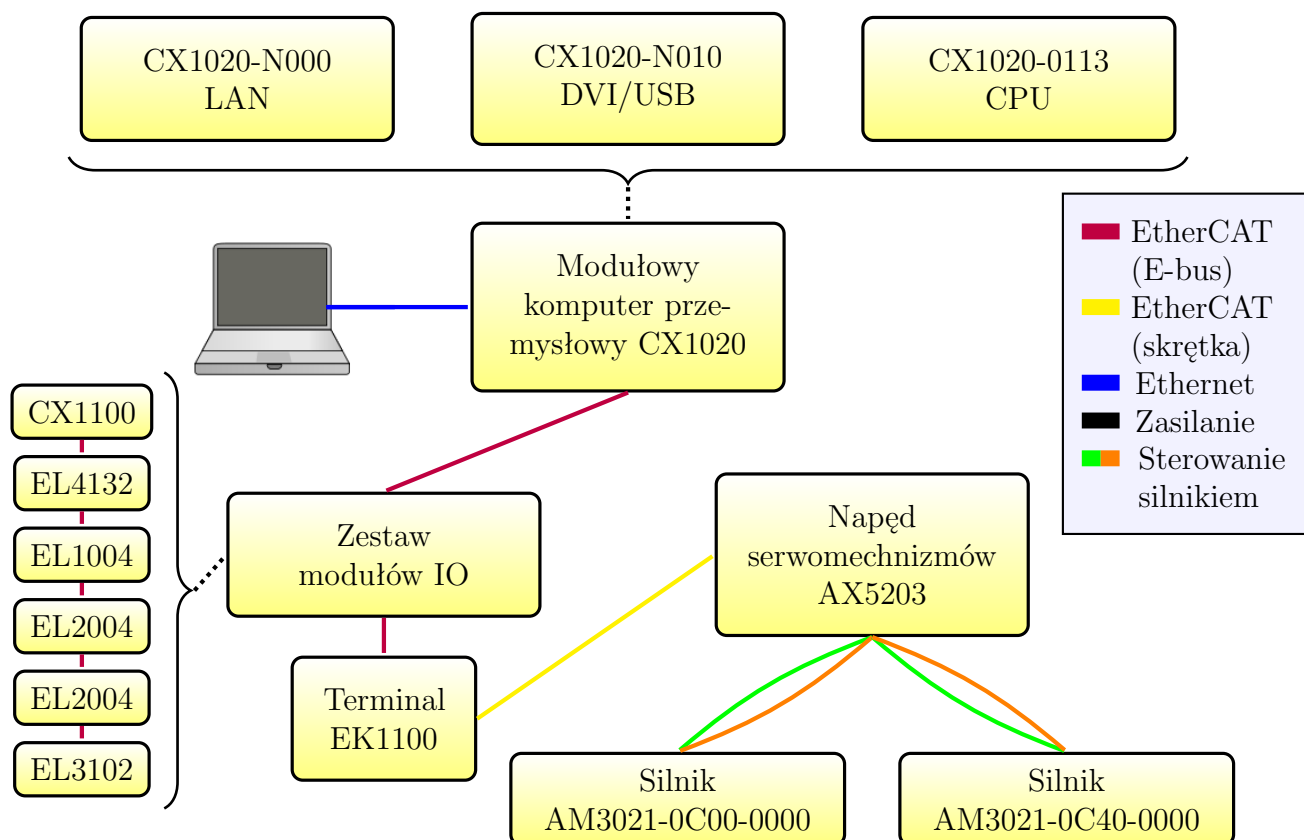
Stanowisko typu CP (Rysunek 3) Adres IP: 157.158.57.121	Stanowisko typu CX (Rysunek 4) Adres IP: 157.158.57.47
<ol style="list-style-type: none"> 1. Serwomechanizm AM3021-0C00-0000. 2. Serwomechanizm AM3021-0C40-0000. 3. Wyspa EK1100 z zestawem modułów IO: <ul style="list-style-type: none"> • Terminal sieci EtherCAT EK1100. • 2-kanalowy moduł wyjść analogowych EL4132, • 4-kanalowy moduł wejść cyfrowych EL1004, • 2 4-kanalowe moduły wyjść cyfrowych EL2004, • 2-kanalowy moduł wejść analogowych EL3102. 4. Napęd serwomechanizmów AX5203 (2 osiowy). 5. Komputer przemysłowy C6925. 6. Zasilacz. 	<ol style="list-style-type: none"> 1. Serwomechanizm AM3021-0C00-0000. 2. Serwomechanizm AM3021-0C40-0000. 3. Zestaw modułów IO: <ul style="list-style-type: none"> • 2-kanalowy moduł wyjść analogowych EL4132, • 4-kanalowy moduł wejść cyfrowych EL1004, • 2 4-kanalowe moduły wyjść cyfrowych EL2004, • 2-kanalowy moduł wejść analogowych EL3102. 4. Terminal sieci EtherCAT EK1100. 5. Napęd serwomechanizmów AX5203 (2 osiowy). 6. Modułowy komputer przemysłowy CX1020: <ul style="list-style-type: none"> • Interfejs USB/DVI CX1020-N010 , • Ethernet CX1020-N000, • CPU CX1020-0113, • Zasilacz CPU i magistrali I/O CX1100. 7. Zasilacz.

Tablica 1: Dostępne stanowiska laboratoryjne.

Wszystkie połączenia oznaczone na schematach jako EtherCAT tworzą jedną sieć. Autor celowo i świadomie rozróżnił dwa typy połączeń, aby zwrócić uwagę na zastosowanie na stanowiskach różnych medium transmisyjnych. Linie oznaczone jako skrętka są widocznym na stanowisku przewodem ethernetowym, natomiast łącze E-bus jest niewidoczne, ponieważ znajduje się na składaniu modułów.



Rysunek 3: Schemat stanowiska typu CP.



Rysunek 4: Schemat stanowiska typu CX.

Stanowiska podłączone są do sieci lokalnej Ethernet w laboratorium, więc komunikacja z nimi odbywa się tak samo jak z każdym innym urządzeniem sieciowym. Podstawy programowania i korzystania ze sterowników autor poznał zapoznając się z odpowiednią literaturą [4–8] oraz uczęszczając w toku studiów na zajęcia obowiązkowe oraz specjalizacyjne.

1.3.1 Sterownik PLC

W realizacji wykorzystane zostały stanowiska firmy Beckhoff wyposażone w jednostki centralne pracujące pod kontrolą systemu Windows CE (Microsoft Windows Compact Edition). Na jednostce takiej uruchamiane są programy do sterowania z poziomu komputera (ang. *Soft PLC*). Jest to rozwiązanie alternatywne dla klasycznych sterowników swobodnie programowalnych w postaci dedykowanego urządzenia (ang. *Hard PLC*), nazywanych przez niektórych prawdziwymi (ang. *True PLC*). Koncepcja ta powstała i jest rozwijana, ponieważ te klasyczne sterowniki posiadają zbyt małe możliwości obliczeniowe oraz szybkość działania jednostki centralnej. W tradycyjnych rozwiązaniach niestety zwiększanie tych możliwości (ilość dostępnej pamięci oraz szybkości działania) powoduje bardzo szybki wzrost ceny gotowego urządzenia. Niezbędnym elementem konfiguracji zestawu, który zostaje przekształcony w „soft PLC” jest karta komunikacyjna umożliwiająca połączenie sterownika z modułami sygnałowymi i wykonawczymi na obiekcie z wykorzystaniem sieci przemysłowej. Przykładowe zastosowanie programu do sterowania z poziomu komputera zostało szczegółowo opisane w [21]. Tego typu rozwiązanie ma następujące zalety:

- duże zwiększenie możliwości obliczeniowych przy stosunkowo niewielkim wzroście kosztów,
- możliwość integracji PLC i systemu SCADA na jednym urządzeniu (podobnie jak w panelach operatorskich ze zintegrowanymi sterownikami PLC),
- możliwość zastosowania istniejącej infrastruktury na obiekcie w przypadku przebudowy; należy jedynie zamienić istniejący sterownik typu „hard” na jednostkę wyposażoną w odpowiedni moduł komunikacyjny,
- teoretycznie możliwość zastosowania istniejącego oprogramowania z jednostki „hard PLC”, po modyfikacji ewentualnych różnic między systemami.

Taki „sterownik PLC w komputerze PC” wykorzystuje standardowe języki programowania sterowników PLC (zgodność z normą IEC 61131-3) do tworzenia logiki sterującej takiej jak:

- **IL – Instruction List** to tekstowy język programowania składający się z serii instrukcji, z których każda znajduje się w oddzielnej linii i zawiera operator z jednym lub więcej argumentem (zależnie od funkcji),
- **LD – Ladder Diagram** jest graficznym językiem programowania, który swoją strukturą przypomina obwód elektryczny. Doskonały do łączenia POU (Program Organization Units). LD składa się z sieci cewek i styków ograniczonej przez linie prądowe. Linia z lewej strony przekazuje wartość logiczną TRUE, z tej strony zaczyna się też wykonywać linia pozioma,
- **FBD – Function Block Diagram** jest graficznym językiem programowania przypominającym sieć, której elementy to struktury reprezentujące funkcje logiczne bądź wyrażenia arytmetyczne, wywołania bloków funkcyjnych itp,
- **SFC – Sequential Function Chart** to graficzny język programowania, w którym łatwo jest ukazać chronologię wykonywania przez program różnych procesów,
- **ST – Structured Text** jest tekstowym językiem programowania, złożonym z serii instrukcji takich jak If..then lub For...do,
- **CFC – Continuous Function Chart** jest graficznym językiem programowania, który w przeciwieństwie do FBD nie działa w sieci, lecz w swobodnie położonej strukturze, co pozwala np. na stworzenie sprzężenia zwrotnego.

Autor zdecydował się na napisanie oprogramowania w języku ST, ponieważ nadawał się on najlepiej do implementacji założonej funkcjonalności. Stworzony kod jest czytelny i przejrzysty.

1.3.2 Komputer

Projekt w całości był realizowany na laptopie autora, podłączanym do sieci w laboratorium. Na komputerze uruchomiana była maszyna wirtualna. Na jednej zainstalowane było środowisko TwinCAT do programowania sterownika oraz do tworzenia i uruchamiania wizualizacji. Wizualizacje tworzone w środowisku TwinCAT można uruchomić bezpośrednio na komputerze wyposażonym w odpowiednie oprogramowanie lub na urządzeniu docelowym po podpięciu do niego monitora (o ile urządzenie docelowe posiada wyjście DVI lub odpowiedni interfejs systemowy w postaci odrębnego modułu).

1.4 Plan pracy

Realizacja projektu została podzielona na następujące etapy:

- Zapoznanie się ze sterownikami Beckhoff oraz oprogramowaniem TwinCAT,
- Zapoznanie się z dostępnymi serwonapędami Beckhoff,
- Projekt i realizacja stanowiska,
- Skonfigurowanie stanowiska do współpracy z systemem wizualizacji,
- Testowanie i uruchamianie,
- Prezentacja projektu i ewentualne korekty.

Powyższy plan pracy stanowił dla autora wyznacznik kolejnych działań. W praktyce poszczególne punkty są wymienne i wpływają na siebie wzajemnie.

2 Analiza tematu

Analiza tematu polegała przede wszystkim na zapoznaniu się z informacjami o standardzie EtherCAT oraz narzędziami programistycznymi do tworzenia oprogramowania sterownika oraz wizualizacji.

Najważniejszym punktem analizy było zapoznanie się z informacjami znajdującymi się na stronie internetowej EtherCAT Technology Group, czyli twórców i organizacji zajmującej się popularyzowaniem standardu [20]. Dodatkowo autor przeczytał wiele artykułów polsko oraz anglojęzycznych poświęconych właśnie protokołowi EtherCAT [11–19]. Kilka z nich (starszych) traktowało go jako coś bardzo przyszłościowego i obiecującego, natomiast pozostała część opisywała go już jako coś, co aktualnie bardzo szybko popularyzuje się i usprawnia procesy przemysłowe. W czasie tworzenia dokumentu opisującego przeprowadzone badania autor przeszukiwał Internet w celu uzyskania bardziej szczegółowych informacji takich jak budowa nagłówków i znaczenie oraz rozmiar poszczególnych pól. Poszukiwania te pozwoliły znaleźć sporo interesujących informacji i rozwiązań, które mogą stać się podstawą kolejnych badań w przyszłości.

Zanim autor przeczytał materiały dotyczące standardu EtherCAT poznał podstawy obsługi środowiska TwinCAT oraz jego elementów składowych pozwalające na realizację tematu, a w szczególności:

- TwinCAT System Manager - centralne narzędzie konfiguracyjne,
- TwinCAT PLC - narzędzie do tworzenia programów,
- TwinCAT NC/CNC - grupa narzędzi do sterowania osiami w różnych trybach.

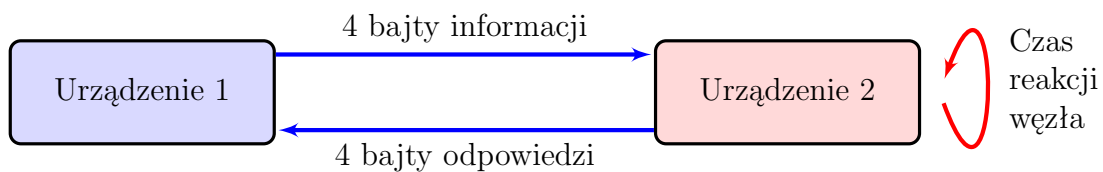
Poznanie tych podstaw pozwoliło autorowi na stworzenie potrzebnego oprogramowania oraz na konfigurację stanowisk w sposób odpowiedni do przeprowadzenia założonych badań.

2.1 EtherCAT

EtherCAT jest nowoczesnym protokołem sieciowym przeznaczonym do stosowania w aplikacjach przemysłowych, szczególnie takich, które wymagają działania całego systemu w czasie rzeczywistym. Nazwa standardu jest skrótem od hasła: „Ethernet for Control Automation Technology”. W zakresie warstwy fizycznej bazuje na Ethernetie. Dodatkowo zaimplementowano w nim mechanizmy w zakresie organizacji transmisji danych pozwalające na pominięcie głównych ograniczeń sieci Ethernet. Dzięki temu EtherCAT jest obecnie jednym z popularniejszych oraz szybciej rozwijających się protokołów komunikacyjnych w przemyśle.

2.1.1 Przetwarzanie „w locie”

W dużej części aplikacji przemysłowych dane mają mały rozmiar rzędu pojedynczych bajtów. Wykorzystując standardowy Ethernet i jego ramki stosunek danych użytecznych do narzutu protokołu jest bardzo niekorzystny. Minimalny rozmiar ramki ethernetowej wynosi 8 bajtów. Zakładając, że urządzenie okresowo przesyła 4 bajty informacji o swoich aktualnych ustawieniach, a w odpowiedzi otrzymuje również 4 bajty zestawu komend i informacji kontrolnych, przy założeniu nieskończenie krótkiego czasu odpowiedzi węzła, użyteczna przepustowość wyniesie zaledwie $\frac{4}{84} \approx 4,8\%$ [1, 2]. Jeżeli średni czas odpowiedzi będzie dłuższy, na przykład wyniesie $10 \mu s$, to użyteczna przepustowość spadnie do zaledwie 1,9%. Przebieg takiej transmisji został przedstawiony na Rysunku 5.



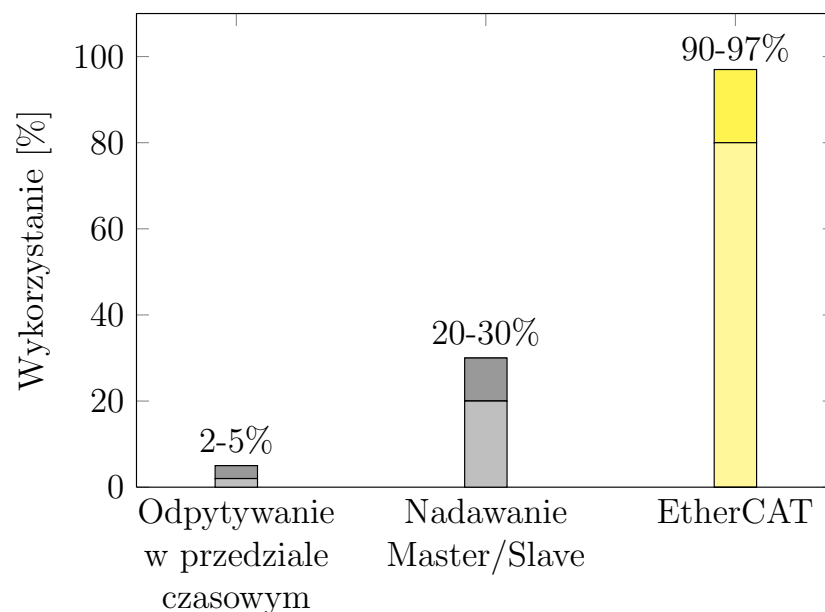
Rysunek 5: Przykład transmisji małej ilości danych (4 bajty) zwykłym Ethernetem.

