



**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: Damian Karbowski

Kierujący pracą: dr inż. Jacek Stój

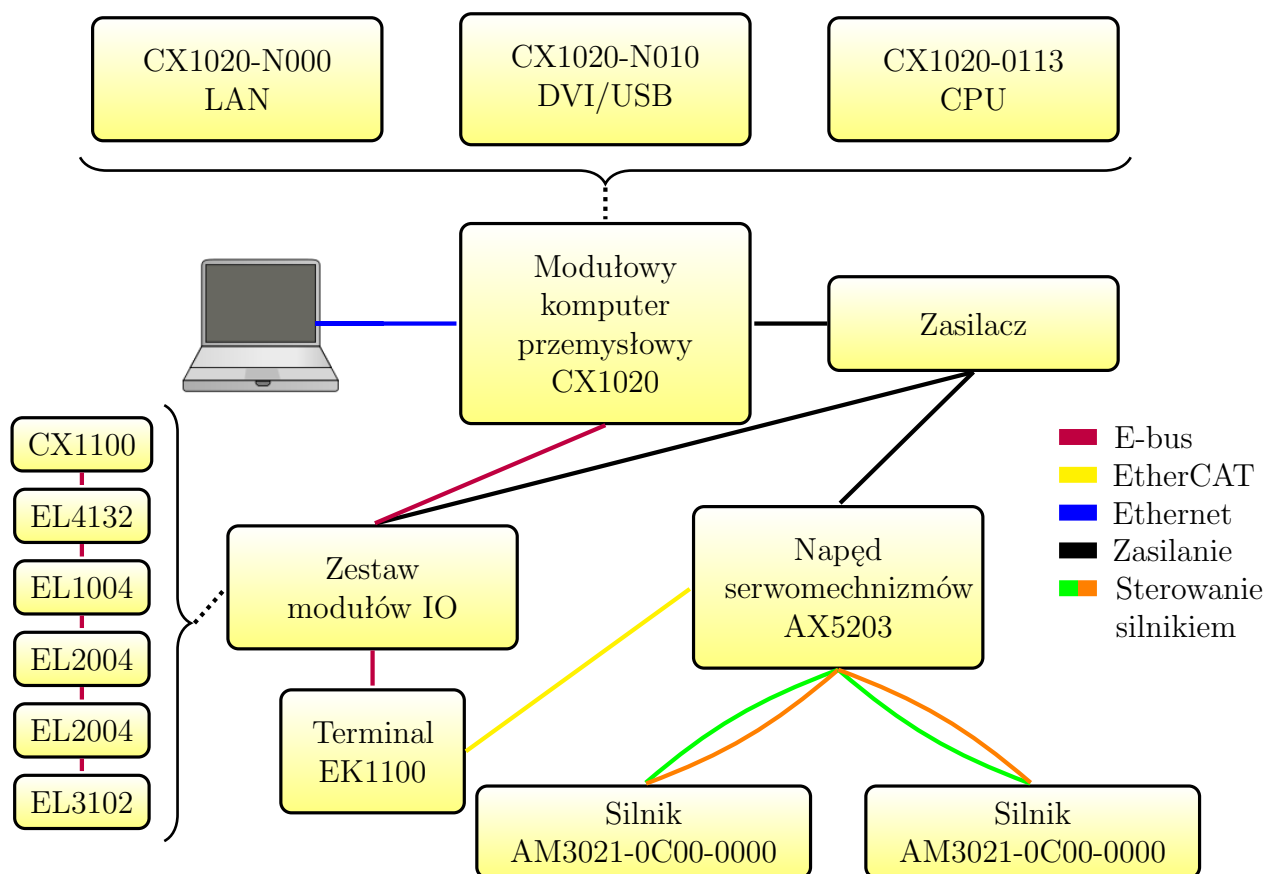
Gliwice, 1 września 2013

# Spis treści

<b>1</b>	<b>Wstęp</b>	<b>3</b>
1.1	Stanowisko laboratoryjne . . . . .	3
1.1.1	Sterownik PLC . . . . .	3
1.1.2	Komputer . . . . .	6
1.2	Analiza tematu . . . . .	7
1.3	EtherCAT . . . . .	7
1.3.1	Determinizm, przepustowość . . . . .	8
1.3.2	Przetwarzanie „w locie” . . . . .	8
1.3.3	Transmisja i synchronizacja w sieciach EtherCAT . . . . .	8
1.3.4	EtherCAT Technology Group . . . . .	8
1.3.5	Telegram EtherCAT . . . . .	9
1.3.6	. . . . .	9
1.3.7	. . . . .	9
1.3.8	. . . . .	9
1.3.9	. . . . .	9
1.3.10	. . . . .	9
1.4	Założenia . . . . .	9
1.5	Plan pracy . . . . .	11
<b>2</b>	<b>Oprogramowanie sterownika</b>	<b>12</b>
2.1	Specyfikacja zewnętrzna . . . . .	12
2.2	Specyfikacja wewnętrzna . . . . .	12
<b>3</b>	<b>Wizualizacja HMI</b>	<b>13</b>
3.1	Specyfikacja zewnętrzna . . . . .	13
3.2	Specyfikacja wewnętrzna . . . . .	13
<b>4</b>	<b>Badania</b>	<b>14</b>
4.1	Opóźnienia pojedynczego odcinka sieci . . . . .	14
4.2	Wpływ topologii na opóźnienia . . . . .	14
4.3	Czas stabilizacji sieci po zmianach . . . . .	14
<b>5</b>	<b>Uruchamianie i testowanie</b>	<b>15</b>
5.1	Przebieg testowania . . . . .	15
5.2	Napotkane problemy . . . . .	16
<b>6</b>	<b>Wnioski</b>	<b>17</b>
<b>7</b>	<b>Bibliografia</b>	<b>18</b>
<b>8</b>	<b>Spis rysunków, tablic i kodów źródłowych</b>	<b>19</b>
