POLITECHNIKA ŚLĄSKA

WYDZIAŁ AUTOMATYKI, ELEKTRONIKI I INFORMATYKI

KIERUNEK: AUTOMATYKA I ROBOTYKA

Praca dyplomowa inżynierska

Synteza układu regulacji wykorzystującego komponenty IoT

autor: Mateusz Dzieżok

kierujący pracą: dr hab. inż. Tomasz Kłopot

Gliwice, styczeń 2022

Spis treści

Streszczenie

Obiektem pracy jest implementacja układu regulacji wykorzystującego komponenty IoT. Układ regulacji odpowiada za sterowanie procesem działającym na sterowniku PLC, komunikując się poprzez sieć działającą w formie wymiany danych przez protokół OPC UA. Regulacja odbywa się na zasadzie algorytmu regulacji predykcyjnej MPC jako algorytm DMC.

Algorytm DMC zaimplementowany w języku Python działa na zewnątrz sterownika PLC w topologii wykorzystującej *edge computing*, regulując model w sterowniku na zasadzie symulacji obiektu wyższego rzędu. Proces strojenia regulatora DMC działa na zasadzie wyznaczenia FOPDT z pobranej odpowiedzi skokowej obiektu oraz wyznaczeniu nastaw na podstawie otrzymanego, przybliżonego modelu.

Słowa kluczowe: DMC, edge computing, IoT, MPC, OPC UA, PLC, Python.

Rozdział 1

Wstęp

W przemyśle 4.0 wiele rozwiązań technologicznych zostaje przeniesionych na urządzenia sieciowe oraz technologie zdalne. Przy wprowadzaniu nowych, bardziej zaawansowanych algorytmów regulacji, które wymagają mocy obliczeniowych, atrakcyjne staje się przeniesienie skomplikowanych obliczeń na zewnętrzne urządzenie. W tak powstałej sieci IoT działanie jest możliwie dzięki architekturze OPC, która pozwala na nawiązanie komunikacji pomiędzy urządzeniami.

Takie podejście daje możliwość regulacji układu algorytmem DMC, który należy do grupy MPC działającej na zasadzie obliczeń wykonywanych w ramach przewidywania przyszłych wartości. DMC jest algorytmem MPC, który jest obliczany na podstawie modelu układu i oferuje najlepsze wyniki dla złożonych układów wyższych rzędów w porównaniu do konkurencyjnych algorytmów regulacji. Przeniesienie regulatora poza sterownik oferuje większą moc obliczeniową, jednocześnie pozwalając na poprawę jakości regulacji.

Regulacja DMC polega na predykcji wyjścia układu poprzez analizę odpowiedzi skokowej pobranej z obiektu. Bazując na odpowiedzi skokowej oraz stosując przybliżenie obiektu modelem pierwszego rzędu z opóźnieniem (FOPDT) możliwe jest wyznaczenie nastaw regulatora DMC, aby oferował on najlepszą odpowiedź dla układów wyższych rzędów. Aby poprawnie ustawić regulację układu na zewnętrznym obiekcie, potrzebna do wyznaczenia jest jedynie odpowiedź skokowa obiektu regulowanego.

Rozdział 2

Analiza tematu

2.1 Sformułowanie problemu

Istotą problemu jest regulacja wstępnie nieznanego układu wyższego rzędu, dla którego przybliżenie modelem pierwszego rzędu nie jest zadowalające. Domyślnym rozwiązaniem jest regulacja PID na podstawie odpowiedzi układu używając sposobów doboru parametrów, lecz jest to rozwiązanie uniwersalne, wymagające poświęcenia jakości regulacji.

Analizując obiekty wyższych rzędów, należy zastosować inne podejście do regulacji układu. Dla rzeczywistych obiektów, wymagane jest, aby regulacja była odporna na wpływ zewnętrznych czynników takich jak zakłócenia. Aby rozwiązać wyżej sformułowany problem, wymagana jest regulacja obiektu, która byłaby w stanie poprawnie i wydajnie sterować obiektem regulacji. Obiektem przyjętym w symulacji jest obiekt cieplny zawierający zbiornik wody oraz grzałkę. Dla efektywnego sterowania procesem wymagana jest bardziej zaawansowana regulacja, pozwalająca na efektywną regulację układów wyższych rzędów.

2.2 Osadzenie tematu w kontekście aktualnego stanu wiedzy o poruszanym problemie

Obiekty cieplne często są obiektem regulacji w przemyśle. W wielu dziedzinach techniki wymagane jest stosowanie zaawansowanych rozwiązań, aby sterować mocą

elementów, a co za tym idzie, regulowaniem temperatury. Obiekty regulacji często stanowią problem w postaci transportu energii cieplnej koncentrowanej w całym układzie. Oznacza to, że równania wymagane, aby zapisać przebiegający proces, są zaawansowane i często wymagające przybliżeń.

Dla elementów cieplnych popularnym rozwiązaniem jest stosowanie przybliżenia modelem inercyjnym pierwszego rzędu z czasem opóźnienia (FOPDT), który oferuje znaczące przybliżenie, lecz wciąż zapewniając że błąd znajduje się w granicach wystarczających do prowadzenia regulacji.