W celu zapewnienia determinizmu oraz zwiększenia przepustowości stosuje się w standardach przemysłowych różne rozwiązania. Jednym z przykładów jest zastępowanie procedury dostępu do medium transmisyjnego z wykorzystaniem wielodostępu z wykrywaniem nośnej oraz detekcją kolizji (ang. *Carrier Sense Multiple Access/with Collision Detection* w skrócie *CSMA/CD*) na m.in. mechanizm odpytywania. Nie jest istotnym jaką metodę dokładnie zastosujemy dopóki ramki są rozsyłane pojedynczo z oraz do urządzeń. Działania te nie wyeliminują problemu utraty przepustowości kanału transmisyjnego i jego wykorzystania. Dlatego ten właśnie element transmisji danych z wykorzystaniem Ethernetu został potraktowany bardzo szczególnie przy projektowaniu standardu EtherCAT.

Zatem zamiast standardowej dla Ethernetu transmisji pakietowej z koniecznością odtworzenia pofragmentowanych danych zastosowano mechanizm wykorzystujący telegramy zbudowane z datagramów, które są szczegółowo opisane w podrozdziale 2.1.2. Rozwiązanie to polega na tym, że w pojedynczej ramce są zawarte informacje przeznaczone dla wielu różnych węzłów podrzędnych. Transmisja takiej ramki inicjowana jest przez węzeł nadrzędny, a następnie przechodzi ona przez kolejne węzły podrzędne sieci, które przetwarzają ją w locie. W takim podejściu twórcy standardu EtherCAT potraktowali ramkę ethernetową jak pewnego rodzaju pamięć RAM, do której zapisywane i z której odczytywane są dowolne informacje w oparciu o adresy lokalizujące poszukiwane dane w tej pamięci. Przetwarzanie to polega na tym, że w momencie odebrania ramki sprawdzane

jest, czy znajduje się w niej informacja przeznaczona dla tego właśnie węzła. Jeżeli zostanie wykryte, że tak jest to węzeł odczytuje odpowiedni fragment danych oraz uzupełnia je ewentualnie o informacje potwierdzające odbiór lub inne wymagane przez węzeł nadrzędny. Następnie ramka jest przesyłana do kolejnego w topologii węzła lub zawracana, jeśli dany węzeł jest ostatnim w sieci.

Dzięki takiemu podejściu protokół charakteryzuje się bardzo dużym wykorzystaniem kanału transmisyjnego w porównaniu do odpytywania w przedziale czasowym (ang. *Polling Timeslicing*) oraz nadawania Master/Slave (ang. *Broadcast Master/Slave*) znanych ze zwykłego Ethernetu, co pokazano na Rysunku 6 [13, 16, 17, 20].



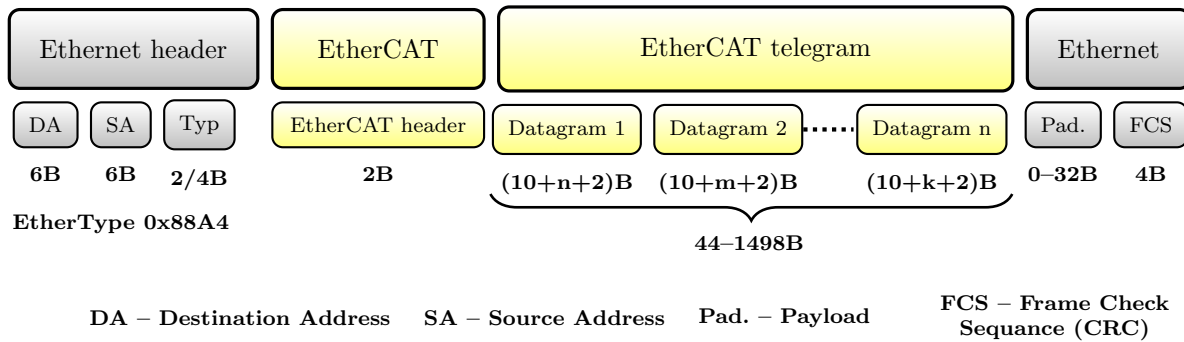
Rysunek 6: Współczynnik wykorzystania kanału transmisyjnego w Ethernetie (dwa pierwsze wykresy od lewej strony) i EtherCAT.

Wydańność tego rozwiązania jest dosyć duża, choć zależy od liczby elementów podłączonych do sieci. W praktyce czas opóźnienia nie wzrasta powyżej 1 ms i zazwyczaj jest znacznie (kilku- lub kilkunastokrotnie) krótszy. Czas synchronizacji nie przekracza 1 μs . Niniejsza praca miała na celu przebadanie i sprawdzenie, czy faktycznie protokół działa tak dobrze jak zapewniali jego twórcy.

Bardzo ważnym elementem węzła sieci jest tak zwana jednostka zarządzania pamięcią FMMU (ang. *fieldbus memory management unit*). Odpowiada ona między innymi za uniezależnienie szybkości transferu danych od wydajności i mocy obliczeniowej jednostki lokalnej CPU, kontrolę ruchu bez opóźnień oraz za odwzorowanie adresu logicznego na adres fizyczny.

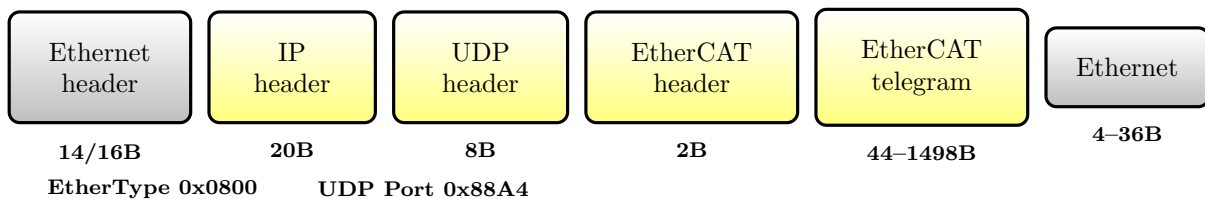
2.1.2 Telegram EtherCAT

Jak pokazano na Rysunku 7, telegram EtherCAT jest upakowany w ramce Ethernet i zawiera jeden lub więcej datagramów EtherCAT dostarczanych do urządzeń podrzędnych. Dane pomiędzy węzłami są przekazywane jako obiekty danych procesowych (ang. *process data objects*, w skrócie *PDO*). Każdy obiekt tego typu zawiera adres konkretnego węzła lub kilku węzłów typu slave.



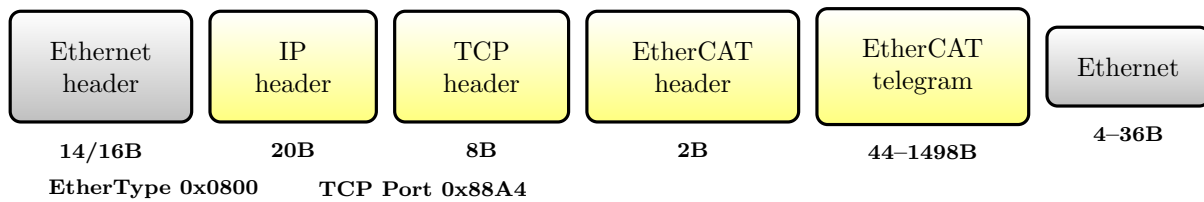
Rysunek 7: Ramka w transmisji EtherCAT i jej podział na datagramy.

Dane w systemach opartych na sieciach EtherCAT mogą być przesyłane między różnymi sieciami poprzez protokół UDP, a nie tylko w ramach jednej podsieci. Tam gdzie konieczny jest routing pakietów w oparciu o protokół IP, ramki EtherCAT są przesyłane w ramach pakietów protokołu UDP lub TCP (Rysunki 8 oraz 9). Pakiety tego typu mogą zostać uformowane i nadane z wykorzystaniem zwykłych urządzeń ethernetowych, dzięki czemu możliwe jest sterowanie aparaturą polową EtherCAT z poziomu klasycznego komputera PC wyposażonego w kartę sieciową.



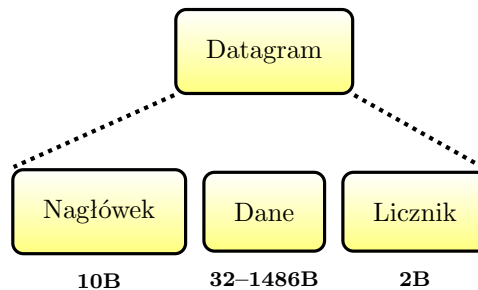
Rysunek 8: Ramka w transmisji EtherCAT z uwzględnieniem UDP i IP.

Rozwiązanie to zapewnia elastyczną komunikację pomiędzy urządzeniami pracującymi w standardzie Ethernet oraz EtherCAT, a także umożliwia wymianę danych pomiędzy urządzeniami znajdującymi się w segmentach sieci podłączonych do różnych routerów. W takim przypadku prędkość wymiany danych zależy w dużym stopniu od prędkości działania routerów, co należy uwzględnić przy ewentualnym projektowaniu systemu pracującego w rozdzielonej sieci.



Rysunek 9: Ramka w transmisji EtherCAT z uwzględnieniem TCP i IP.

Każdy datagram EtherCAT jest komendą, która zawiera zgodnie z Rysunkiem 10 nagłówki, dane i licznik roboczy. Nagłówki i dane są używane do wyspecyfikowania operacji, które muszą być wykonane przez węzeł podrzędny. Natomiast licznik roboczy jest aktualizowany przez węzeł slave informując węzeł master, że odebrał on i przetwarza komendę.



Rysunek 10: Budowa datagramu.

Nagłówek datagramu przedstawiono na Rysunku 11. Jak widać składa się on ze sporej ilości pól wymagających dokładnego objaśnienia.

Cmd – Typ rozkazu EtherCAT (ang. *EtherCAT Command Type*).

Idx – Indeks jest numerycznym identyfikatorem używanym przez węzeł nadrzędny do wykrywania zdublowanych lub zagubionych datagramów, pole to nie powinno być nigdy modyfikowane przez węzły podrzędne.

Address – Adres urządzenia zapisany w sposób zależny od trybu adresacji.

Len – Długość danych zawartych w analizowanym datagramie (rozmiar pola danych).

R – Pole zarezerwowane, powinno mieć zawsze wartość 0.

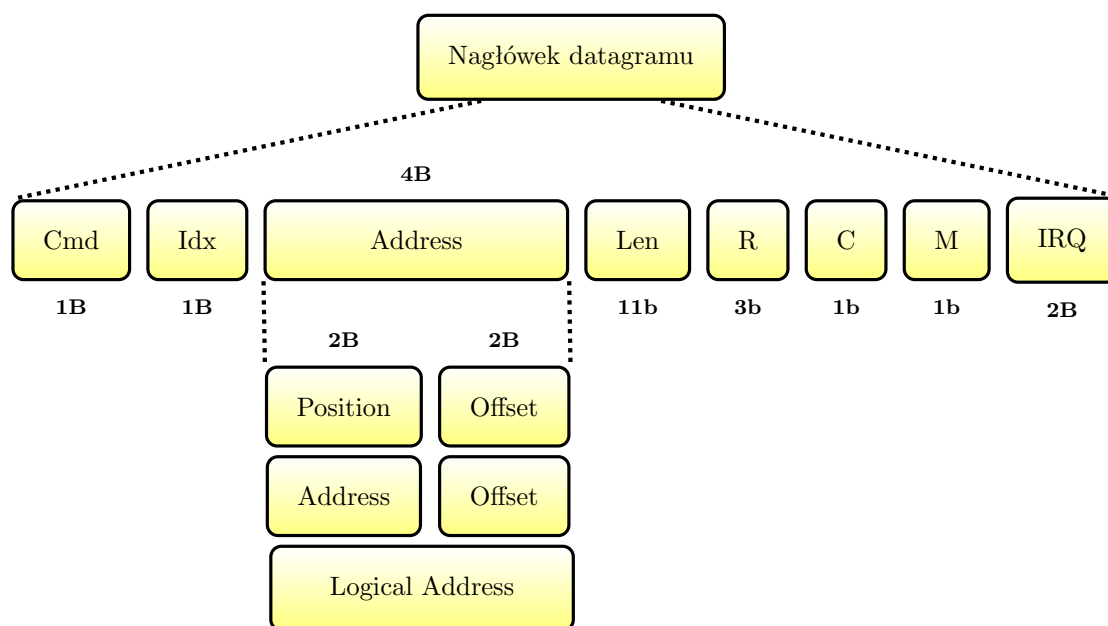
C – Pole zabezpieczające ramkę przed zapętleniem

0: oznacza, że przetwarzana ramka nie jest zapętłona,

1: oznacza, że przetwarzana ramka wykonała już jeden obieg pętli.

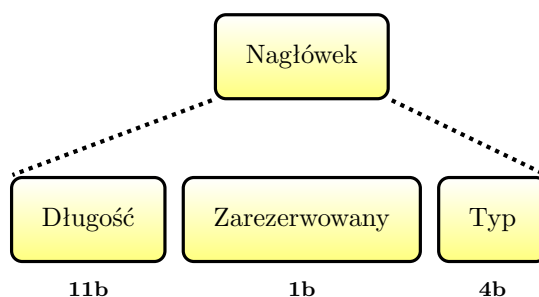
M – Pole to określa, czy w aktualnie przetwarzanym telegramie są kolejne datagramy
 0: oznacza, że przetwarzany datagram jest ostatni,
 1: oznacza, że za przetwarzanym datagramem są kolejne.

IRQ – Pole żądania zdarzeń od węzłów podrzędnych, pole jest wynikiem sumy logicznej (ang. *logical OR*) żądań wszystkich węzłów podrzędnych.



Rysunek 11: Budowa nagłówka datagramu.

Oprócz części zawierającej dane pogrupowane w datagramy, telegram EtherCAT zawiera również nagłówek, którego budowę przedstawiono na Rysunku 12. Pierwsze pole określa długość wszystkich datagramów w danym telegramie, która zależy od ilości węzłów oraz długości wiadomości. Drugie pole jest bitem zarezerwowanym, a jego wartość powinna wynosić 0. Ostatnie pole określa typ protokołu EtherCAT.



Rysunek 12: Budowa nagłówka EtherCAT.

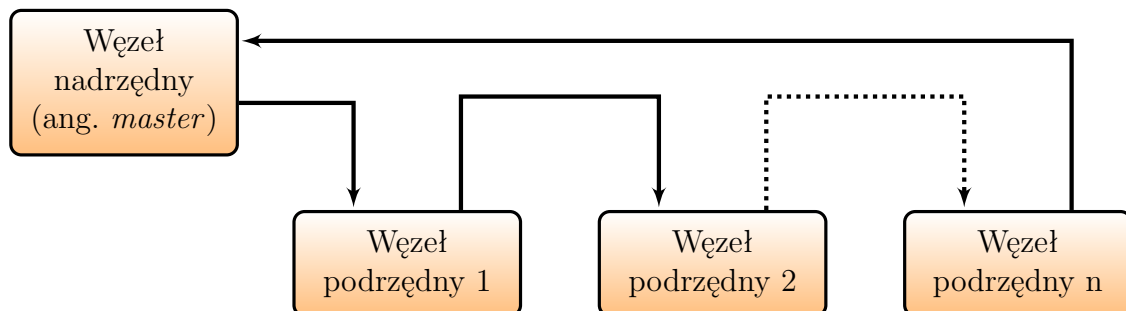
Standard przewiduje 3 dopuszczalne typy [9]:

- Typ 1: Protokół EtherCAT dla urządzeń – wymiana datagramów (ang. *EtherCAT Device Protocol, EtherCAT Datagram(s)*),
- Typ 4: EAP Wymiana danych procesowych – wymiany cykliczne (ang. *EtherCAT Automation Protocol, Process data communications*),
- Typ 5: EAP Wymiana komunikatów na żądanie – wymiany acykliczne (ang. *EtherCAT Automation Protocol, Mailbox communication*).

Typ ten definiuje typ wiadomości, co pozwala na prawidłową interpretację znajdujących się dalej danych. Zależnie od typu różni się również sposób wymiany komunikatów np. cykliczne i acykliczne.

2.1.3 Topologia

Kolejne ramki nadawane są przez urządzenie pełniące rolę kontrolera i przesyłane do najbliższego urządzenia podrzędnego. Urządzenie to ma przydzielony adres, który jednoznacznie identyfikuje datagram, dzięki czemu może odczytać przeznaczone dla siebie dane. Niezależnie od tego, czy urządzenie coś zapisało, czy też tylko odczytywało fragment ramki, przesyła ją dalej do kolejnego z urządzeń zgodnie z tym, co pokazano na Rysunku 13. Duża szybkość tej metody transmisji wynika m.in. z tego, że nie ma potrzeby dekodowania całej ramki danych. Pozwala to przekazywać ramki pomiędzy urządzeniami ze znacznie większą prędkością.



Rysunek 13: Przykładowa topologia sieci.

Można zauważyć, że opisana procedura transmisji wymaga zastosowania topologii magistrali, która w przypadku klasycznego Ethernetu jest nieoptymalna i ma wiele wad. Problem ten został rozwiązany poprzez zwielokrotnienie portów ethernetowych instalowanych w dużej części urządzeń zgodnych z EtherCAT. Powszechnie spotykane są urządzenia z dwoma lub trzema portami, a te wystarczają już do realizacji topologii gwiazdy, czy nawet zmieszanie topologii gwiazdy i magistrali w ramach jednej sieci i utworzenie struktury mocno redundantnej.