8.1	Spis rysunków . . . . .	19
8.2	Spis tablic . . . . .	19
8.3	Spis kodów źródłowych . . . . .	19







Rysunek 2: Schemat stanowiska typu CX

Stanowisko typu CP (Rysunek 1)	Stanowisko typu CX (Rysunek 2)
<ol style="list-style-type: none"> <li>1. 2 silniki AM3021-0C00-0000,</li> <li>2. Wyspa EK1100 z zestawem modułów IO: <ul style="list-style-type: none"> <li>• Terminal sieci EtherCAT EK1100,</li> <li>• 2-kanalowy moduł wyjść analogowych EL4132,</li> <li>• 4-kanalowy moduł wejść cyfrowych EL1004,</li> <li>• 2 4-kanalowe moduły wyjść cyfrowych EL2004,</li> <li>• 2-kanalowy moduł wejść analogowych EL3102,</li> </ul> </li> <li>3. Napęd serwo mechanizmów AX5203 (2 osiowy napęd),</li> <li>4. Komputer przemysłowy C6925,</li> <li>5. Zasilacz.</li> </ol>	<ol style="list-style-type: none"> <li>1. 2 silniki AM3021-0C00-0000,</li> <li>2. Zestaw modułów IO: <ul style="list-style-type: none"> <li>• 2-kanalowy moduł wyjść analogowych EL4132,</li> <li>• 4-kanalowy moduł wejść cyfrowych EL1004,</li> <li>• 2 4-kanalowe moduły wyjść cyfrowych EL2004,</li> <li>• 2-kanalowy moduł wejść analogowych EL3102,</li> </ul> </li> <li>3. Terminal sieci EtherCAT EK1100,</li> <li>4. Napęd serwo mechanizmów AX5203 (2 osiowy napęd),</li> <li>5. Modułowy komputer przemysłowy CX1020: <ul style="list-style-type: none"> <li>• Interfejs USB/DVI CX1020-N010 ,</li> <li>• Ethernet CX1020-N000,</li> <li>• CPU CX1020-0113,</li> <li>• Zasilacz CPU i magistrali I/O CX1100-0004,</li> </ul> </li> <li>6. Zasilacz.</li> </ol>