2.3 Studia literaturowe - opis znanych rozwiązań oraz algorytmów

Regulacja MPC jest bardzo rozwiniętym oraz popularnym zagadnieniem, które jest implementowane do prowadzenia regulacji wykonując obliczenia algorytmu na platformie sterownika PLC. Wykorzystany w tym projekcie DMC, będący algorytmem MPC, został opisany w publikacji *Sterowanie Zaawansowane obiektów przemysłowych* [3]. Dzięki użyciu regulacji DMC istnieje wiele rozwiązań doboru parametrów regulatora, a jedną z takich metod jest strategia strojenia regulatorów DMC ze zredukowanymi horyzontami, jak przedstawiono w publikacji [1].

Rozdział 3

Wymagania i narzędzia

Do realizacji projektu wymagany jest sterownik typu S7-1500, jak i środowisko programowania sterowników do prowadzenia symulacji obiektu. Aby wprowadzić algorytm DMC w formie *edge computing*, potrzebna jest zewnętrzna maszyna podłączona do tej samej sieci, w której znajduje się sterownik. Takie rozwiązanie sieci zapewni komunikację między nimi.

Narzędzia wykorzystane w projekcie

- TIA Portal
- S7-PLCSIM (opcjonalne, aby symulować sterownik)

Języki programowania wykorzystane w projekcie

- S7-SCL
- Python 3.7

Rozdział 4

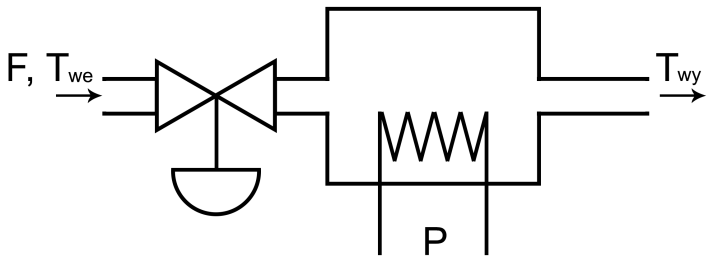
Symulacja układu

W tym rozdziale rozwijane jest zagadnienie symulacji układu, aby jak najbardziej odpowiadała ona rzeczywistemu obiektowi. Realizowana symulacja zostaje użyta do wizualizacji procesu oraz do przedstawienia regulacji. Symulację tę będzie prowadził sterownik PLC S7-1500 metodą Eulera w języku S7-SCL.

4.1 Modelowanie rzeczywistego układu

Aby zrealizować symulację układu, należy przybliżyć rzeczywisty obiekt do modelu wyższego rzędu, aby zminimalizować różnice i błędy pomiędzy rzeczywistym układem a modelem. Układ został przedstawiony za pomocą schematu na rysunku 4.1.

Aby obliczenia symulacji były najbardziej podobne do rzeczywistego, ciągłego procesu, należy dokonywać obliczeń symulacji znacznie częściej niż zmienne w tym procesie zostają obserwowane lub pobierane do pozostałych części projektu. Należy zapewnić, aby symulacja odbywała się częściej niż przebiega regulacja, a funkcjonowanie regulatora przebiegało podobnie jak na obiekcie ciągłym w czasie.



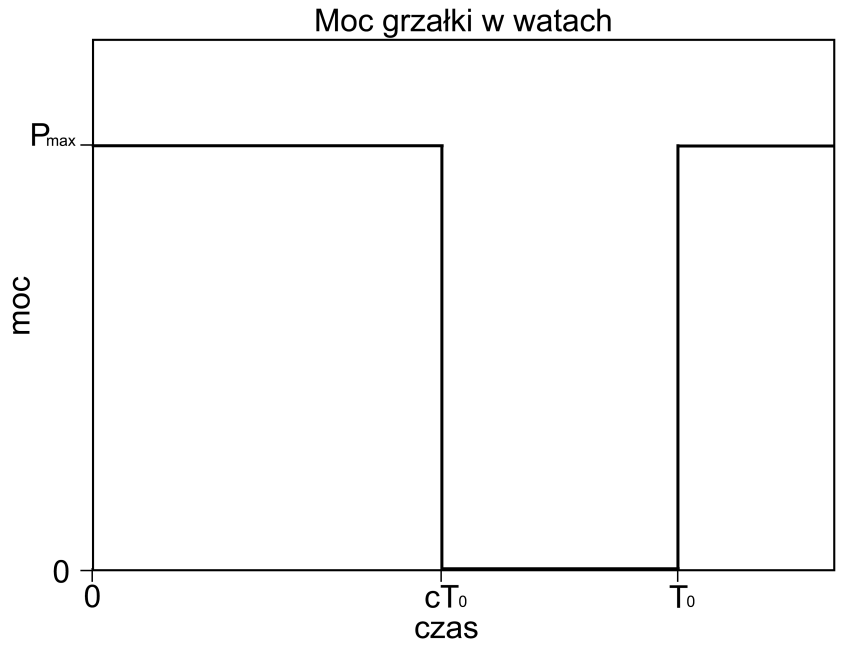
Rysunek 4.1: Schemat układu z grzałką.