2.1.4 Warstwa fizyczna

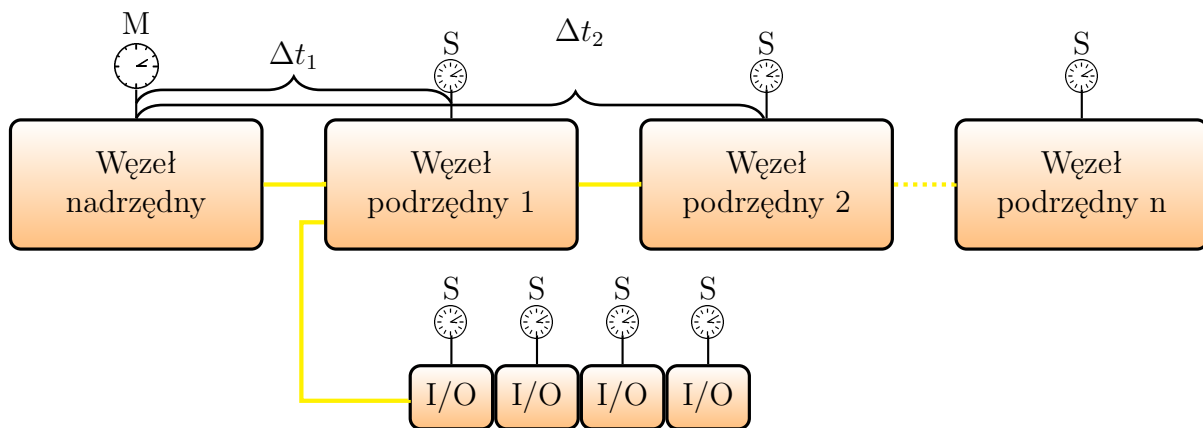
Jako medium komunikacyjne w sieciach EtherCAT można wykorzystać kable miedziane (100Base-TX), światłowody (100Base-FX) lub łącze E-bus w technologii LVDS. Te ostatnie wprowadzono ze względu na to, że transmisja w sieciach EtherCAT często jest realizowana na krótkich dystansach. E-bus sprawdza się w realizacji łączności na odległość do ok. 10 m. Kable miedziane sprawdzają się na większych odległościach poniżej 100 metrów, natomiast użyteczna długość światłowodów może dochodzić do 20 kilometrów. Warunkiem wykorzystania światłowodów jest obsługa pełnego duplexu. Wymóg ten wynika z faktu, że transmisja danych jest tak szybka, iż z reguły ramka odpowiedzi dociera do urządzenia nadrzędnego zanim całe zapytanie zostanie wysłane. Z tego powodu, aby przesył danych pomiędzy urządzeniami nie był zakłócony musi istnieć możliwość jednoczesnego przekazywania informacji w dwóch kierunkach bez spadku transferu.

W obrębie jednej sieci EtherCAT można dowolnie zmieniać medium transmisyjne zależnie od potrzeb. Na przykład wewnątrz szafy sterowniczej, gdzie występują niewielkie odległości między węzłami można zastosować z powodzeniem łącze E-bus, natomiast do połączenia szafy z modułami znajdującymi się przy maszynach wykonawczych można zastosować kabel miedziany lub światłowód w przypadku zdecydowanie większych odległości oraz prowadzenia przewodów w otoczeniu występowania zaburzeń elektromagnetycznych. Niestety operacja takiego łączenia wymaga zastosowania dodatkowych modułów, np. aby połączyć skrętkę miedzianą oraz światłowód należy zastosować moduł sprzęgający EK1501 oraz terminal przyłączy EK1521 (oba produkcji firmy Beckhoff).

2.1.5 Synchronizacja

Dobra synchronizacja elementów składowych systemu jest bardzo istotna, a szczególnie w przypadku równoległej realizacji zależnych od siebie zadań. Specjalnie opracowana dla protokołu EtherCAT technika zegara rozproszonego pozwala zsynchronizować ze sobą urządzenia z dokładnością większą niż $1\ \mu s$.

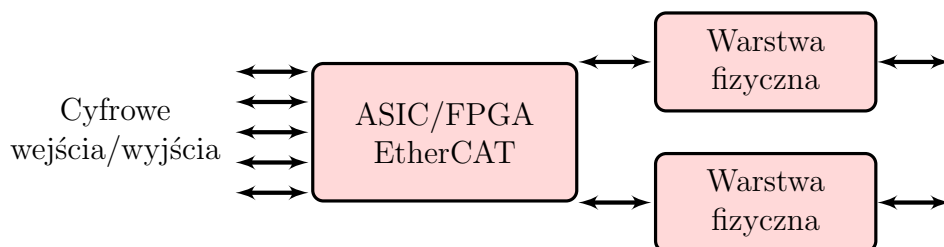
Podejście to polega na wykorzystaniu znaczników czasu zapisywanych przez każdy węzeł podrzędny. Na ich podstawie węzeł master oblicza opóźnienia propagacji sygnałów dla każdego węzła slave. Zegar w każdym węźle podrzędnym jest regulowany z wykorzystaniem wyliczonych opóźnień. Po zainicjowaniu połączenia w celu utrzymania synchronizacji zegarów muszą one przekazywać między sobą regularnie informacje, by uniknąć powstania ewentualnych różnic. Rozwiązanie to jest zgodne z protokołem precyzyjnej synchronizacji zegara dla sieciowych systemów kontrolno-pomiarowych lub inaczej protokołu czasu precyzyjnego (ang. *Precision Time Protocol*, w skrócie *PTP*) IEEE 1588 [29].



Rysunek 14: Schemat pracy zegarów rozproszonych.

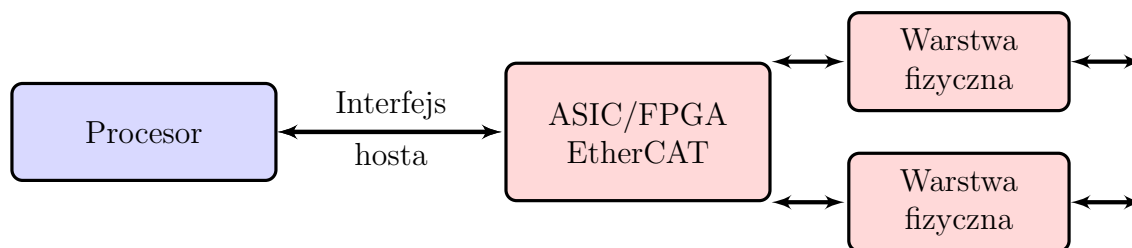
2.1.6 Realizacja węzłów EtherCAT

Wiele najprostszych i najtańszych urządzeń z interfejsem EtherCAT można zrealizować z wykorzystaniem pojedynczego układu scalonego FPGA lub ASIC. Przykładami takich prostych urządzeń z zastosowaniem tego rozwiązania są moduły wejść/wyjść cyfrowych. Tego typu węzły nie wymagają tworzenia dodatkowego oprogramowania, ponieważ cała funkcjonalność jest realizowana w pełni sprzętowo. Uogólniona i uproszczona architektura tej metody realizacji została przedstawiona na Rysunku 15.



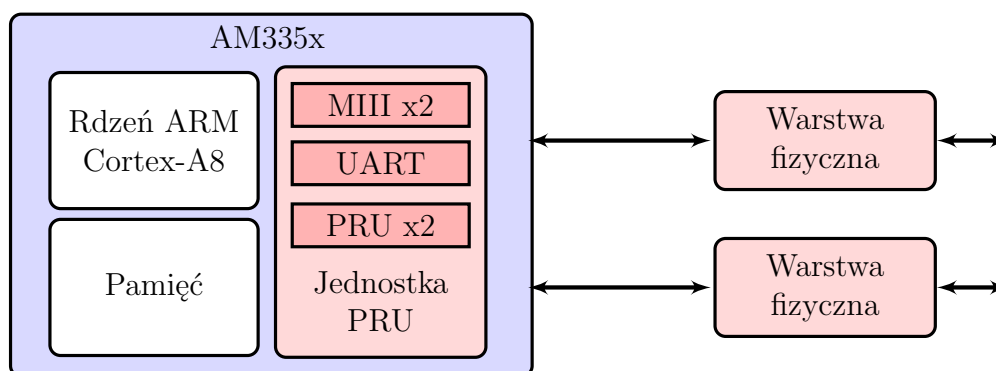
Rysunek 15: Budowa węzła EtherCAT z wykorzystaniem pojedynczego układu FPGA lub ASIC.

Jeżeli węzeł potrzebuje dodatkowej mocy obliczeniowej lub wymaga realizacji jakiegoś złożonego oprogramowania, którego nie da się zrealizować w prosty sposób sprzętowo (w układzie FPGA) do układu ASIC/FPGA EtherCAT dołączany jest zewnętrzny procesor często wyposażony w pamięć typu Flash tak jak na Rysunku 16. Procesor taki ma zapewnić obsługę przetwarzania na poziomie aplikacji. Niestety koszt tego typu architektury jest wyższy niż w prostszym przypadku pozbawionym zewnętrznej jednostki, ale konstruktor bazujący na tym rozwiązaniu ma większe pole manewru w doborze procesora odpowiedniego dla wymagań i budżetu realizowanego projektu.



Rysunek 16: Budowa węzła z wykorzystaniem układu ASIC/FPGA EtherCAT z dołączonym zewnętrznym procesorem.

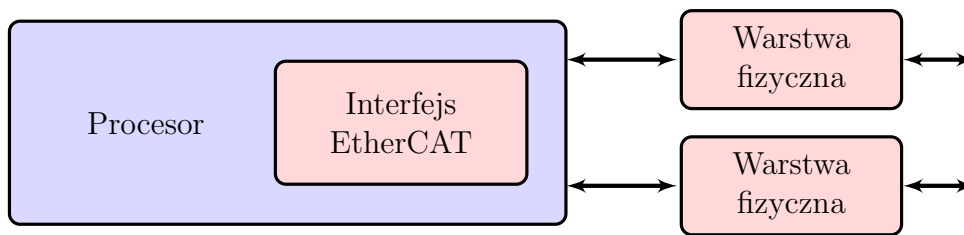
Kolejnym możliwym do wykorzystania rozwiązaniem jest zastosowanie układu FPGA z wbudowanym procesorem jak na Rysunku 17. Wspólną cechą przedstawionych dotychczas możliwych konstrukcji jest fakt, że wymagają z reguły zastosowania dwóch układów. Wynika z tego niestety, że zajmują więcej miejsca oraz zwiększają koszty urządzenia docelowego. Alternatywą pozwalającą wyeliminować oba opisane problemy jest zastosowanie elementów jednoukładowych. Ich wykorzystanie pozwala zredukować całkowity koszt konstrukcji nawet o 30%.



Rysunek 17: Budowa układu FPGA z wbudowanym procesorem.

Jednym z przykładów opisanego jednoukładowego rozwiązania są mikrokontrolery z rdzeniem ARM Cortex-A8 z rodziny Sitara AM335x produkowanymi przez amerykańską firmę Texas Instruments (Rysunek 18). Kluczowym elementem takiego układu scalonego jest programowalna jednostka czasu rzeczywistego (ang. *programmable real-time unit*, w skrócie *PRU*).

W PRU zaimplementowana została warstwa MAC standardu EtherCAT. Dzięki temu odpowiada ona bezpośrednio za przetwarzanie przepływających przez nią telegramów, dekodowanie adresów urządzeń oraz wykonywanie zapisanych w datagramie komend. W wyniku takiego rozwiązania, że wszystkie założenia oraz cała funkcjonalność jest realizowana właśnie przy użyciu opisywanego PRU, zamontowany wewnątrz procesor może być



Rysunek 18: Budowa mikrokontrolera Sitara AM335x wyposażonego w programowalną jednostkę czasu rzeczywistego.

zastosowany do realizacji bardziej zaawansowanych oraz złożonych zadań wynikających ze specyfiki konkretnego sprzętu.

2.1.7 EtherCAT Technology Group

Standard EtherCAT został opracowany w 2003 roku przez Beckhoff Automation, niemiecką firmę z branży automatyki przemysłowej. Następnie powołano organizację EtherCAT Technology Group (ETG), która zajęła się standaryzacją tego protokołu. Stowarzyszenie to obecnie zajmuje się też organizowaniem szkoleń oraz popularyzacją tego standardu.

Aktualnie w skład ETG wchodzi ponad 2480 firm (dane z dnia 1 września 2013). Najważniejszym członkiem organizacji jest oczywiście firma BECKHOFF Automation. Pozostałe duże i znane firmy wchodzące w jej skład to między innymi: ABB, Brother Industries, BMW Group, Częstochowa University of Technology, Epson, FANUC, Festo, GE Intelligent Platforms, Hitachi, Hochschule Ingolstadt, Mitsubishi, Microchip Technology, Mentor Graphics, Nikon, National Instruments, OLYMPUS, Panasonic, Rzeszów University of Technology, Red Bull Technology, Samsung Electronics, TRW Automotive, Volvo Group, Volkswagen oraz Xilinx.

Na powyższej liście można zaobserwować, że w skład organizacji wchodzi firmy z bardzo wielu branż, a nawet ośrodki naukowe. Autor pracy wybrał duże i dobrze znane sobie firmy, aby pokazać jak wiele z nich interesuje się rozwojem przemysłowych protokołów komunikacyjnych.

2.1.8 Maszyna stanów EtherCAT

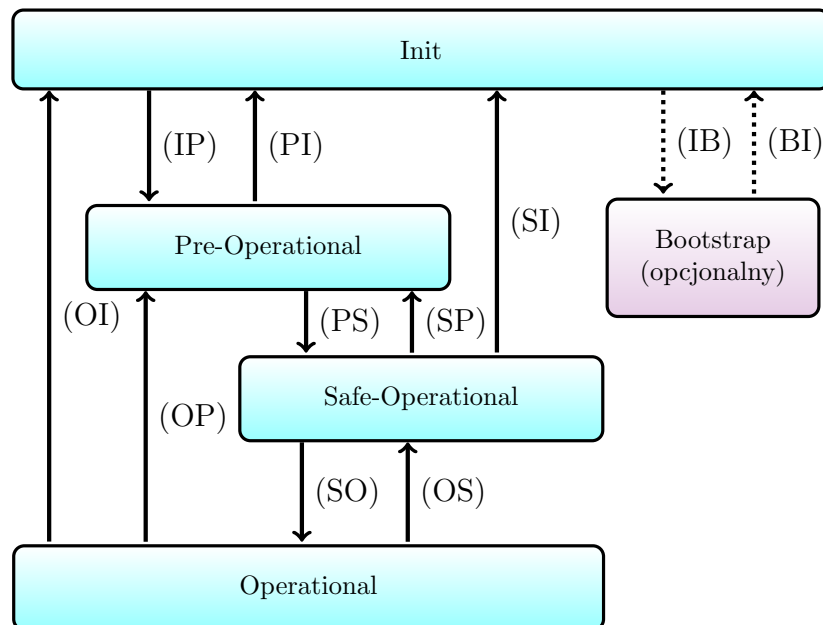
Maszyna stanów EtherCAT (ang. *EtherCAT State Machine*, w skrócie *ESM*) jest rozwiązaniem odpowiedzialnym za koordynowanie oprogramowania węzłów nadrzędnych i podrzędnych podczas ich uruchamiania oraz w czasie ich normalnej pracy [23]. Zmiana stanów jest zazwyczaj zapoczątkowana poprzez żądanie wysłane przez węzeł master.

Zmiana taka jest potwierdzana przez lokalną aplikację po wykonaniu powiązanych z tym operacji. Dobrowolne zmiany stanów dowolnego węzła również są możliwe. Proste urządzenia bez dodatkowej logiki np. w postaci mikrokontrolera mogą być skonfigurowane do wykorzystywania emulowanej maszyny stanów. Takie urządzenie po prostu przyjmuje i akceptuje dowolne zmiany stanów w sposób automatyczny.

Zgodnie z informacjami z dokumentacji producenta istnieją cztery stany, które powinny być wspierane przez węzeł podrzędny oraz jeden stan opcjonalny:

- Init,
- Pre-Operational,
- Safe-Operational,
- Operational,
- Bootstrap (opcjonalny).

Wszystkie stany oraz dopuszczalne przejścia między nimi zostały przedstawione na Rysunku 19, a serwisy z nimi związane w Tabelicy 2.



Rysunek 19: Maszyna stanów EtherCAT.

Warto zauważyć, że nie wszystkie teoretyczne przejścia są dozwolone w praktyce. Na przykład, aby przejść ze stanu Init do stanu Operational należy wykonać następującą sekwencję: Init \Rightarrow Pre-Operational \Rightarrow Safe-Operational \Rightarrow Operational. Każdy stan definiuje wymagane serwisy. Przed zmianą stanu węzeł potwierdza czy wszystkie serwisy wymagane przez żądany stan zostały odpowiednio dostarczone lub zatrzymane.