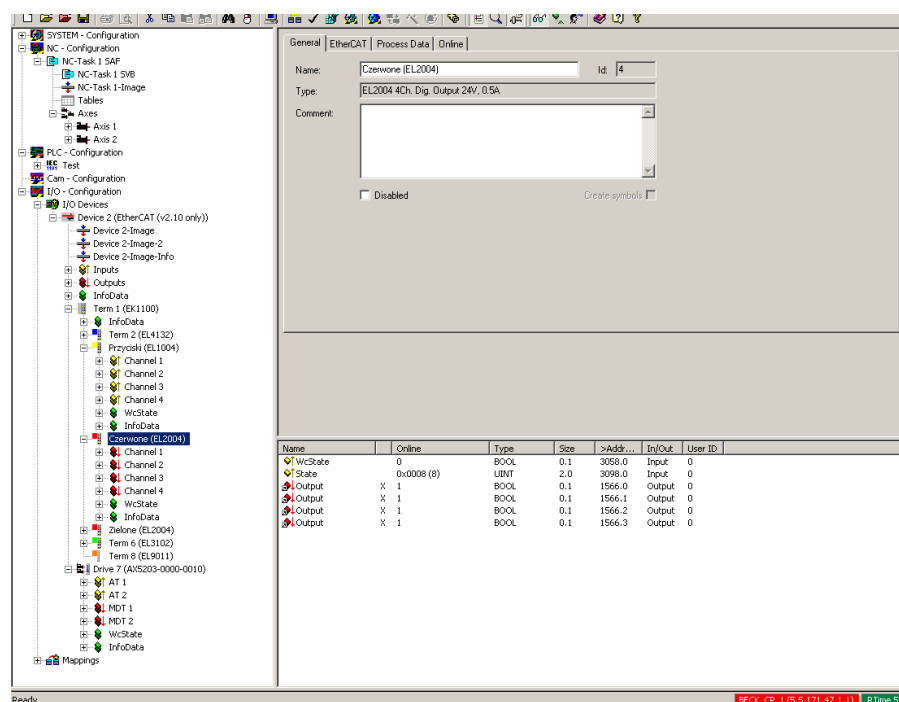
Tablica 1: Dostępne stanowiska laboratoryjne

w postaci dedykowanego urządzenia (ang. Hard PLC). 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 zaczyna się z nowej 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, a w luźno poło onej strukturze, co pozwala na np. stworzenie sprzężenia zwrotnego.

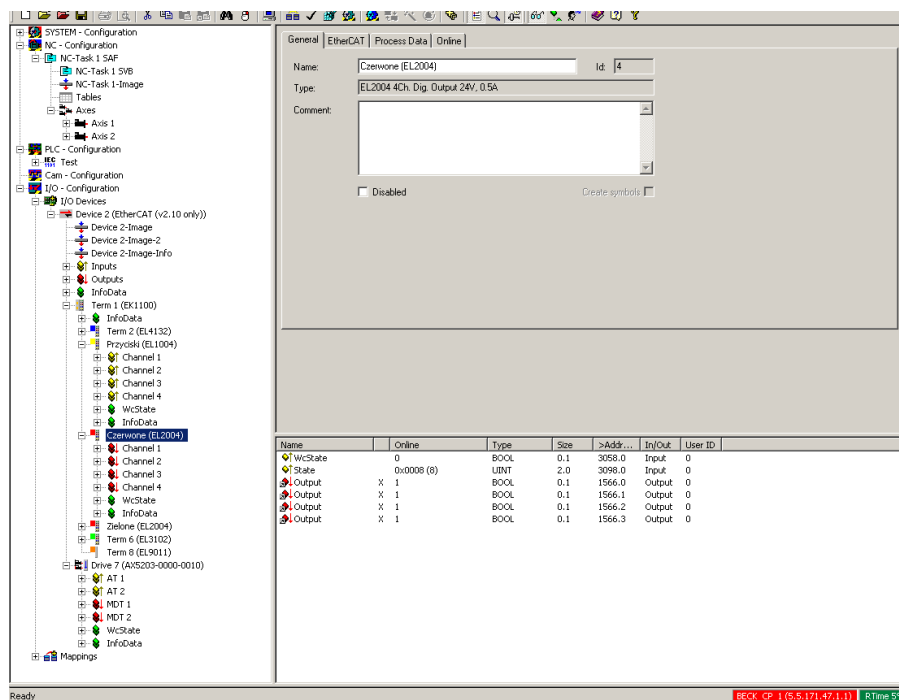
Stanowiska podłączone są do sieci lokalnej Ethernet w laboratorium, więc komunikacja z nimi odbywa się tak samo jak z każdym innym urządzeniem sieciowym. Podstawy programowania i korzystania ze sterowników autor poznał zapoznając się z odpowiednią literaturą [1, 2, 3, 4, 5] oraz uczęszczając w toku studiów na zajęcia obowiązkowe oraz specjalizacyjne. Konfigurację sterowników wraz z modułami przedstawiają Rysunki 3 oraz 4.



Rysunek 3: Konfiguracja stanowiska typu CP

### 1.1.2 Komputer

Projekt w całości był realizowany na laptopie autora, podłączanym do sieci w laboratorium. Na komputerze uruchomiana była maszyna wirtualna. Na jednej zainstalowane



Rysunek 4: Konfiguracja stanowiska typu CX

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).