Stan/Zmiana stanu	Serwisy
INIT	<ul style="list-style-type: none"> • Brak komunikacji na warstwie aplikacji • Master uzyskał dostęp do rejestrów informacyjnych warstwy łącza danych
INIT \Rightarrow PREOP	<ul style="list-style-type: none"> • Master konfiguruje rejestry adresowe warstwy łącza danych oraz kanały SyncManager dla komunikacji asynchronicznej • Master inicjalizuje synchronizację zegarów rozproszonych • Master żąda stanu 'Pre-Operational' • Oczekiwanie na potwierdzenie statusu warstwy aplikacji
PREOP	<ul style="list-style-type: none"> • Asynchroniczna wymiana komunikatów w warstwie aplikacji • Brak wymiany danych procesowych
PREOP \Rightarrow SAFEOP	<ul style="list-style-type: none"> • Master konfiguruje parametry wykorzystując komunikację asynchroniczną • Master konfiguruje rejestry warstwy łącza danych • Master żąda stanu 'Safe-Operational' • Oczekiwanie na potwierdzenie statusu warstwy aplikacji
SAFEOP	<ul style="list-style-type: none"> • Asynchroniczna wymiana komunikatów w warstwie aplikacji • Wymiana danych procesowych, ale tylko zmienne wejściowe są przetwarzane, wyjściowe pozostają w stanie 'Safe'
SAFEOP \Rightarrow OP	<ul style="list-style-type: none"> • Master wysyła poprawne dane wyjściowe • Master żąda stanu 'Operational' • Oczekiwanie na potwierdzenie statusu warstwy aplikacji
OP	<ul style="list-style-type: none"> • Dane wejściowe oraz wyjściowe są poprawne
BOOT	<p>Opcjonalne lecz zalecane jeśli aktualizacje oprogramowania są konieczne</p> <ul style="list-style-type: none"> • Zmiany stanów tylko z/do INIT • Brak wymiany danych procesowych • Asynchroniczna wymiana komunikatów w warstwie aplikacji • Dopuszczalne specjalne konfiguracje skrzynki odbiorczej

Tablica 2: Serwisy maszyny stanów.

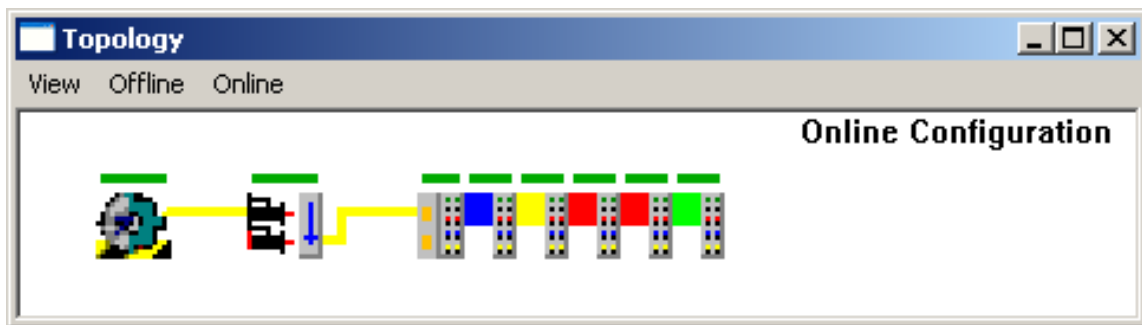
3 Stanowiska badawcze

W niniejszym rozdziale opisane zostaną stanowiska przygotowane do przeprowadzenia kolejnych badań.

3.1 Podstawowe stanowiska

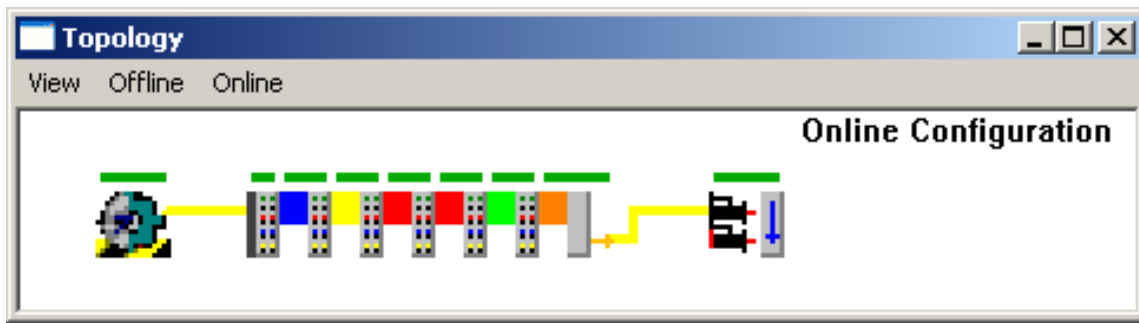
Pierwszym etapem prac mających na celu przygotowanie stanowisk badawczych było skonfigurowanie podstawowej wersji obu stanowisk wraz ze wszystkimi dostępnymi elementami [10]. Stanowiska zostały skonfigurowane i uruchomione w sposób całkowicie niezależny od siebie.

Tak stworzone konfiguracje pozwoliły uzyskać odpowiednie topologie: jak na Rysunku 20 dla stanowiska typu CP, oraz jak na Rysunku 21 dla stanowiska typu CX. W stanowisku typu CP liczba skonfigurowanych węzłów podrzędnych wynosi 7, natomiast na stanowisku typu CX wynosi ona 8.



Rysunek 20: Topologia stanowiska typu CP.

Na zrzucie ekranu dla stanowiska typu CP zauważyć można, że do węzła z uruchomionym oprogramowaniem TwinCAT (w tym przypadku sterownika PLC) za pomocą ekranowanej skrętki został podłączony napęd serwo mechanizmów, a dopiero do niego za pomocą tego samego medium transmisyjnego zdalny zestaw modułów I/O. Podgląd topologii niestety nie uwzględnia połączenia pomiędzy elementami składowymi wyspy, które jest zrealizowane z wykorzystaniem łącza E-bus. Moduły przylegają bezpośrednio do siebie, co jest zgodne ze stanem fizycznym, ale zdaniem autora połączenie takie powinno być w pewien sposób zobrazowane, ponieważ osoba nie znająca dobrze protokołu, w tym szczególnie studenci na zajęciach laboratoryjnych mogą nie być świadomi, że są to osobne węzły sieci EtherCAT.



Rysunek 21: Topologia stanowiska typu CX.

Topologia stanowiska typu CX różni się zasadniczo tylko kolejnością węzłów. Moduły wejścia/wyjścia są tutaj podłączone do jednostki centralnej protokołem EtherCAT w oparciu o łączy E-bus. Ostatnim tak podłączonym elementem jest terminal przyłączeniowy EK1100, do którego za pomocą kabla miedzianego (żółte połączenie na rysunku) podłączony jest napęd serwomechanizmów. W przypadku tej topologii niezrozumiałe jest dla autora wprowadzenie żółtego połączenia pomiędzy jednostką centralną, a pierwszym modułem, ponieważ jest to niekonsekwentne oraz sugeruje, że w tym miejscu występuje połączenie kablem miedzianym.

Głównym problemem związanym z uruchomieniem pełnych możliwości stanowisk było prawidłowe skonfigurowanie oraz uruchomienie serwomechanizmów napędzanych przez moduł AX5203, co zostanie szczegółowo opisane w podrozdziale 5.1.

Zgodnie z założeniami należało przygotować oprogramowanie sterownika PLC oraz wizualizację umożliwiającą obserwację zależności czasowych występujących w protokole EtherCAT. Autor próbował znaleźć w standardowych bibliotekach timer umożliwiający zmierzenie czasu trwania stanu wysokiego sygnału dyskretnego. Niestety wszystkie dostępne w środowisku zegary samoczynnie resetowały zmierzoną wartość czasu po zakończeniu odmierzania. Z tego powodu autor zdecydował się na własną implementację stopera. Kod stworzonego bloku funkcyjnego przedstawiono na Listingu 1. Stoper jest uruchamiany w momencie pojawienia się wartości *'TRUE'* na wejściu *start_stop* i odmierza czas aż do ponownej zmiany na wartość *'FALSE'*. Bezpośrednio przed zatrzymaniem stopera odmierzona wartość jest przepisywana do zmiennej wyjściowej, dodatkowo w czasie pracy stoper na bieżąco przepisuje zmierzoną wartość do zmiennej wyjściowej. Zdaniem autora tak stworzony stoper można określić jako stoper z pamięcią. Przykładowa definicja i zastosowanie stopera zostały przedstawione w Listingu 2.

```

1 FUNCTION_BLOCK STOPWATCH
2 VAR_INPUT
3     max_value: TIME:=t#10S;
4     start_stop: BOOL := FALSE;
5 END_VAR
6 VAR_OUTPUT
7     value: TIME := t#0s;
8 END_VAR
9 VAR
10     timer: TON;
11 END_VAR
12
13 IF timer.PT <> max_value THEN
14     timer.PT := max_value;
15 END_IF;
16
17 IF start_stop = TRUE THEN
18     timer(IN:=TRUE);
19     value :=timer.ET;
20 ELSE
21     IF timer.IN THEN
22         value :=timer.ET;
23     END_IF;
24     timer(IN:=FALSE);
25 END_IF;

```

Kod źródłowy 1: Kod źródłowy stopera.

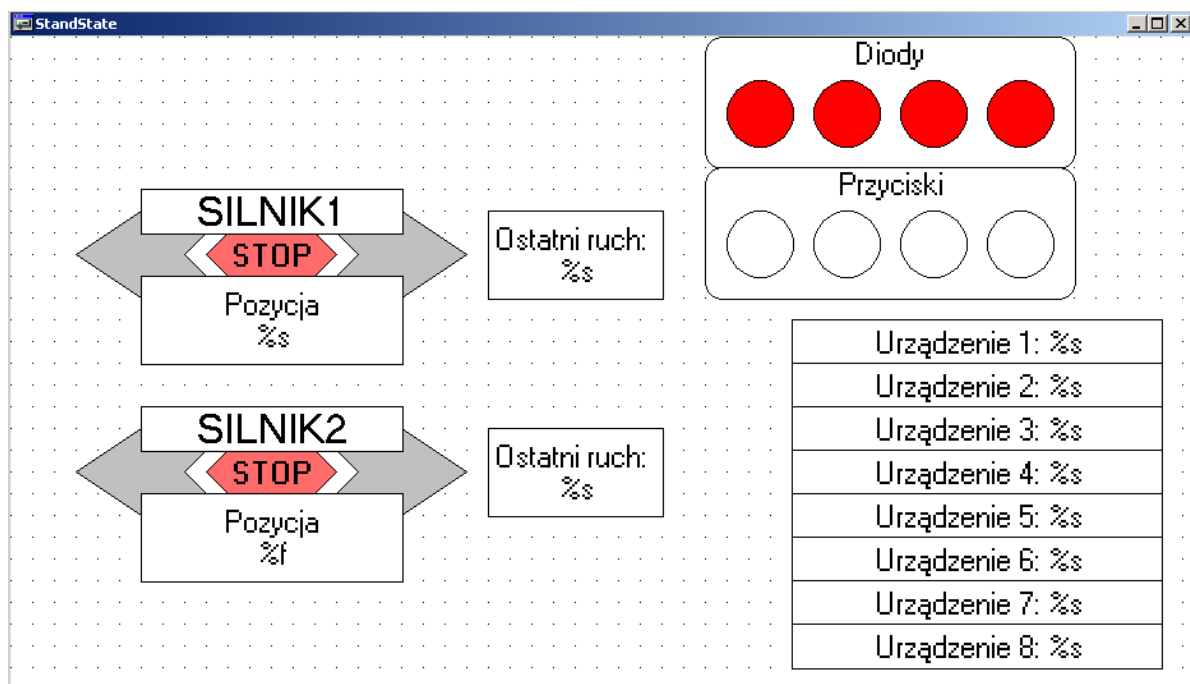
```

1 PROGRAM MAIN
2 VAR
3     stoper_device8: STOPWATCH := (max_value:=t#20s);
4     stoper_axis1: STOPWATCH := (max_value:=t#120s);
5 END_VAR
6 ...
7 stoper_device8.start_stop := state_device8 <> 8;
8 stoper_device8();
9
10 stoper_axis1.start_stop := NOT axis1_Not_moving;
11 stoper_axis1();

```

Kod źródłowy 2: Wywołania odmierzania czasu stworzonym blokiem funkcyjnym *STOPWATCH*.

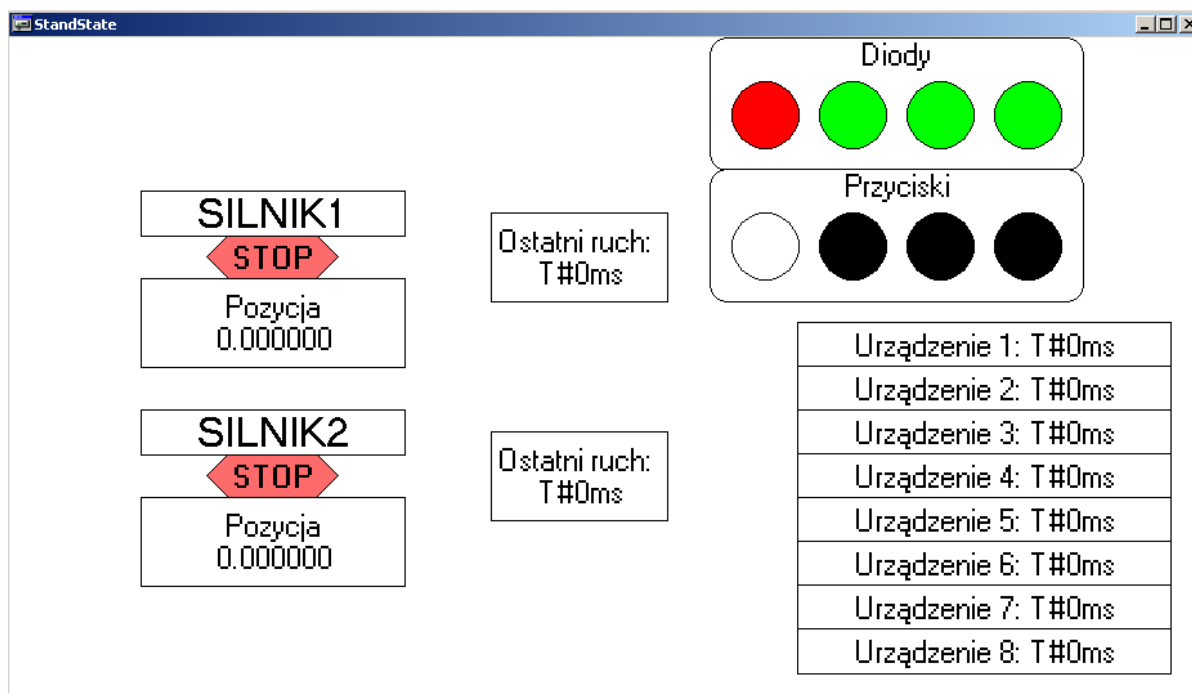
Po skonfigurowaniu stanowiska oraz po stworzeniu odpowiedniego oprogramowania ostatnim elementem do przygotowania była wizualizacja. Widok projektu przygotowanej wizualizacji został przedstawiony na Rysunku 22.



Rysunek 22: Projekt wizualizacji.

Podstawowym elementem jest przedstawienie stanu najważniejszych urządzeń wejściowych i wyjściowych. W górnej części znajdują się elementy odwzorowujące stan diod LED oraz przycisków monostabilnych. Prawa strona przedstawia stan obu silników, a dokładnie, czy są one w ruchu oraz w którym kierunku. Dodatkowo została umieszczona informacja z enkodera o aktualnej pozycji silnika. W celu przystosowania wizualizacji do obserwowania czasów mierzonych przez autora w swoich badaniach została ona uzupełniona o wartości mierzone za pomocą stworzonego stopera. Przykładowy wygląd stworzonej wizualizacji w czasie pomiarów został przedstawiony na Rysunku 23.

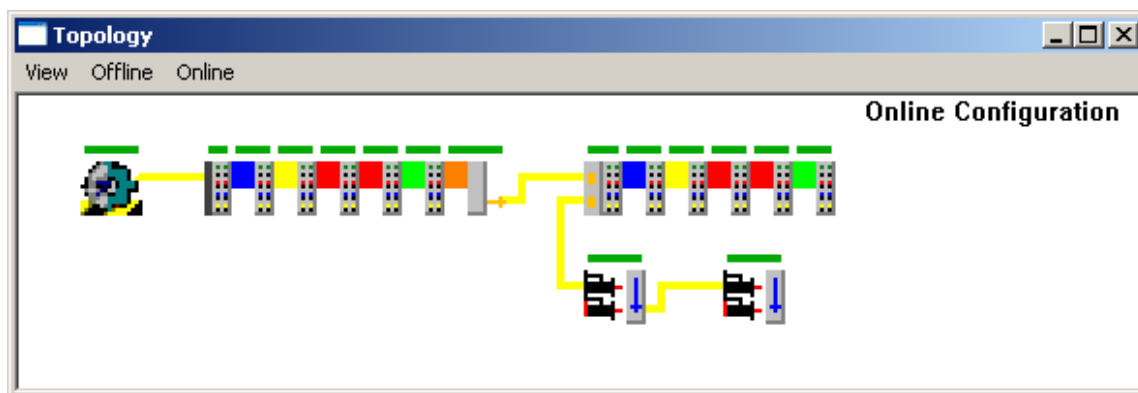
Na ekranie można zaobserwować stan początkowy stanowiska po uruchomieniu. Silniki są ustawione na pozycjach 0, stan pracy żadnego z węzłów nie był zaburzany oraz naciśnięto 3 przyciski, co poskutkowało wyłączeniem diod czerwonych i załączeniem diod zielonych.



Rysunek 23: Przykładowa uruchomiona wizualizacja.

3.2 Rozbudowane stanowisko

Stanowisko to posłuży do zbadania, jaki wpływ na wcześniejsze eksperymenty ma zwiększenie liczby węzłów. Liczba wszystkich węzłów podrzędnych wynosi 15 i jest to maksymalna ilość jaką można uzyskać na dwóch wykorzystywanych stanowiskach. Użytkowana topologia rozbudowanego stanowiska została przedstawiona na Rysunku 24. Na przedstawionym zrzucie ekranu widać, że podstawą do stworzenia rozszerzonej wersji było stanowisko CX z podłączonymi za pomocą łącza E-bus modułami wejścia/wyjścia. Do zestawu tego, za pomocą kabla miedzianego (żółte połączenia na rysunku), zostały podłączone kolejno: zdalne moduły wejścia/wyjścia, pierwszy napęd serwomechanizmów oraz na końcu drugi napęd.



Rysunek 24: Topologia stanowiska typu CX++.

Na tak przygotowanym stanowisku, w celu uniknięcia różnic powstałych przez zastosowane oprogramowanie sterownika PLC, autor zdecydował się uruchomić projekt przygotowany dla wersji podstawowej. Wybrane zostały po dwa przyciski, diody czerwone oraz diody zielone z każdego stanowiska. Celem przeprowadzenia eksperymentów związanych z opóźnieniem transmisji do wizualizacji i oprogramowania zostały przypisane silniki, których automatyczne wykrycie udostępnia środowisko do zarządzania projektem.

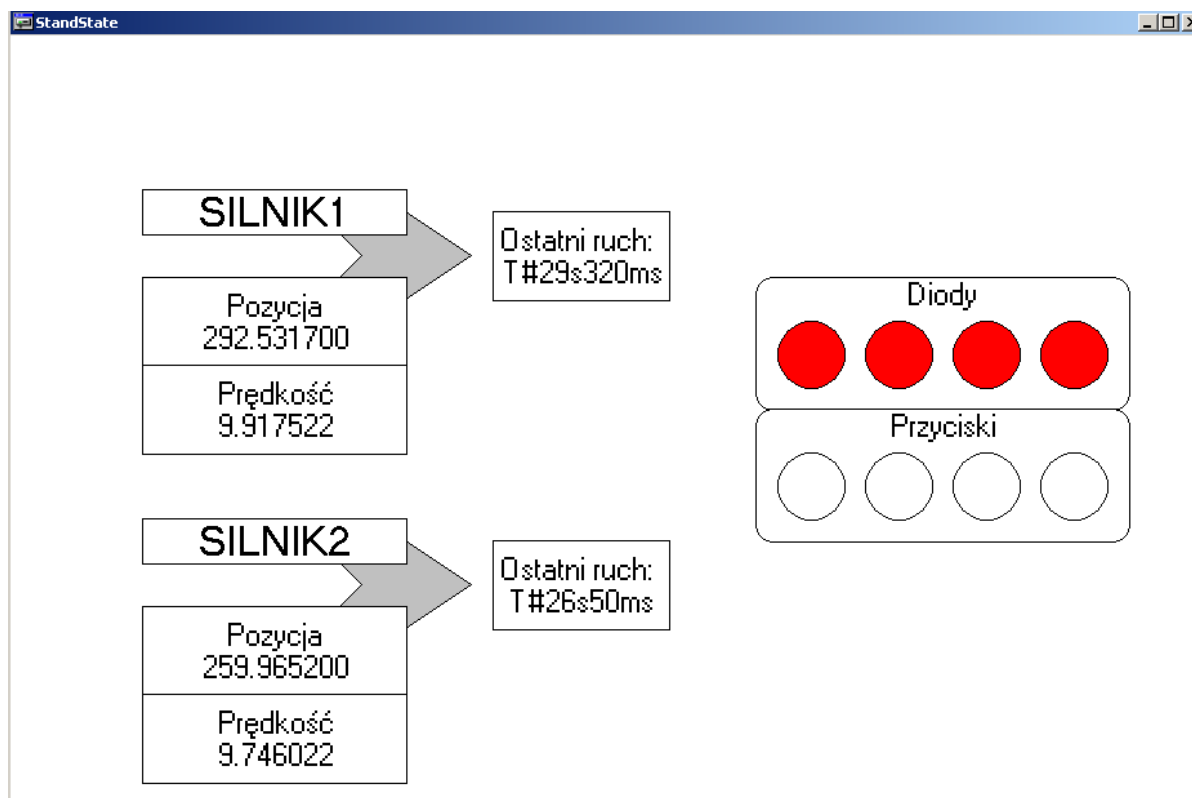
Na rozbudowanym już stanowisku autor postanowił zrealizować drugą koncepcję pomiaru zależności czasowych w protokole EtherCAT. Pomysłem było stworzenie oprogramowania umożliwiającego ruch silnika przez zadany czas. Stworzone wcześniej oprogramowanie zostało rozszerzone o warunek zezwolenia na ruch silnika. Tak jak poprzednio stoper jest wyzwalany uruchomieniem silnika, a w tym przypadku sygnał Q pochodzący z timera znajdującego się wewnątrz bloku funkcyjnego *STOPWATCH* (aktywowany w momencie odmierzenia zadanego czasu) posłużył do cofnięcia zezwolenia na ruch. Zmodyfikowany kod źródłowy został przedstawiony w Listingu 3.

```
1 PROGRAM MAIN
2 VAR
3     stoper_axis1: STOPWATCH := (max_value:=t#120s);
4     stoper_axis2: STOPWATCH := (max_value:=t#120s);
5 END_VAR
6
7 Enable_1 := NOT stoper_axis1.timer.Q;
8 Enable_2 := NOT stoper_axis2.timer.Q;
9
10 stoper_axis1.start_stop := NOT axis1_Not_moving;
11 stoper_axis1();
12 axis1_driving_time := stoper_axis1.value;
13 stoper_axis2.start_stop := NOT axis2_Not_moving;
14 stoper_axis2();
15 axis2_driving_time := stoper_axis2.value;
```

Kod źródłowy 3: Oprogramowanie odmierzające drogę przebytą w zadanym czasie.

Decydując się na zmianę oprogramowania sterownika autor postanowił zmodyfikować również wizualizację. Przede wszystkim usunięto z niej elementy związane z pomiarem czasu stabilizacji stanu urządzeń.

Stworzoną wizualizację typu drugiego przedstawiono na Rysunku 25.



Rysunek 25: Przykładowa uruchomiona wizualizacja typu drugiego.

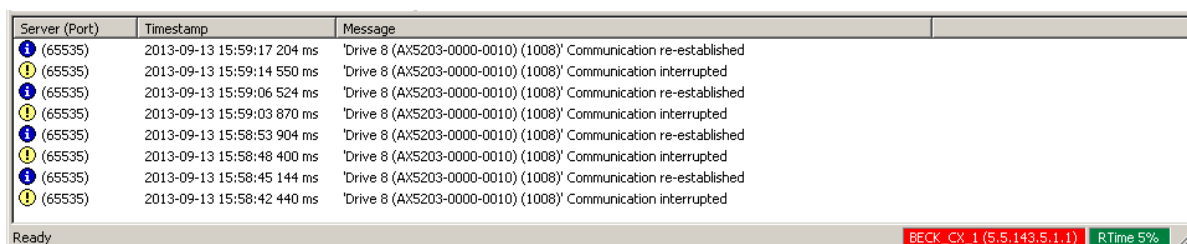
Tak przygotowany kod sterownika PLC oraz wizualizacja posłużyły do przeprowadzenia badania drogi przebytej przez silnik w zdanym czasie, ze stałą prędkością.

4 Badania

W niniejszym rozdziale opisany został przebieg przeprowadzonych badań oraz analiza ich wyników. W kolejnych podrozdziałach zostaną przedstawione różne przeprowadzone eksperymenty. Otrzymane wyniki znajdują się na dołączonej do pracy płycie CD w odpowiednich plikach, o nazwach zgodnych z podrozdziałem opisującym dane badanie i jego wyniki.

4.1 Czas stabilizacji sieci po odłączeniu i podłączeniu dowolnego urządzenia

Badanie miało na celu sprawdzenie, czy zgodnie z zapewnieniami producenta możliwe jest rozłączanie i ponowne podłączanie urządzeń do sieci „w locie” bez konieczności dodatkowej ingerencji. Do przeprowadzenia pomiarów autor odłączał przewód ethernetowy i podłączał go ponownie do gniazda. Oczywiście taka metoda może mieć wpływ na dokładność dokonywanych pomiarów, ale zdaniem autora jest on stosunkowo niewielki, ze względu na fakt, że od momentu odłączenia do wykrycia tego zajścia przez urządzenie nadzorujące pracę sieci, mija pewien czas, który w optymistycznym przypadku może być krótszy niż sam proces ingerencji autora. Aby poprawić dokładność należałoby zastosować samodzielnie stworzone urządzenie, co zostanie opisane w perspektywach rozwoju niniejszej pracy. Wszystkie zmierzone czasy oraz treści komunikatów zostały odczytane w środowisku TwinCAT System Manager, a przykładowy został przedstawiony na Rysunku 26.



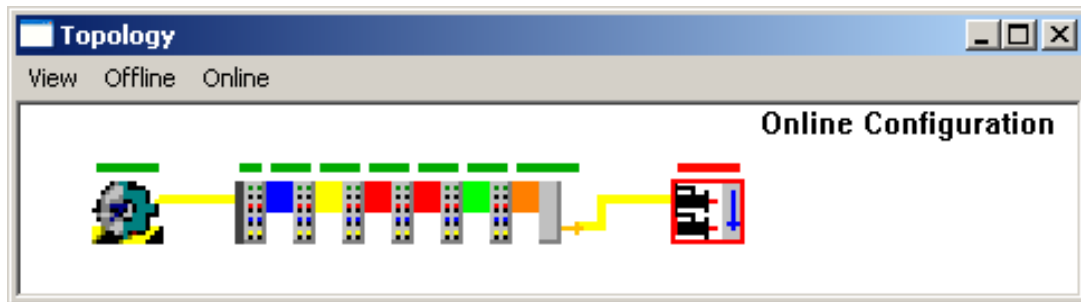
Server (Port)	Timestamp	Message
! (65535)	2013-09-13 15:59:17 204 ms	'Drive 8 (AX5203-0000-0010) (1008)' Communication re-established
! (65535)	2013-09-13 15:59:14 550 ms	'Drive 8 (AX5203-0000-0010) (1008)' Communication interrupted
! (65535)	2013-09-13 15:59:06 524 ms	'Drive 8 (AX5203-0000-0010) (1008)' Communication re-established
! (65535)	2013-09-13 15:59:03 870 ms	'Drive 8 (AX5203-0000-0010) (1008)' Communication interrupted
! (65535)	2013-09-13 15:58:53 904 ms	'Drive 8 (AX5203-0000-0010) (1008)' Communication re-established
! (65535)	2013-09-13 15:58:48 400 ms	'Drive 8 (AX5203-0000-0010) (1008)' Communication interrupted
! (65535)	2013-09-13 15:58:45 144 ms	'Drive 8 (AX5203-0000-0010) (1008)' Communication re-established
! (65535)	2013-09-13 15:58:42 440 ms	'Drive 8 (AX5203-0000-0010) (1008)' Communication interrupted

Ready BECK_CX_1 (5.5.143.5.1.1) RTime 5%

Rysunek 26: Topologia stanowiska z odłączonym jednym węzłem.

4.1.1 Pojedyncze urządzenie

Eksperyment miał na celu sprawdzenie, po upływie jakiego czasu od odłączenia i ponownego podłączenia pojedynczego węzła sieci zaczyna on znów funkcjonować poprawnie. Zaburzoną pracę sieci przedstawiono na Rysunku 27, który został wygenerowany przy użyciu środowiska TwinCAT System Manager, które udostępnia możliwość podglądu topologii sieci w czasie rzeczywistym. Do eksperymentu wybrany został węzeł końcowy w topologii.



Rysunek 27: Topologia stanowiska z odłączonym jednym węzłem.

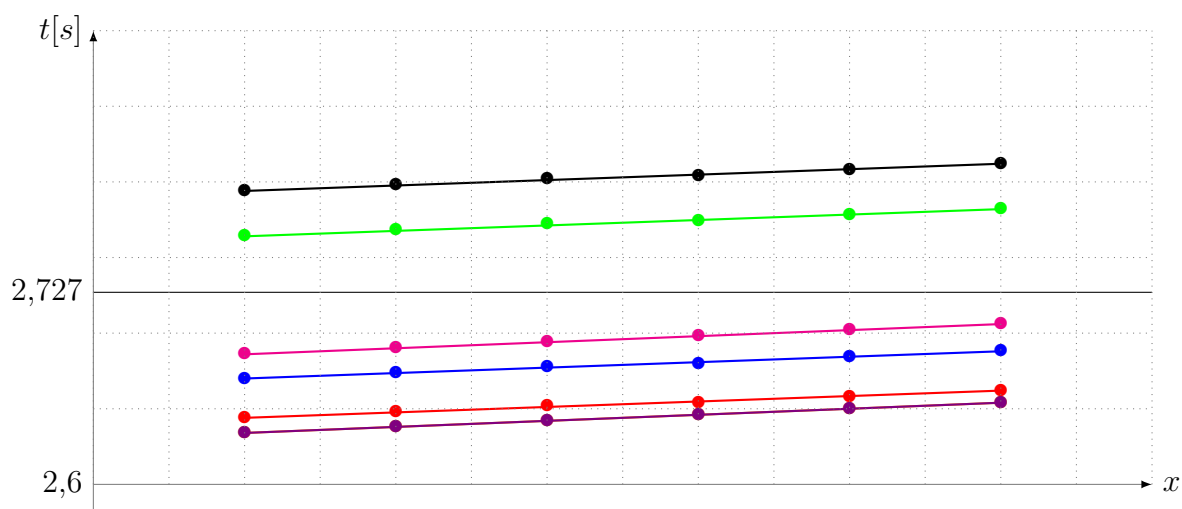
Liczba próbek	Wartość średnia	Wartość minimalna	Wartość maksymalna	Odchylenie standardowe
20	2,715s	2,644s	3,184s	0,131

Tablica 3: Wyniki przeprowadzonego badania.

Wyniki przeprowadzonego eksperymentu zostały zebrane w Tablicy 3. Na podstawie zmierzonych wartości oraz po ich analizie statystycznej uzyskano wykres przedstawiony na Rysunku 28.

Analiza wyników przeprowadzonego badania potwierdza, że istnieje możliwość odłączenia i ponownego podłączenia pojedynczego węzła sieci „w locie”. Czas potrzebny na stabilizację połączenia po jego utracie jest zdaniem autora zadowalający. Obserwując uzyskany wykres można dojść do wniosku, że różnice pomiędzy kolejnymi iteracjami eksperymentu są stosunkowo małe, o czym świadczy odchylenie standardowe, którego wartość wynosi 0,131. Zdaniem autora głównym źródłem różnic jest zastosowana metoda pomiarowa.

Na podstawie zmierzonych wartości oraz po ich analizie statystycznej uzyskano wykres przedstawiony na Rysunku 30. Różnymi kolorami są oznaczone kolejne iteracje badania, a punkty tego samego koloru oznaczają kolejne węzły zdalnej stacji wejść/wyjść.



Rysunek 30: Pomiary czasu ponownego podłączenia oraz obliczona wartość średnia.

Spoglądając na uzyskane wyniki oraz wykres można zaobserwować, że zwiększenie liczby odłączanych i podłączanych ponownie węzłów nie wpłynęło na możliwość dokonywania tej operacji w czasie pracy sieci. Ciekawa zdaniem autora jest różnica czasu pomiędzy podłączeniem się pierwszego urządzenia z zestawu, a każdym kolejnym wynoszący 2 lub 4 milisekundy. Tak więc, liczba modułów wejścia/wyjścia ma proporcjonalnie niewielki wpływ na czas potrzebny do ustabilizowania się całego zdalnego zestawu.