## 1.2 Analiza tematu

Analiza tematu polegała przede wszystkim na zapoznaniu się z narzędziami programistycznymi do tworzenia oprogramowania sterownika oraz wizualizacji. W wyniku analizy autor poznał podstawy obsługi środowiska TwinCAT oraz jego elementów składowych 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.

## 1.3 EtherCAT

Standard EtherCAT został wprowadzony i jest rozwijany przez organizację EtherCAT Technology Group. Nazwa standardu jest w gruncie rzeczy skrótem od hasła „Ethernet for Control Automation Technology”

Zastosowano jednak istotnie inną metodę, która pozwoliła uzyskać dokładną synchronizację i krótkie opóźnienia transmisji.



## EtherCAT - Ethernet for Control Automation Technology

Zamiast dzielić czas transmisji na dane priorytetowe i mniej ważne lub nadawać priorytety wszystkim pakietom, skorzystano z faktu, że większość danych i rozkazów transmitowanych przez urządzenia przemysłowe wymagające krótkich czasów reakcji jest bardzo mała. W praktyce rozmiar porcji informacji wymienianych jednorazowo przez pojedyncze przyrządy podłączone do sieci jest nawet mniejszy niż minimalny rozmiar ramki stosowanej w klasycznym Ethernetie. Korzystając z tego faktu, 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 pożądane dane w tej pamięci.

Kolejne ramki nadawane są przez urządzenie pełniące rolę kontrolera i przesyłane do najbliższego urządzenia podrzędnego. Urządzenie to ma przydzielony adres, który jednoznacznie identyfikuje pewien fragment ramki (zwany datagramem), dzięki czemu może odczytać przeznaczone dla siebie dane. W przypadku gdy urządzenie podrzędne samo chce zainicjować komunikację, zapisuje pod odpowiednim adresem w odebranej ramce informacje przeznaczone dla innego urządzenia podłączonego do sieci EtherCAT. Niezależnie od tego, czy urządzenie coś zapisało, czy też tylko odczytywało fragment ramki, przesyła ją dalej do kolejnego z urządzeń. Duża szybkość tej metody transmisji wynika m.in. z tego, że nie ma potrzeby dekodowania całej ramki danych ani enkapsulacji w protokoły TCP/IP danych zapisywanych. Pozwala to błyskawicznie przekazywać ramki pomiędzy urządzeniami.

Wydajność 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  $\mu$ s.

### 1.3.1 Determinizm, przepustowość

### 1.3.2 Przetwarzanie „w locie”

### 1.3.3 Transmisja i synchronizacja w sieciach EtherCAT

### 1.3.4 EtherCAT Technology Group



Rysunek 5: Logo EtherCAT Technology Group (<http://www.ethercat.org>).

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 na dzień 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.

Jak widać na powyższej liście 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 firm interesuje się rozwojem przemysłowych protokołów komunikacyjnych.

### 1.3.5 Telegram EtherCAT

Jako pokazano na rysunku 2, telegram EtherCAT jest inkapsulowany w ramce Ethernet i zawiera jeden lub więcej datagramów EtherCAT dostarczanych do urządzeń slave.

Każdy datagram EtherCAT jest komendą, która zawiera nagłówek, dane i licznik roboczy. Nagłówek i dane są używane do wyspecyfikowania operacji, które muszą być wykonane przez slave, natomiast licznik roboczy jest aktualizowany przez slave informując mastera, że slave odebrał i przetwarza komendę.