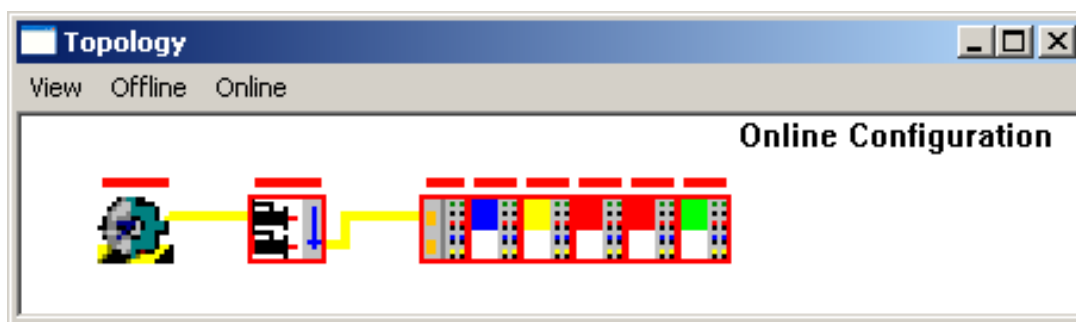
4.1.3 Wszystkie węzły sieci

Eksperyment miał na celu sprawdzenie, po upływie jakiego czasu od odłączeniu i ponownego podłączenia wszystkich węzłów sieci, wróci ona do normalnego stanu. Zaburzoną pracę sieci przedstawiono na Rysunku 31.

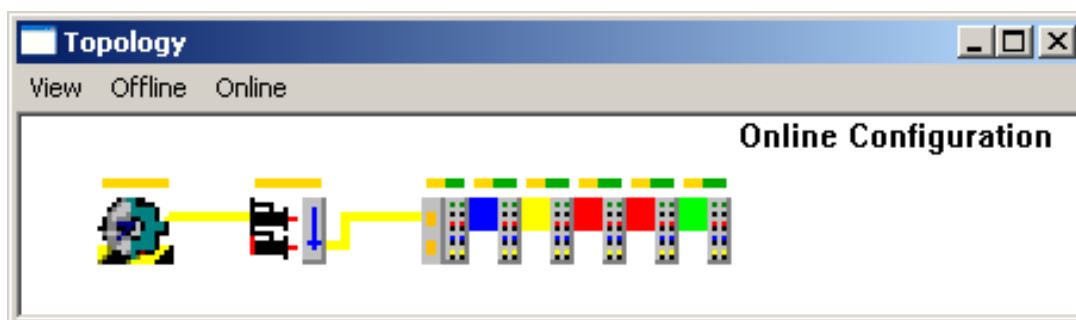
Stan sieci bezpośrednio po ponownym podłączeniu kabla sieciowego i w fazie inicjalizacji ponownego połączenia przedstawiono na Rysunku 32. Można zaobserwować, że węzły połączone w zdalną stację wejść/wyjść są już w jednej z kolejnych faz inicjalizacji, a pozostałe jeszcze jej nie rozpoczęły.

Liczba próbek	Wartość średnia	Wartość minimalna	Wartość maksymalna	Odchylenie standardowe
10	4,790s	4,149s	5,455s	0,446s

Tablica 5: Wyniki przeprowadzonego badania.



Rysunek 31: Topologia stanowiska z odłączonymi wszystkimi węzłami.



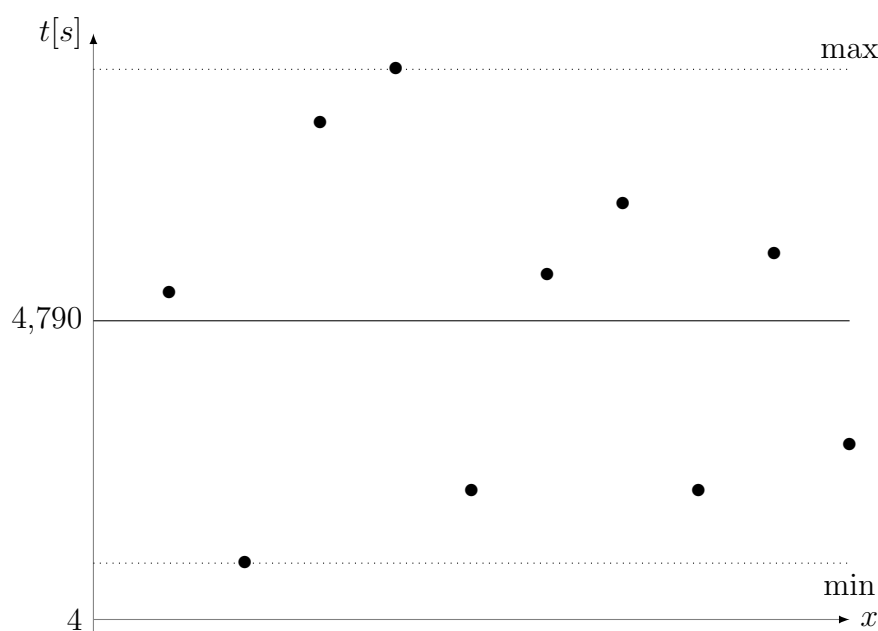
Rysunek 32: Topologia stanowiska w czasie ponownego podłączania węzłów.

Wyniki przeprowadzonego eksperymentu zostały zebrane w Tablicy 3. Na podstawie zmierzonych wartości oraz po ich analizie statystycznej uzyskano wykres przedstawiony na Rysunku 33.

Analiza wyników przeprowadzonego badania pozwala wysnuć wniosek, że możliwe jest całkowite rozłączenie i ponowne podłączenie sieci „w locie”. Czas potrzebny na stabilizację połączenia po jego utracie również w tym wypadku jest zdaniem autora zadowalający. Niestety jednak porównując uzyskane wielkości do tych z badania pojedynczego węzła sieci można zauważyć, że czas ten wzrósł dość znacząco i w przypadku jeszcze większej liczby węzłów przestanie on już być akceptowalny. Obserwując uzyskany wykres można dojść do wniosku, że różnice pomiędzy kolejnymi iteracjami eksperymentu są stosunkowo małe i wpływ na nie może mieć w większości zastosowana metoda pomiarowa.

4.1.4 Pełna stabilizacja urządzenia

W czasie badań autor zauważył, że zmierzone w pierwszych trzech eksperymentach czasy określają jedynie czas potrzebny na ustabilizowanie sieci, ale nie uwzględniają tego, że po nawiązaniu połączenia urządzenie wymaga jeszcze pewnego czasu na ustawienie się w prawidłowy stan 'Operational'. Eksperyment ten został przeprowadzony z wykorzystaniem stworzonego oprogramowania oraz wizualizacji. Autor dokonał pomiaru czasu od



Rysunek 33: Pomiary czasu ponownego podłączenia oraz obliczona wartość średnia.

momentu zmiany stanu urządzenia z OP, aż do jego powrotu do prawidłowego stanu.

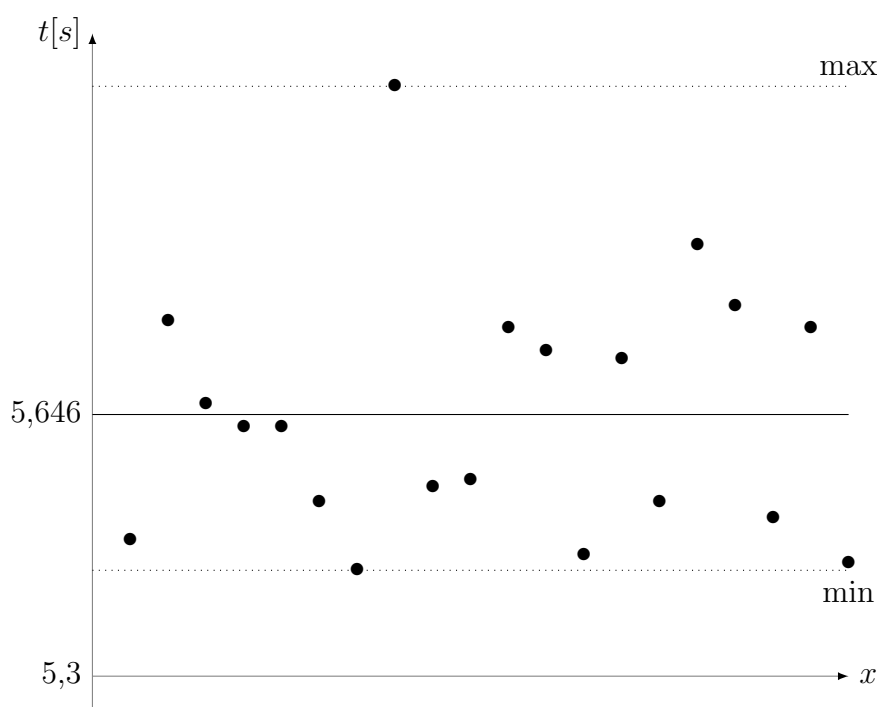
Liczba próbek	Wartość średnia	Wartość minimalna	Wartość maksymalna	Odchylenie standardowe
20	5,646s	5,440s	6,080s	0,165

Tablica 6: Wyniki przeprowadzonego badania.

Wyniki przeprowadzonego eksperymentu zostały zebrane w Tablicy 6. Na podstawie zmierzonych wartości oraz po ich analizie statystycznej uzyskano wykres przedstawiony na Rysunku 34.

Analiza wyników przeprowadzonego badania wykazuje, że czas potrzebny na przywrócenie węzła do prawidłowego działania jest dwukrotnie wyższy niż w przypadku odzyskiwania połączenia. Czas potrzebny na stabilizację połączenia po jego utracie jest zdaniem autora zadowalający, ale już nie tak dobry jak czas stabilizacji połączenia. Obserwując uzyskany wykres można dojść do wniosku, że różnice pomiędzy kolejnymi iteracjami eksperymentu są stosunkowo małe, o czym świadczy odchylenie standardowe, którego wartość wynosi 0,165. Zdaniem autora głównym źródłem różnic jest zastosowana metoda pomiarowa.

Dodatkowym wynikiem badania jest wykrycie faktu, że oprócz odłączonego węzła stan swój zmienia węzeł będący terminalem przyłączeniowym (EK1100). Fakt ten nie zostaje wykryty i zgłoszony przez środowisko oraz nie był widoczny na podglądzie topologii w czasie zaburzania. Autor postanowił przebadać i przeanalizować czas potrzebny



Rysunek 34: Pomiary czasu ponownego podłączenia oraz obliczona wartość średnia.

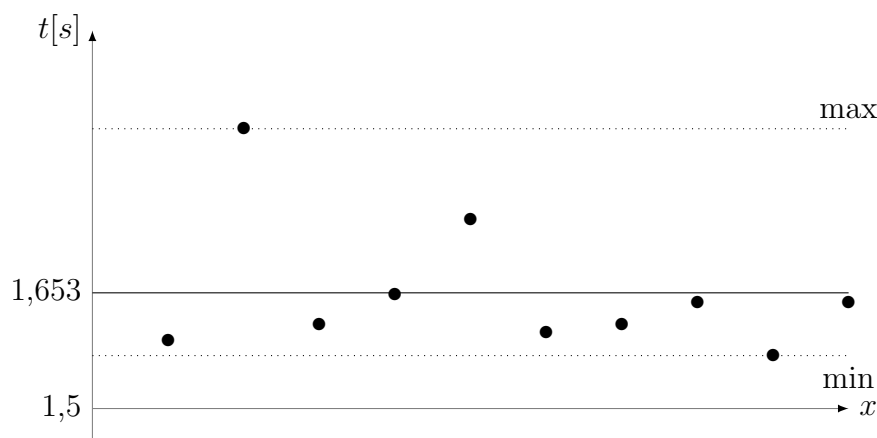
na powrót tego węzła do prawidłowego działania. Fakt ten może mieć duże znaczenie przy innych topologiach, gdzie do jednego terminala będzie podłączonych więcej węzłów podrzędnych. W takiej sytuacji autor przewiduje przerwanie komunikacji z wszystkimi węzłami podpiętymi do terminala.

Liczba próbek	Wartość średnia	Wartość minimalna	Wartość maksymalna	Odchylenie standardowe
10	1,653s	1,570s	1,870s	0,091

Tablica 7: Wyniki przeprowadzonego badania.

Wyniki przeprowadzonego eksperymentu zostały zebrane w Tablicy 7. Na podstawie zmierzonych wartości oraz po ich analizie statystycznej uzyskano wykres przedstawiony na Rysunku 35.

Wyniki przeprowadzonego eksperymentu mają taką samą charakterystykę jak we wszystkich dotychczasowych badaniach. Czas powrotu do prawidłowego stanu pracy w tym przypadku jest aż trzy razy mniejszy niż dla węzła odłączanego. Zdaniem autora wynika to z prostszego oprogramowania wewnętrznego terminala przyłączeniowego (emulowana maszyna stanów) lub węzeł zaburzony rozpoczyna swoją potencjalizację po węźle poprzedzającym.

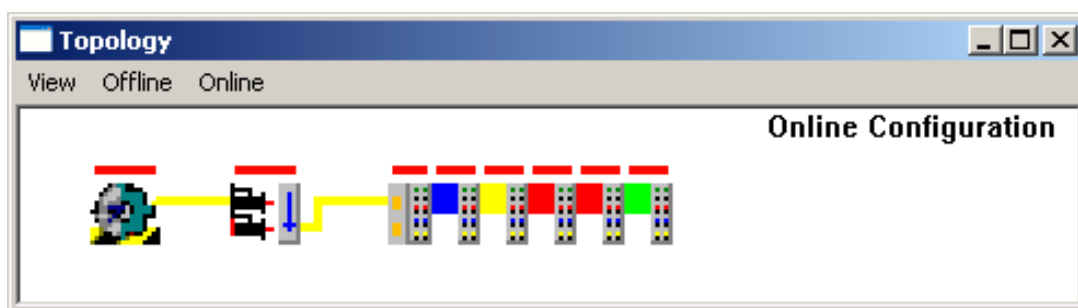


Rysunek 35: Pomiary czasu ponownego podłączenia oraz obliczona wartość średnia.

4.1.5 Problemy

W czasie dziesiątek przeprowadzonych w tym badaniu pomiarów autor doprowadził do sytuacji, w której sieć nie powróciła już samoczynnie do prawidłowego działania. Jest to sprzeczne z zapewnieniami twórców standardu i dlatego przypadek ten zostanie tu szczegółowo opisany i przeanalizowany.

Treść wiadomości odczytanej ze środowiska była następująca: *Device 2 (EtherCAT (v2.10 only)): 'INIT to PREOP' failed! Error: 'read slave count'. Communication Error '0x707 (1799)'*. Do błędu doszło w momencie zaburzenia pracy całej sieci, tj. odłączenia wszystkich węzłów podrzędnych od węzła nadrzędnego jak w badaniu 4.1.3. Stan sieci po wystąpieniu błędu przedstawiono na Rysunku 36.



Rysunek 36: Stan sieci po błędzie 0x707.

Na rysunku można zaobserwować, że urządzenia są podłączone do sieci, ale ich stan jest błędny (brak czerwony ramek wokół węzłów). Analizując stan stanowiska i sieci w środowisku zaobserwowano, że w sieci nie są przesyłane dane. Dalsza analiza wykazała, że węzeł AX-5203 jest w prawidłowym stanie inicjalizacji (0x0001), natomiast problematyczna okazała się zdalna wyspa, która ma nieprawidłowy stan 0x5C01, który według

jednej z dokumentacji producenta oznacza błąd podczas resetowania stanu [26]. Okazało się, że problem ten można prosto rozwiązać bez potrzeby restartowania całego stanowiska poprzez wymuszenie zmiany stanu węzła nadrzędnego z INIT na OP (opcja dostępna z poziomu środowiska), co skutkuje wysłaniem takiego samego żądania do wszystkich węzłów podrzędnych i sieć ponownie zaczyna funkcjonować. Wynik analizy znajduje odzwierciedlenie bezpośrednio w treści wiadomości opisującej błąd.

Podsumowując wszystkie powyższe badania przeprowadzone w tym podrozdziale autor wyciągnął dwa kluczowe wnioski:

1. odłączanie oraz ponowne podłączanie pojedynczego węzła, grupy węzłów, a nawet całej sieci jest możliwe „w locie” w pracującym systemie; należy jednak pamiętać, że operacja stabilizacji sieci oraz przywracanie urządzeń do prawidłowego stanu wymaga czasu i w przypadku nie wszystkich instalacji będzie to dopuszczalne,
2. należy pamiętać, że jeżeli w czasie przywracania prawidłowego stanu sieci jakiegokolwiek urządzenie nie przejdzie prawidłowo ze stanu inicjalizacji do stanu operacyjnego to działanie całej sieci zostaje zawieszone.