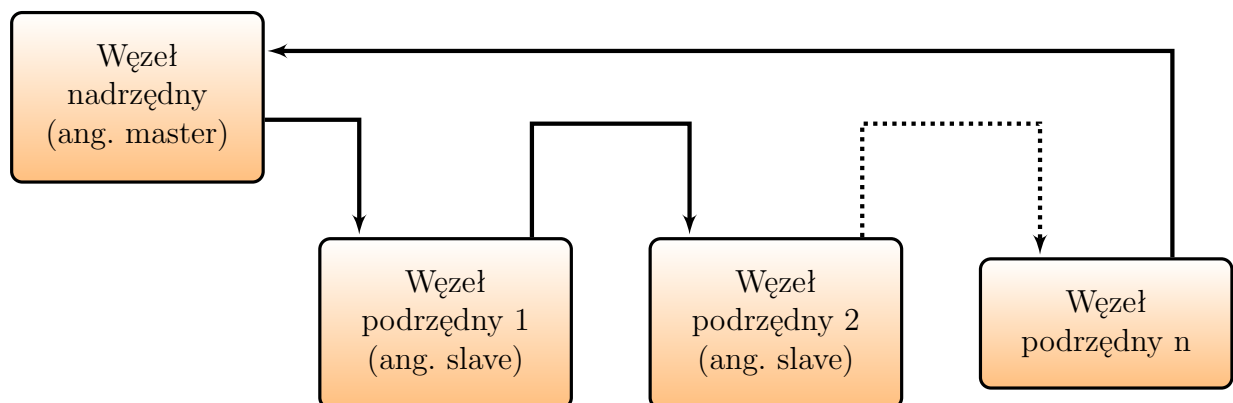
### 1.3.6

### 1.3.7

### 1.3.8

### 1.3.9

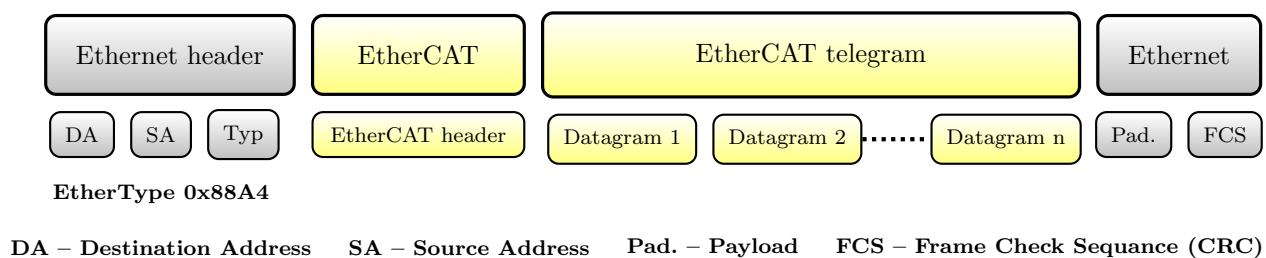
### 1.3.10



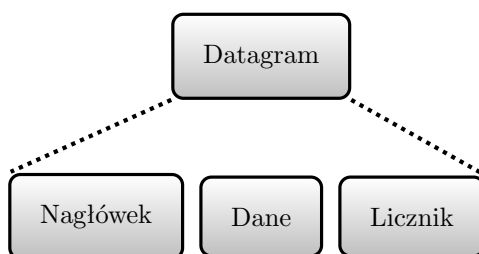
Rysunek 6: Przykładowa topologia sieci

## 1.4 Założenia

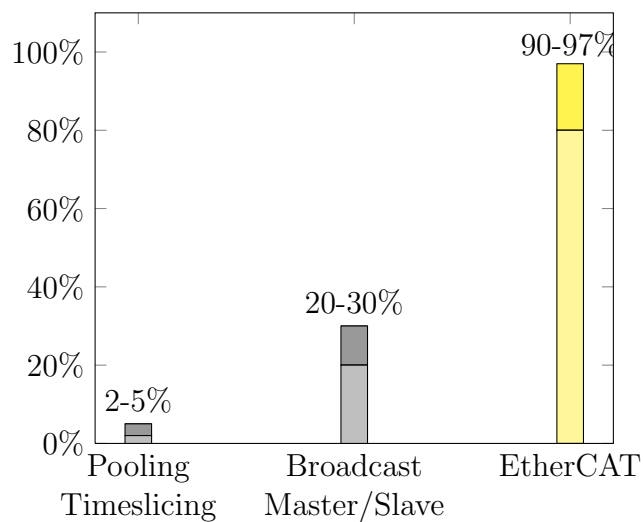
Oprogramowanie dla dostępnego stanowiska laboratoryjnego powinno zostać stworzone przy użyciu środowiska TwinCAT. Funkcjonalności robota wchodzące w skład pro-



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



Rysunek 8: Budowa datagramu



Rysunek 9: Współczynnik wykorzystania kanału transmisyjnego w Ethernetie (dwa pierwsze wykresy od lewej strony) i EtherCAT

jektu, to:

- sterowanie ręczne z pilota podłączonego bezpośrednio do sterownika,
- sterowanie ręczne z wizualizacji,
- sterowanie automatyczne,
- wizualizacja stanu stanowiska.

Powyżej zostały wymienione założenia podstawowe, jednak autor nie wyklucza zrealizowania dodatkowych zadań, które nie zostały zamieszczone w pierwotnej koncepcji realizacji projektu.

## **1.5 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,
- Przygotowanie stanowiska do współpracy z systemem wizualizacji,
- Testowanie i uruchamianie,
- Przedstawienie projektu i ewentualne korekty.