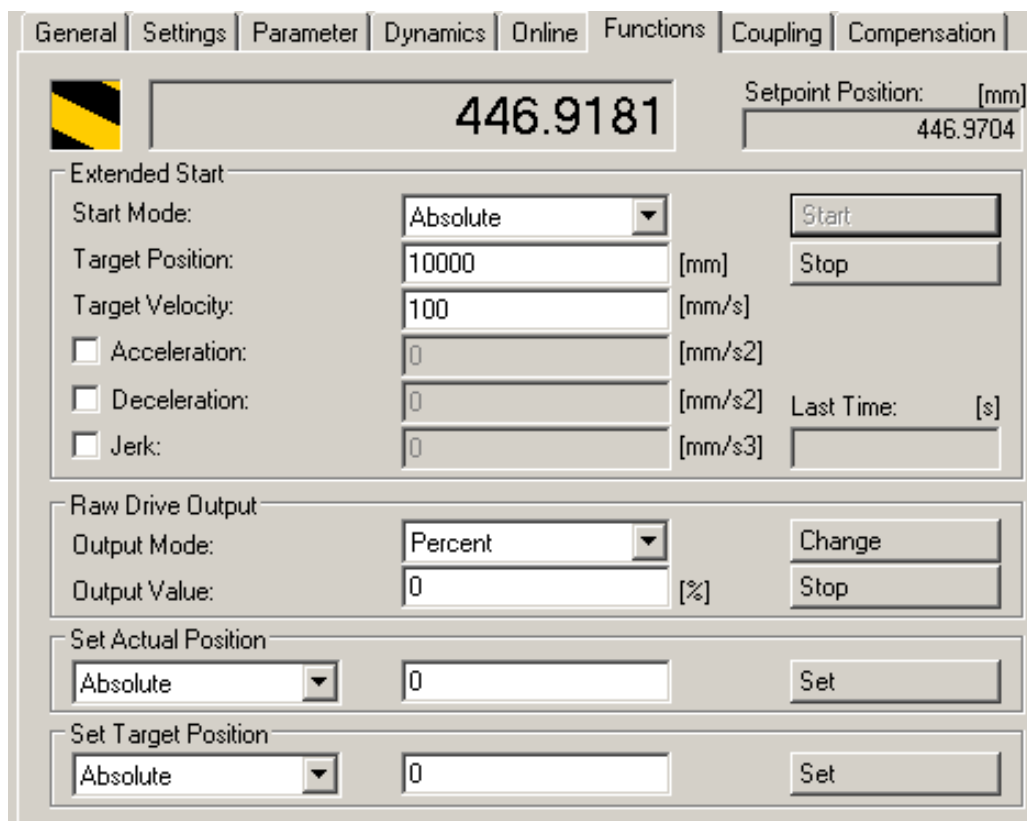
4.2 Opóźnienie transmisji

Eksperyment miał na celu sprawdzenie, czy w protokole występują opóźnienia związane z transmisją danych. Dla przebadania autor postanowił zastosować dostępne na stanowisku serwomechanizmy.

4.2.1 Badanie różnic czasu

Jako fundament wykorzystano podstawową konfigurację sterowników i wizualizację. Do badania wykorzystano po jednym serwomechanizmie z każdego stanowiska, a do ich sterowania zastosowano dostępny z poziomu TwinCAT System Managera moduł TwinCAT NC Server, którego przykładowy wygląd przedstawiono na Rysunku 37. Zastosowanie tej metody było uzasadnione tym, że obserwując zmienne związane z pracą silnika autor dostrzegł, że odpowiednie parametry są regularnie przetwarzane i wymieniane pomiędzy jednostką centralną, a napędem. Duża ilość przesyłanych danych sugeruje, że jeżeli występują jakieś opóźnienia w transmisji to eksperyment powinien je wykazać.

Autor zadawał różne funkcje podając pozycje docelowe oraz żądaną prędkość. Wykorzystując stworzone oprogramowanie oraz wizualizację mierzono czas trwania ruchu silnika. Te same zadania były równolegle uruchamiane na obu stanowiskach. Autor zawsze przeprowadzał pomiar dwukrotnie tzn. od pozycji zerowej do pozycji docelowej i z po-



Rysunek 37: Moduł pozwalający sterować silnikiem z wykorzystaniem funkcji.

wrotem żeby wykluczyć przypadkowe różnice. Wiele kombinacji odległości i prędkości poskutkowało uzyskiwaniem czasów od pojedynczych sekund do kilku minut.

Niestety badanie to nie wykazało żadnych opóźnień w transmisji, ponieważ:

- czasy tam i z powrotem w każdej próbie były równe,
- czasy odmierzone na dwóch niezależnych stanowiskach o różnej topologii oraz liczbie węzłów podrzędnych zawsze były równe.

Autor dokładał wszelkich starań, aby uzyskać rozbieżne wyniki, ale dalsze wydłużanie czasu pracy silników nie przynosiło efektów. Nawet odłączanie i ponowne podłączanie zdalnych modułów wejścia/wyjścia w czasie pracy silnika nie wpłynęło na otrzymywane wyniki. Dodatkowo autor przeprowadził to samo badanie wykorzystując stanowisko rozszerzone do 15 węzłów slave, ale również w tym przypadku osiągnięte wyniki nie wykazały występowania opóźnień transmisji.

4.2.2 Badanie różnic odległości

W porównaniu do poprzedniego eksperymentu zastosowano drugi typ stworzonego oprogramowania oraz wizualizacji. Tak samo, jak w pierwszym badaniu opóźnienia transmisji tutaj również wykorzystano sterowane z poziomu środowiska serwomechanizmy. Drugie podejście było uzasadnione przede wszystkim niepowodzeniem pierwszego eksperymentu. Pracujące ze stałą, zadaną prędkością przez określony czas silniki powinny po zakończeniu pracy stopera zatrzymać się w pewnej odległości, różnej dla obu silników.

Otrzymane wyniki potwierdziły przypuszczenia autora. W każdym pomiarze silniki zatrzymywały się w różnych punktach. Analiza wyników

5 Uruchamianie i testowanie

W rozdziale zawarto podsumowanie przebiegu prac nad projektem. Opisane zostaną tu problemy, które wystąpiły w czasie realizacji projektu. Ponadto zawarto tu opis przebiegu procesu testowania.

5.1 Napotkane problemy

Podczas tworzenia projektu napotkane i przeanalizowane zostały następujące problemy:

- Problem z automatycznym uruchamianiem stworzonego projektu PLC:

Po utworzeniu oprogramowania sterownika PLC oraz po odpowiednim skonfigurowaniu go w oprogramowaniu TwinCAT System Manager tj. linkowaniu zmienionych programu do odpowiednich fizycznych wejść i wyjść modelu oraz aktywowaniu tak przygotowanej konfiguracji, która z kolei wymusza zresetowanie systemu i nie następuje uruchomienie projektu sterownika PLC. W początkowej fazie autor przełączał się na oprogramowanie TwinCAT PLC Control gdzie logował się do sterownika i ręcznie uruchamiał stworzony przez siebie kod. Niestety to rozwiązanie na dłuższą metę okazało się niewystarczające i czasochłonne. Rozmowa przeprowadzona z promotorem uświadomiła autorowi, że w sterownikach firmy Beckhoff trzeba w specjalny sposób przygotować oprogramowanie, które ma być uruchamiane automatycznie, tzn. trzeba utworzyć projekt, który jest bootowalny. Początkowo takie podejście wydało się autorowi bardzo dziwne, ale po dłuższym zastanowieniu oraz kilku rozmowach z bardziej doświadczonymi w branży osobami okazało się, że ma ono swoje plusy. Przykładowo w przypadku tworzenia oprogramowania w fazie rozwojowej restart urządzenia pozwala przerwać całkowicie wykonywanie oprogramowania zawierającego błędy mające destrukcyjny wpływ na model lub w praktyce na obiekt przemysłowy. Po zastosowaniu nowej metody uruchamianie i testowanie tworzonego oprogramowania stało się zdecydowanie prostsze.

- Problem z wykryciem jednego z silników:

Postępując zgodnie z informacjami znalezionymi w odpowiednich materiałach szkoleniowych udało się wykryć w środowisku silnik tylko na jednej osi. Autor przyjrzał się silnikom i znalazł na nich tabliczki znamionowe z informacją, że oba silniki są identyczne. Rozwiązanie zostało znalezione dopiero po zapoznaniu się z przykładową konfiguracją stanowiska dostarczoną przez promotora. Okazało się, że drugi

silnik (o innym niż pierwszy symbolu) był skonfigurowany „na sztywno”. Autor zastosował to rozwiązanie u siebie, co poskutkowało uruchomieniem drugiego silnika. Dokładniejsza analiza silników wykazała, że na jednym z silników znajduje się druga tabliczka znamionowa z innym symbolem. Tak więc, wbrew wcześniejszym ustaleniom na stanowisku znajdują się dwa odmienne silniki różniące się od siebie podłączeniem do napędu oraz wyposażeniem wewnętrznym. Z tego właśnie powodu jeden z silników nie był wykrywany, ponieważ nie udostępnia on takiej możliwości.

- Problem ze zbyt wysokim napięciem napędu serwomechanizmów:

Pomimo skonfigurowania stanowiska zgodnie z materiałami szkoleniowymi producenta [10, 25] napędy serwomechanizmów AX5203 zgłaszały błąd zbyt wysokiego napięcia. Rozwiązanie znajdowało się w jednym z wymienionych wcześniej materiałów szkoleniowych, a mianowicie konfigurację parametrów napędu serwomechanizmów należało dodać do specjalnej 'Listy startowej' modułu. Po zastosowaniu tej opcji oraz ponownym uruchomieniu stanowiska, napęd zwrócił informację, że obie osie są gotowe do działania.

- Problem z poruszeniem silników:

Problem objawiał się tym, że pomimo wykluczenia dwóch poprzednich błędów silniki nie ruszały z miejsca. Silniki zgłaszały gotowość do pracy, ale mimo to ręczne wymuszenie ruchu silnika w przód lub tył nie przynosiło efektu. Co więcej w środowisku pojawiała się informacja, że silnik pracuje oraz porusza się w żądanym kierunku. Niestety informacje ze środowiska nie zgadzały się ze stanem fizycznym i silnik stał w miejscu.

- Utrata komunikacji z jednym ze sterowników:

Przy pewnej modyfikacji konfiguracji stanowiska typu CX związanej z próbą uruchomienia silników została utracona komunikacja z urządzeniem. Precyzując, przestało ono odpowiadać na poprzednim adresie. Sytuacja była o tyle dziwna, że system na urządzeniu działał całkowicie normalnie. Po podpięciu zewnętrznego monitora okazało się, że w systemie operacyjnym karty sieciowe są skonfigurowane prawidłowo oraz udaje się nawiązać połączenie i uzyskać żądany adres (konfiguracja adresów jest statyczna). Próby wykorzystania programu ping nie dały początkowo żadnego efektu, ponieważ urządzenie nie odpowiadało. W licznych próbach i pomysłach udało się ustalić, że urządzenie podczas uruchamiania (dokładnie podczas uruchamiania systemu operacyjnego) odpowiada na kilka zapytań (od 4 do 6 w kilku próbach), podobnie w momencie zamykania systemu. To odkrycie za-

sugerowało autorowi, że coś blokuje, przekonfigurowuje lub wywłaszcza urządzenia. Pierwszy został przeanalizowany autostart systemu Windows, lecz okazał się on pusty. Pojawił się pomysł przywrócenia urządzenia do ustawień fabrycznych, lecz nie udało się odnaleźć takiej możliwości. Problem udało się rozwiązać uruchamiając w systemie operacyjnym standardowy menadżer plików (w tym przypadku explorer), odnajdując stworzone pliki konfiguracji TwinCAT i usuwając je. (*\HardDisk\TwinCAT\Boot*)

Wszystkie problemy zostały rozwiązane i w ostatecznej wersji oprogramowania nie wpływają one w negatywny sposób na pracę modelu.

6 Wnioski

Protokół EtherCAT jest rozwiązaniem zdecydowanie bardzo nowoczesnym, zaawansowanym oraz innowacyjnym. Jest on stosunkowo bardzo młody w porównaniu z innymi znanymi autorowi protokołami komunikacyjnymi stosowanymi powszechnie w przemyśle takimi jak:

- Modbus – wprowadzony na rynek przez firmę Modicon w 1980 roku [1],
- Genius/N-80 – Sieć Genius to rozwiązanie firmy GE FANUC wprowadzone na rynek w 1985r., znane również pod nazwą N-80 - firmy ALSTOM [1],
- Profibus (**P**rocess **F**ield **B**us) – wprowadzony przez Niemiecki Departament Edukacji i Badań (niem. Bundesministerium für Bildung und Forschung, w skrócie BMBF) w roku 1994 [1],
- CAN (Controller Area Network) – wprowadzony przez Robert Bosch GmbH w roku 1994 [1],
- EGD (Ethernet Global Data) – wprowadzony przez firmy GE Fanuc Automation oraz GE Drive Systems w roku 1998.

Fakt, że protokół jest tak młody wpływa bardzo pozytywnie na liczbę materiałów dostępnych w Internecie. Niestety jest też drobna wada, a mianowicie informacje są często niekompletne i trzeba przeszukiwać Internet oraz dokumentację producenta, aby uzyskać pożądane szczegóły. Po zapoznaniu się z kilkoma stronami oraz artykułami można uzyskać kompletny zakres wiedzy o tym protokole.

Standard ten umożliwia wykorzystanie wzrastających mocy obliczeniowych sterowników PLC, przy zachowaniu narzuconych rygorów czasowych. Duży wpływ na tak szybki rozwój ma fakt, że technologia jest otwarta oraz położono duży nacisk na współpracę wszystkich zainteresowanych rozwojem stron. Olbrzymim plusem jest fakt wykorzystania standardowej struktury Ethernetu, co upraszcza proces integracji w obrębie jednego systemu obu standardów.

Badany protokół jest bardzo rozbudowany i skomplikowany. Jego zrozumienie wymaga czasu oraz zapoznania się z dużą ilością dokumentacji. Mnogość możliwości i konfiguracji będąca niewątpliwą zaletą tego standardu może na początku przerażać, ale jest to uzasadnione faktem tak rozbudowanej i wszechstronnej funkcjonalności. Autor sam był początkowo zaskoczony tak wysokim stopniem zaawansowania protokołu przemysłowego, w porównaniu przykładowo do poznanego dobrze wcześniej protokołu Modbus.

Po zakończeniu projektu posłuży on jako podstawa do stworzenia i przeprowadzenia ćwiczeń laboratoryjnych prowadzonych w ramach działalności dydaktycznej prowadzo-

nej przez pracowników Zespołu Przemysłowych Zastosowań Informatyki. Można z wykorzystaniem dołączonych konfiguracji, projektów oraz dokumentacji przygotować proste laboratorium z podstaw programowania sterowników firmy Beckhoff, bardziej zaawansowane programowanie ze sterowaniem silnikami lub najbardziej zaawansowane laboratorium związane z działaniem protokołu EtherCAT.

Temat bardzo dobrze nadaje się do pogłębiania i prowadzenia dalszych badań. Autor dopuszcza taką możliwość w ramach prac badawczych w czasie studiów III stopnia. Zarówno sterowniki firmy Beckhoff oparte o koncepcję „soft PLC” jak i badany protokół EtherCAT są na tyle rozbudowane i rozwojowe, że zawsze znajdzie się jakiś aspekt do przeanalizowania od strony teoretycznej i eksperymentalnej. W momencie powstawania niniejszej pracy trzy bardzo interesujące kwestie wydają się autorowi ciekawą postawą do przeprowadzenia badań.

Pierwszym eksperymentem zaplanowany przez autora jest przetestowanie elementów sieci EtherCAT pochodzących od innych producentów. Jak już zostało to opisane w rozdziale zawierającym analizę tematu węzły EtherCAT mogą być realizowane na wiele sposobów, zależnie od wymagań funkcjonalnych. Autor bardzo chętnie podjąłby badania mające na celu zaprojektowanie własnego układu cyfrowych wejść/wyjść i porównanie takiego rozwiązania z dostępnymi na rynku, aby wyznaczyć stosunek jakości do kosztów. W dalszym etapie zdecydowanie warto by przyjrzeć się bardziej elastycznym i rozbudowanym rozwiązaniom pozwalającym na realizację bardziej zaawansowanych węzłów. Interesująca wydaje się możliwość konstruowania zupełnie od podstaw węzłów o dowolnej funkcjonalności. Dzięki temu można by spróbować zbudować węzeł eksperymentalny mający za zadanie tylko monitorowanie przepływających przez niego ramek i ich ewentualne gromadzenie. Być może taka analiza krążących w sieci ramek pozwoliłaby wykryć pewne interesujące właściwości lub zachowania charakterystyczne dla omawianego protokołu.

Zdaniem autora najbardziej interesujące pod względem badawczym są jednostki wyposażone w procesor dedykowany do realizacji bardziej złożonych zadań niezależnie od działania protokołu. Ciekawym rozwiązaniem jest oferowany przez firmę Texas Instruments mikrokontroler z rdzeniem ARM Cortex-A8 z rodziny Sitara AM335x. Być może rozwiązania firm wchodzących w skład ETG okażą się lepsze lub gorsze pod względem szybkości w porównaniu do pierwszych twórców protokołu.