Powyższy plan pracy stanowił dla autora wyznacznik kolejnych działań. Jednak powszechnie wiadomo, że w praktyce poszczególne punkty są wymienne i wpływają na siebie wzajemnie.

## 2 Oprogramowanie sterownika

W niniejszym rozdziale opisane zostało oprogramowanie sterujące modelem. W kolejnych podrozdziałach zostanie przedstawiona specyfikacja zewnętrzna oraz wewnętrzna.

### 2.1 Specyfikacja zewnętrzna

Specyfikacja zewnętrzna przedstawiona w dalszej części podrozdziału zawiera opis, jak korzystać z oprogramowania wgranego do sterownika przez jego autora. Opisane zostało, jak ustawiać odpowiednie zmienne, aby uzyskać żądany efekt.

Lista zmiennych wejściowych i wyjściowych wymieniana między sterownikiem a modelem została już opisana w pierwszym rozdziale, w Tablicach ?? oraz ?. Pozostałe zmienne znajdują się w wewnętrznej pamięci sterownika.

Oprogramowanie może sterować modelem w sposób automatyczny lub ręczny. Tryb automatyczny w trybie obsługi magazynu zostanie opisany w podrozdziale 2.1.2. Tryb ręczny może być realizowany przy pomocy zadajnika podpiętego do sterownika lub przy pomocy przycisków umieszczonych na odpowiednim ekranie wizualizacji. W trybie tym o pracy robota decydujący jest stan przycisków. Dopuszczalne są wszystkie możliwe ruchy w przedziale od wyłącznika krańcowego do wartości maksymalnej.

### 2.2 Specyfikacja wewnętrzna

Podrozdział specyfikacja wewnętrzna opisuje sposób rozwiązania przez autora kwestii sterowania modelem przy użyciu dostępnego na stanowisku sterownika oraz poszczególnych trybów sterowania. W tworzeniu oprogramowania zostały wykorzystane następujące języki programowania:

- język drabinkowy (Ladder), wykorzystany do stworzenia głównych elementów programu,
- S7-SCL, który został zastosowany do korzystania z tablic. Niestety do korzystania z nich nie można zastosować języka LAD, ponieważ nie da się w nim odwoływać do elementów tablicy przez indeksy będące zmiennymi, a jedynie przez stałe. Po zapoznaniu się z dokumentacją okazało się, że taka możliwość istnieje w języku STL, ale jest to metoda skomplikowana w implementacji. Właśnie dlatego najlepszym i najprostszym rozwiązaniem okazują się S7-SCL, który jest kompilowany do kodu w języku STL.

## 3 Wizualizacja HMI

Zaimplementowana przez autora wizualizacja ma na celu zobrazowanie działania modelu oraz umożliwienie operatorowi wpływania na jego działanie. Kolejne podrozdziały zawierają opis specyfikacji zewnętrznej oraz wewnętrznej. Część odnosząca się do specyfikacji zewnętrznej jest skróconą instrukcją obsługi użytkownika. Specyfikacja wewnętrzna jest opisem, jak zostały zrealizowane poszczególne elementy i w jaki sposób wizualizacja współpracuje ze sterownikiem.

### 3.1 Specyfikacja zewnętrzna

Specyfikacja zewnętrzna przedstawiona w dalszej części podrozdziału stanowi skróconą instrukcję obsługi wizualizacji oraz opis możliwości oferowanych przez poszczególne ekrany.

Autor projektu wykorzystał w swojej pracy szereg elementów dostępnych standardowo w środowisku Simatic WinCC flexible. Podstawowymi elementami sterującymi są przyciski w trybie tekstowym oraz przeźroczystym. Głównymi obiektami służącymi do prezentacji informacji są: pola tekstowe, pola wejściowo-wyjściowe oraz pola daty i godziny. Dodatkowo celem uatrakcyjnienia wizualizacji wykorzystane zostały suwaki (ang. *slider*), obrazki oraz zegarek.

Obsługa tej części projektu jest realizowana za pomocą myszy i klawiatury podłączonych do komputera. Za pomocą klawiatury wybieramy interesujący nas ekran lub wprowadzamy żadaną wartość pozycji docelowej na ekranie testowania trybu automatycznego.

### 3.2 Specyfikacja wewnętrzna

Wizualizacja komunikuje się z komputera klasy PC ze sterownikiem za pośrednictwem protokołu Ethernet w sieci lokalnej. Odniesienia do odpowiednich adresów w pamięci sterownika dokonywane są za pomocą nazw symbolicznych zdefiniowanych w tablicy Tags. Do działania wizualizacja używa tylko jednej zmiennej wewnętrznej i jest to zmienna tablicowa *MagazynDTEnable* z elementami typu bool. Elementy te odpowiadają za wyświetlanie dat oraz godzin na ekranie ze stanem magazynu po kliknięciu na wybraną komórkę. Obsługa wyświetlania dat polega na tym, że po kliknięciu w wybrane pole ustawiana jest odpowiednia zmienna w tej tablicy na wartość *true*, a po zwolnieniu klawisza myszki na wartość *false*. Za zmiany te odpowiadają niewidzialne przyciski umieszczone na tych polach.

Wizualizacja wpływa na pracę sterownika poprzez zmianę pojedynczych bitów za pomocą umieszczonych na ekranie przycisków. Wpływa ona również poprzez modyfikowanie wybranych zmiennych odpowiadających pozycjom docelowym lub poprzez dodawanie odpowiednich zadań do kolejki. Bardziej zaawansowane operacje zostały zrealizowane za pomocą skryptów napisanych w języku VBScript, które są bardzo prostą i szybką opcją wykonywania bardziej zaawansowanych czynności.

## **4 Badania**

W niniejszym rozdziale opisany został przebieg przeprowadzonych badań oraz analiza ich wyników. W kolejnych podrozdziałach zostaną przedstawione kolejne różne eksperymenty.

### **4.1 Opóźnienia pojedynczego odcinka sieci**

### **4.2 Wpływ topologii na opóźnienia**

### **4.3 Czas stabilizacji sieci po zmianach**

## 5 Uruchamianie i testowanie

Rozdział ten zawiera podsumowanie przebiegu prac nad projektem. Opisane zostaną tu wszelkie poważne problemy, które wystąpiły w czasie realizacji projektu. Ponadto zawarto tu opis przebiegu procesu testowania.

### 5.1 Przebieg testowania

W procesie weryfikacji poprawności działania projektu zastosowano testowanie wstępujące.

Głównym testerem był autor projektu więc większość testów przebiegała na zasadzie białej skrzynki (ang. *white box*), bardzo często z użyciem podglądu stanu w środowisku TwinCAT System Manager. Takie testowanie pozwala stosunkowo łatwo wyszukać źródło błędu i je wyeliminować.

Autor kilka razy przeprowadzał testy stosując metodę czarnej skrzynki (ang. *black box*), nie biorąc pod uwagę zależności wykonywanych czynności, od realizowanego przez sterownik kodu. Kilukrotnie w czasie realizacji projektu do testów zgłaszały się osoby trzecie, które były nim zaintrygowane. Testy wykonane przez takie osoby są niezwykle cenne ze względu na dużą nieprzewidywalność oraz całkowitą niezależność działań od rozwiązań ze względu na brak ich znajomości.

W czasie realizacji autor stosował testowanie oparte na dwóch metodach analizy. Testowanie oprogramowania można wykonywać pod kątem analizy statycznej i dynamicznej. Analiza statyczna polega na sprawdzaniu kodu źródłowego i znajdowaniu w nim błędów bez uruchamiania sprawdzanego kodu. Ta metoda była stosowana poza laboratorium, gdzie brak był dostępu do sterownika i modelu. Podczas analizy dynamicznej oprogramowanie jest uruchamiane i badane pod kątem ścieżki przebiegu i czasu wykonywania. Ta metoda z kolei była najważniejsza i często wyniki tych testów były zaskakujące w stosunku do przeprowadzonych wcześniej z zastosowaniem analizy statycznej.

Ostatnim etapem testów były te przeprowadzone w obecności promotora oraz te wykonane przez niego. Ostatecznie oprogramowanie zostało zatwierdzone i uznane za spełniające wszystkie wstępne założenia przedstawione w podrozdziale 1.2.



## 5.2 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 zmiennych programu do odpowiednich fizycznych wejść oraz wyjść modelu oraz aktywowaniu tak przygotowanej konfiguracji, które z kolei wymusza zresetowanie systemu 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 męczące i czasochłonne. Okazało się, że w sterownikach firmy Beckhoff trzeba w specjalny sposób przygotować oprogramowanie, które ma być uruchamiane w sposób automatyczny, 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 reset 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:

aa

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

## 6 Wnioski

Tworzenie rozbudowanego oprogramowania do obsługi robota 3D pracującego w magazynie wysokiego składowania wymaga rozwiązania wielu problemów programistycznych. Autor w pierwszej kolejności zapoznał się z kursami programowania w środowisku Step 7 oraz WinCC flexible [7, 8, 9] dostępnymi w sieci.