Drugim ciekawym rozwiązaniem i pomysłem, na który autor natrafił w sieci w czasie analizy tematu, rozwiązywania problemów oraz pisania niniejszej pracy jest EtherLab [27, 28]. Jest to technologia łącząca sprzęt i oprogramowanie w celach testowych oraz do sterowania procesów przemysłowych. Jest to niejako technika zbudowana z dobrze zna-

nych i niezawodnych elementów. EtherLab pracuje jako działający w czasie rzeczywistym moduł jądra otwartego systemu Linux, który komunikuje się z urządzeniami peryferyjnymi poprzez protokół EtherCAT. Rozwiązanie jest całkowicie darmowe i otwarte, co bez wątpienia jest jego olbrzymią zaletą. Można zdecydować się na pobranie sobie wszystkich komponentów i ich samodzielne uruchomienie lub zakup gotowego preinstalowanego zestawu startowego bezpośrednio od twórców. Oprogramowanie całego zestawu może zostać wygenerowane przy użyciu Simulinka/RTW lub napisane ręcznie w języku C. Następnie tak przygotowane oprogramowanie jest uruchamiane w środowisku kontrolującym proces (jądro Linuksa oraz moduł czasu rzeczywistego) komunikującym się z „obiektem przemysłowym” poprzez EtherCAT. Dodatkowo można rozszerzyć możliwości całego zestawu poprzez Ethernet TCP/IP dołączając interfejs użytkownika (ang. *Frontend*) w wersji dla Linuksa lub Windowsa albo jeden z innych dodatkowych serwisów. Przykładowe serwisy to:

- Raportowanie poprzez SMS,
- Zdalne usługi: Internet, ISDN, DSL,
- Usługi sieciowe: Web, DHCP, Drukowanie,
- Logowanie danych (ang. *data logging*).

Jeżeli autor będzie miał taką możliwość to na pewno chętnie przyjrzy się tej koncepcji ze względu na swoją sympatię do systemu Linux oraz wszystkich rozwiązań go wykorzystujących. Ciekawe wydają się badania wydajności takiego rozwiązania oraz porównanie ich z drogimi rozwiązaniami komercyjnymi.

Trzecim pomysłem, który powstał na etapie realizacji pracy dyplomowej oraz zainspirowanym po części realizacją przez kolegów z roku projektu z przedmiotu Projektowanie Przemysłowych Systemów Komputerowych, mającego na celu umożliwienie podłączenia do sterownika Beckhoff bazy danych. Studenci zaproponowali i zrealizowali rozwiązanie bazujące na RS-232. Zdaniem autora protokół EtherCAT nadawałby się do tego celu bardzo dobrze głównie ze względu na prędkość działania, ale również ze względu na możliwość bezpośredniej modyfikacji danych zapisywanych do bazy przez dowolny węzeł sieci. Pomysł nie jest w swoich podstawach badawczy, a bardziej praktyczny. Autor dostrzega jednak możliwość przygotowania stanowiska eksperymentalnego, na którym można by przeprowadzić badania wydajności takiego rozwiązania.

Czwartym pomysłem jest przebadanie protokołu z wykorzystaniem stworzonego na naszym wydziale i opisanego w literaturze analizatora sieci czasu rzeczywistego opartych na Ethernecie [22]. Urządzenie pozwoliłoby na zewnętrzną analizę protokołu EtherCAT, a nie wewnątrz działającego systemu, jak to miało miejsce w przypadku badań opisanych

niniejszym dokumentem. Możliwe byłoby porównanie opóźnień wprowadzanych przez węzły typu EtherCAT z tymi wprowadzonymi przez omawiany analizator.

Następnym pomysłem do ewentualnego przebadania w przyszłości znalezionym w sieci jest możliwość uruchomienia kontrolera urządzenia podrzędnego EtherCAT na układzie FPGA firmy Xilinx lub Altera [23, 24]. Rozwiązanie tego typu można swobodnie przetestować i zaimplementować dzięki dostarczonym przez firmę Beckhoff zestawom ewaluacyjnym wyposażonym w wymienione układy. Przykładowe dwa zestawy to EL9830 wyposażony w układ Altery (EP3C25) oraz EL9840 z Xilinxem (XC3S1200).

Teoretycznie dzięki wykorzystaniu w protokole EtherCAT standardu Ethernet możliwe jest sterowanie urządzeniami z poziomu zwykłego komputera klasy PC z wykorzystaniem modelu TCP/IP. Ciekawym wydaje się, na ile skomplikowane okaże się zbudowanie ramki do celów typowo testowych. Autor ma świadomość, że rozwiązanie takie nie nadaje się raczej do profesjonalnego stosowania w przemyśle, ale wydaje się być interesującą alternatywą do testowania działania urządzeń wykonawczych bez konieczności zaprzęgnięcia sterownika PLC. Dodatkowym interesującym aspektem wykorzystania takiego rozwiązania wydaje się lepsze poznanie i zrozumienie zasad działania protokołu. Autor w czasie realizacji jednego z projektów semestralnych prowadzonego w ramach przedmiotu Projektowanie Przemysłowych Systemów Komputerowych i swojej działalności w Kole Naukowym Przemysłowych Zastosowań Informatyki „Industrum” implementował samodzielnie oprogramowanie do komunikacji w sieci ELAN. Działanie to pozwoliło zdecydowanie lepiej zrozumieć zasady działania tego dość prostego, ale rozbudowanego interfejsu.

Tak, jak już zostało napisane w rozdziale opisującym badania do zwiększenia precyzji badań mówiących o czasie ponownego podłączenia odłączonego i podłączonego węzła sieci należałoby zaprojektować i wykonać urządzenie umożliwiające sterowanie długością przerwy w ciągłości linii transmisyjnej. Urządzenie takie można by oprzeć o układ mikroprocesorowy do kontroli oraz tranzystory do przerywania ciągłości magistrali. Pozwoliło by to w bardzo precyzyjny sposób określić czas ponownego podłączenia z pominięciem błędu pomiarowego wprowadzanego przez człowieka. Takie urządzenie pozwoliłoby również przebadać inne możliwe scenariusze. Przykładowo można by spróbować wygenerować przerwę trwającą równowartość czasu potrzebnego na transmisję pojedynczego znaku (ewentualnie kilku znaków). Pozwoliłoby to zaobserwować zachowanie protokołu w momencie wystąpienia błędów transmisji. Ewentualnie wywołując odpowiednio dużo błędów można sprawdzić stabilność pracy protokołu w sytuacji problemów transmisyjnych. Dzięki stworzeniu omówionego urządzenia dałoby się również przebadać inne protokoły komunikacyjne.

7 Bibliografia

Literatura, która została wykorzystana przez autora w czasie powstawania projektu, którą opisuje niniejsza dokumentacja.

- [1] Andrzej Kwiecień: „*Analiza przepływu informacji w komputerowych sieciach przemysłowych.*”, ZN Pol. Śl. s. Studia Informaitca z. 22, Gliwice 2002.
- [2] Piotr Gaj „*Zastosowanie protokołu TCP/IP do transmisji informacji dla potrzeb przemysłowych systemów kontrolno-nadzorczych.*”, Gliwice, 2004.
- [3] Zakład Systemów Zasilania: „*Rdzeń modułowego system czasu rzeczywistego do profesjonalnych aplikacji hybrydowych systemów zasilania i systemów automatycznego nadzoru.*”, Warszawa, grudzień 2007.
- [4] Jerzy Kasprzyk: „*Programowanie sterowników przemysłowych*”, Wydawnictwa Naukowo-Techniczne WNT, Warszawa, 2007.
- [5] „*Programowalne sterowniki PLC w systemach sterowania przemysłowego*”, Politechnika Radomska, Radom, 2001.
- [6] Andrzej Maczyński: „*Sterowniki programowalne PLC. Budowa systemu i podstawy programowania*”, Astor, Kraków, 2001.
- [7] Zbigniew Seta: „*Wprowadzenie do zagadnień sterowania. Wykorzystanie programowalnych sterowników logicznych PLC.*”, MIKOM Wydawnictwo, Warszawa, 2002.
- [8] Janusz Kwaśniewski: „*Programowalne sterowniki przemysłowe w systemach sterowania*”, Wyd. AGH, Kraków, 1999.
- [9] Materiały szkoleniowe: EtherCAT Technology Group: „*EtherCAT for Factory Networking. EtherCAT Automation Protocol (EAP).*”, lipiec 2010.
- [10] Materiały szkoleniowe Beckhoff: „*Podstawy obsługi programu TwinCAT System Manager Część 1.*”, Warszawa, 2009.
- [11] Andrzej Gawryluk: „*EtherCAT – to nie takie trudne. Ethernet jako sieć real-time*”, Elektronika Praktyczna, 2/2010.
- [12] Maneesh Soni: „*EtherCAT w praktyce. Zastosowanie mikrokontrolera Sitara do implementacji EtherCAT.*”, Elektronika Praktyczna, 7/2012.

- [13] Michał Gosk: „*Szybkość, niezawodność, doskonała synchronizacja. EtherCAT - system przyszłości.*”, Magazyn Sensor, 1/2013, kwiecień.
- [14] Monika Jaworowska: „*Determinizm czasowy transmisji w Ethernecie przemysłowym.*”, Portal branżowy dla Automatyków – AutomatykaB2B, [online], 8 listopada 2012, [dostęp 28 sierpnia 2013]. Dostępny w Internecie:
<http://automatykab2b.pl/tematmiesiaca/5061-determinizm-czasowy-transmisji-w-ethernecie-przemyslowym>
- [15] Monika Jaworowska: „*Sieci przemysłowe w standardzie EtherCat.*”, Portal branżowy dla Automatyków – AutomatykaB2B, [online], 20 czerwca 2012, [dostęp 26 sierpnia 2013]. Dostępny w Internecie:
<http://elektronikab2b.pl/technika/16954-sieci-przemyslowe-w-standardzie-ethercat>
- [16] Beckhoff Automation: „*EtherCAT – przyszłość sieci przemysłowych.*”, Portal branżowy dla Automatyków – AutomatykaB2B, [online], 28 czerwca 2011, [dostęp 22 sierpnia 2013]. Dostępny w Internecie:
<http://automatykab2b.pl/prezentacja-artykul/3898-ethercat—przyszlosc-sieci-przemyslowych>
- [17] Beckhoff Automation: „*EtherCAT i światłowody – szybko i skutecznie.*”, Portal branżowy dla Automatyków – AutomatykaB2B, [online], 2 listopada 2011, [dostęp 22 sierpnia 2013]. Dostępny w Internecie:
<http://automatykab2b.pl/prezentacja-artykul/4159-ethercat-i-swiatlowody—szybko-i-skutecznie>
- [18] Beckhoff Automation: „*Nowa, szybka sieć EtherCAT.*”, Portal branżowy dla Automatyków – AutomatykaB2B, [online], 2 listopada 2011, [dostęp 22 sierpnia 2013]. Dostępny w Internecie:
<http://automatykab2b.pl/prezentacja-artykul/4159-ethercat-i-swiatlowody—szybko-i-skutecznie>
- [19] Real Time Automation: „*EtherCAT Open Real-Time Ethernet Network.*”, Portal Real Time Automation, [online], 2009, [dostęp 2 września 2013]. Dostępny w Internecie:
<http://www.rtaautomation.com/ethercat/>
- [20] EtherCAT Technology Group: „*EtherCAT – the Ethernet Fieldbus.*”, Oficjalna strona internetowa EtherCAT Technology Group, [online], [dostęp 30 sierpnia 2013].

Dostępny w Internecie:

<http://www.ethercat.org/en/technology.html>

- [21] Krzysztof Oprzędkiewicz: „*Realizacja predyktora Smitha na platformie sprzętowo-programowej „soft PLC” bazującej na komputerze klasy PC*”, Automatyka / Akademia Górniczo-Hutnicza im. Stanisława Staszica w Krakowie, 2006, t. 10, z. 2, s. 177–186, ISSN 1429-3447.
- [22] Rafał Cupek, Piotr Piękoś, Marcin Poczobut, Adam Ziebinski: „*FPGA based „Intelligent Tap” device for real-time Ethernet network monitoring*”, Computer Networks, Communications in Computer and Information Science, Springer-Verlag Berlin Heidelberg, 2010, t. 79, s. 58-66, ISBN 978-3-642-13860-7.
- [23] Dokumentacja producenta: „*ET1815 / ET1817 EtherCAT Slave Controller IP Core for Xilinx FPGAs IP Core Release 2.02a*”, Wersja 2.2.1, 1 wrzesień 2008.
- [24] Dokumentacja producenta: „*ET1810 / ET1812 EtherCAT Slave Controller IP Core for Altera FPGAs IP Core Release 2.4.0*”, Wersja 1.0 15 marca 2011.
- [25] Dokumentacja producenta: „*AX5000 – Motion control for high dynamic positioning*”, 29 października 2007.
- [26] Dokumentacja producenta: „*AX5805 TwinSAFE drive option card for the AX5000 servo drive*”, 11 marca 2013
- [27] Ingenieurgesellschaft IgH: „*EtherLab Technology Description.*”, Esse, luty 2012.
- [28] EtherCAT Technology Group: „*EtherCAT Master for Linux as part of EtherLab®.*”, Oficjalna strona internetowa EtherCAT Technology Group, [online], [dostęp 10 września 2013]. Dostępny w Internecie:
<http://www.ethercat.org/en/products/263FCDB7639F4DDFB3587C18F46BA289.htm>
- [29] IEEE 1588-2002: IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems.

8 Spis rysunków, tablic i kodów źródłowych

8.1 Spis rysunków

Rysunek 1: Schemat działania systemu czasu rzeczywistego.	4
Rysunek 2: Funkcja zysku dla systemów czasu rzeczywistego.	5
Rysunek 3: Schemat stanowiska typu CP.	11
Rysunek 4: Schemat stanowiska typu CX.	11
Rysunek 5: Przykład transmisji małej ilości danych (4 bajty) zwykłym Ethernetem.	16
Rysunek 6: Współczynnik wykorzystania kanału transmisyjnego w Etherecie (dwa pierwsze wykresy od lewej strony) i EtherCAT.	17
Rysunek 7: Ramka w transmisji EtherCAT i jej podział na datagramy.	18
Rysunek 8: Ramka w transmisji EtherCAT z uwzględnieniem UDP i IP.	18
Rysunek 9: Ramka w transmisji EtherCAT z uwzględnieniem TCP i IP.	19
Rysunek 10: Budowa datagramu.	19
Rysunek 11: Budowa nagłówka datagramu.	20
Rysunek 12: Budowa nagłówka EtherCAT.	20
Rysunek 13: Przykładowa topologia sieci.	21
Rysunek 14: Schemat pracy zegarów rozproszonych.	23
Rysunek 15: Budowa węzła EtherCAT z wykorzystaniem pojedynczego układu FPGA lub ASIC.	23
Rysunek 16: Budowa węzła z wykorzystaniem układu ASIC/FPGA EtherCAT z dołączonym zewnętrznym procesorem.	24
Rysunek 17: Budowa układu FPGA z wbudowanym procesorem.	24
Rysunek 18: Budowa mikrokontrolera Sitara AM335x wyposażonego w programowalną jednostkę czasu rzeczywistego.	25
Rysunek 19: Maszyna stanów EtherCAT.	26
Rysunek 20: Topologia stanowiska typu CP.	28
Rysunek 21: Topologia stanowiska typu CX.	29
Rysunek 22: Projekt wizualizacji.	31
Rysunek 23: Przykładowa uruchomiona wizualizacja.	32
Rysunek 24: Topologia stanowiska typu CX++.	32
Rysunek 25: Przykładowa uruchomiona wizualizacja typu drugiego.	34
Rysunek 26: Topologia stanowiska z odłączonym jednym węzłem.	35
Rysunek 27: Topologia stanowiska z odłączonym jednym węzłem.	36

Rysunek 28: Pomiary czasu ponownego podłączenia oraz obliczona wartość średnia.	37
Rysunek 29: Topologia stanowiska z odłączoną wyspą.	37
Rysunek 30: Pomiary czasu ponownego podłączenia oraz obliczona wartość średnia.	38
Rysunek 31: Topologia stanowiska z odłączonymi wszystkimi węzłami.	39
Rysunek 32: Topologia stanowiska w czasie ponownego podłączania węzłów.	39
Rysunek 33: Pomiary czasu ponownego podłączenia oraz obliczona wartość średnia.	40
Rysunek 34: Pomiary czasu ponownego podłączenia oraz obliczona wartość średnia.	41
Rysunek 35: Pomiary czasu ponownego podłączenia oraz obliczona wartość średnia.	42
Rysunek 36: Stan sieci po błędzie 0x707.	42
Rysunek 37: Moduł pozwalający sterować silnikiem z wykorzystaniem funkcji.	44

8.2 Spis tablic

Tablica 1: Dostępne stanowiska laboratoryjne.	10
Tablica 2: Serwisy maszyny stanów.	27
Tablica 3: Wyniki przeprowadzonego badania.	36
Tablica 4: Wyniki przeprowadzonego badania.	37
Tablica 5: Wyniki przeprowadzonego badania.	38
Tablica 6: Wyniki przeprowadzonego badania.	40
Tablica 7: Wyniki przeprowadzonego badania.	41

8.3 Spis kodów źródłowych

Kod źródłowy 1: Kod źródłowy stopera.	30
Kod źródłowy 2: Wywołania odmierzania czasu stworzonym blokiem funkcyjnym <i>STOPWATCH</i>	30
Kod źródłowy 3: Oprogramowanie odmierzające drogę przebytą w zadanym czasie.	33

9 Załączniki

- Oświadczenie o autorstwie,
- Płyta CD, na której znajdują się:
 - Kody oprogramowania wewnętrznego oraz wizualizacji stworzone w TwinCAT PLC Control,
 - Pliki projektów konfiguracji stanowisk stworzone w TwinCAT System Manager,
 - Pliki zawierające wyniki przeprowadzonych badań,
 - L^AT_EX-owe, TikZ-owe oraz PGF-owe pliki źródłowe pracy magisterskiej.