W procesie tworzenia oprogramowania autor zapoznał się z wieloma zagadnieniami typowymi dla sterowników firmy Beckhoff. Przykładowo,

Wykonanie pojedynczego cyklu sterownika stworzonego przez autora oprogramowania w czasie testowania wynosiło od 1 do 3 ms. Czas ten jest zadowalający biorąc pod uwagę, że sterownik bez funkcji użytkownika oraz z pustym blokiem OB1 ma czas wykonania cyklu równy 1 ms oraz to, że oprogramowanie przynajmniej zdaniem autora jest rozbudowane.

Wizualizacja stanu robota stanowi bardzo atrakcyjną formę korzystania z urządzenia, która jest jednocześnie wyjątkowo przystępna dla osób nie znających zagadnień związanych z programowaniem sterowników przemysłowych. Wizualizacja będzie pełniła ważną rolę w prowadzeniu zajęć laboratoryjnych związanych ze sterownikami firmy Beckhoff

## 7 Bibliografia

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

- [1] Jerzy Kasprzyk: *"Programowanie sterowników przemysłowych"*, Wydawnictwa Naukowo-Techniczne WNT, Warszawa, 2007.
- [2] *"Programowalne sterowniki PLC w systemach sterowania przemysłowego"*, Politechnika Radomska, Radom, 2001.
- [3] Andrzej Maczyński: *"Sterowniki programowalne PLC. Budowa systemu i podstawy programowania"*, Astor, Kraków, 2001.
- [4] Zbigniew Seta: *"Wprowadzenie do zagadnień sterowania. Wykorzystanie programowalnych sterowników logicznych PLC."*, MIKOM Wydawnictwo, Warszawa, 2002.
- [5] Janusz Kwaśniewski: *"Programowalne sterowniki przemysłowe w systemach sterowania"*, Wyd. AGH, Kraków, 1999.
- [6] Dokumentacja producenta: *"Servo Drive AX5000"*, 28 wrzesień 2012.
- [7] Materiały szkoleniowe: Rudolf W. Meier: *"AX5000 – Motion control for high dynamic positioning"*, luty 2007.
- [8] Materiały szkoleniowe: *"Pierwsze kroki w TwinCAT System Manager i TwinCAT PLC Control"*, 29 październik 2007.
- [9] Materiały szkoleniowe: *"Podstawy obsługi programów: TwinCAT System Manager i TwinCAT PLC Control"*, 15 grudzień 2006.
- [10] Andrzej Gawryluk: *"EtherCAT – to nie takie trudne. Ethernet jako sieć real-time"*, Elektronika Praktyczna, 2/2010.
- [11] Michał Gosk: *"Szybkość, niezawodność, doskonała synchronizacja. EtherCAT - system przyszłości."*, Magazyn Sensor, 1/2013, kwiecień.
- [12] 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.

## 8 Spis rysunków, tablic i kodów źródłowych

### 8.1 Spis rysunków

Rysunek 1:	Schemat stanowiska typu CP . . . . .	3
Rysunek 2:	Schemat stanowiska typu CX . . . . .	4
Rysunek 3:	Konfiguracja stanowiska typu CP . . . . .	6
Rysunek 4:	Konfiguracja stanowiska typu CX . . . . .	7
Rysunek 5:	Logo EtherCAT Technology Group ( <a href="http://www.ethercat.org">http://www.ethercat.org</a> ). .	8
Rysunek 6:	Przykładowa topologia sieci . . . . .	9
Rysunek 7:	Ramka w transmisji EtherCAT i jej podział na datagramy . . .	10
Rysunek 8:	Budowa datagramu . . . . .	10
Rysunek 9:	Współczynnik wykorzystania kanału transmisyjnego w Etherne- cie (dwa pierwsze wykresy od lewej strony) i EtherCAT . . . . .	10

### 8.2 Spis tablic

Tablica 1:	Dostępne stanowiska laboratoryjne . . . . .	5
------------	---	---

### 8.3 Spis kodów źródłowych

## 9 Załączniki

- Oświadczenie o autorstwie,
- Płyta CD, na której znajdują się:
  - Kod oprogramowania wewnętrznego TwinCAT PLC Control,
  - Pliki projektu TwinCAT System Manager,
  - Kod wizualizacji oraz pliki projektu !?,
  - Plik wykonywalny wizualizacji !?
  - L<sup>A</sup>T<sub>E</sub>X-owe pliki źródłowe pracy magisterskiej.