Głównym elementem instalacji jest zbiornik z grzałką elektryczną umieszczoną wewnątrz o łącznej mocy rzeczywistej $P_{max} = 12307,8\ W$. Woda do układu jest doprowadzana z sieci wodociągowej z temperaturą T_{we} oraz opuszcza układ do użytku z temperaturą T_{wy}. Pomiary temperatur odbywają się bezpośrednio na wyjściu i wejściu podgrzewacza.

4.2 Grzałka jako element o periodycznych zmianach mocy w czasie

Grzałka w układzie jest załączana lub wyłączana za pomocą przekaźnika, który jest sterowany sygnałem prostokątnym, definiując wypełnienie jako $c*100\%$, gdzie $c \in [0, 1]$ przy z góry narzuconym okresie T_0.

To rozwiązanie umożliwia zmianę średniej mocy grzałki w zależności od parametru c.



Rysunek 4.2: Przebieg periodycznych zmian mocy grzałki w czasie dla wypełnienia.

Przy wystarczająco małym czasie T_0, na podstawie analizy modelu matematycznego podgrzewacza jako układu o parametrach skupionych [2], można stwierdzić, że przy dostatecznie niskim czasie T_0 sygnału odpowiedź układu głównie zależy od składowej stałej $c * P_{max}$, która jest wartością średnią z okresu T_0.

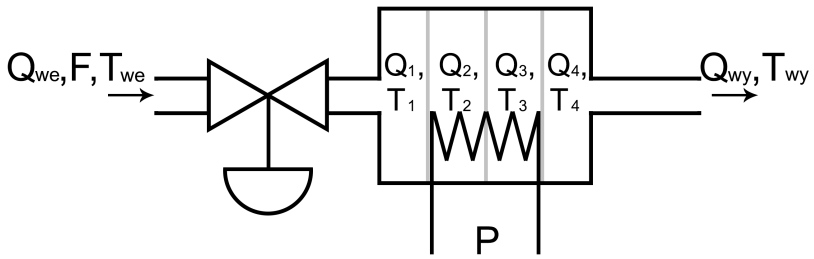
4.3 Wykorzystanie metody Eulera w algorytmie

Metoda Eulera to procedura pierwszego rzędu do rozwiązywania równań różniczkowych dla określonej wartości początkowej. Jest to jedna z podstawowych jawnych metod całkowania numerycznego równań różniczkowych zwyczajnych.

Jako że metoda Eulera jest metodą pierwszego rzędu, to oznacza, że błąd lokalny jest proporcjonalny do kwadratu kroku, a błąd globalny jest proporcjonalny do wielkości kroku.

Aby wykorzystać tę metodę oraz nie przybliżyć modelu zbyt znacząco, obliczenia będą prowadzone dla czterech zmiennych wewnątrz zbiornika rozłożonych wzdłuż przepływu w układzie oraz uzależnionych od umiejscowienia grzałki. Dzięki

takiemu podejściu model układu oferuje mały błąd względem układu rzeczywistego i oferuje model wyższego rzędu.



Rysunek 4.3: Schemat układu z grzałką z podziałem na zmienne do modelowania.

Na podstawie schematu 4.3 stworzone są cztery równania różniczkowe, które określają dynamikę wewnątrz zbiornika.

$$\frac{dQ_1}{dt} = Q_{we} - Q_1$$

$$\frac{dQ_2}{dt} = Q_1 - Q_2 + \frac{1}{2}c * P_{max}$$

$$\frac{dQ_3}{dt} = Q_2 - Q_3 + \frac{1}{2}c * P_{max}$$

$$\frac{dQ_4}{dt} = Q_3 - Q_4$$

Stosując zamianę strumieni ciepła na temperatury:

$$Q_{we} = \rho * c_w * F * T_{we}$$

$$Q_{wy} = \rho * c_w * F * T_{wy}$$

$$Q_1 = \rho * c_w * \frac{1}{4}V * T_1$$

$$Q_2 = \rho * c_w * \frac{1}{4}V * T_2$$

$$Q_3 = \rho * c_w * \frac{1}{4}V * T_3$$

$$Q_4 = \rho * c_w * \frac{1}{4}V * T_4$$

Otrzymujemy:

$$\frac{dT_1}{dt} = \frac{F}{\frac{1}{4}V}(T_{we} - T_1)$$

$$\frac{dT_2}{dt} = \frac{F}{\frac{1}{4}V}(T_1 - T_2) + \frac{1}{2}\frac{c * P_{max}}{\rho * c_w * V}$$

$$\frac{dT_3}{dt} = \frac{F}{\frac{1}{4}V}(T_2 - T_3) + \frac{1}{2}\frac{c * P_{max}}{\rho * c_w * V}$$

$$\frac{dT_4}{dt} = \frac{F}{\frac{1}{4}V}(T_3 - T_4)$$

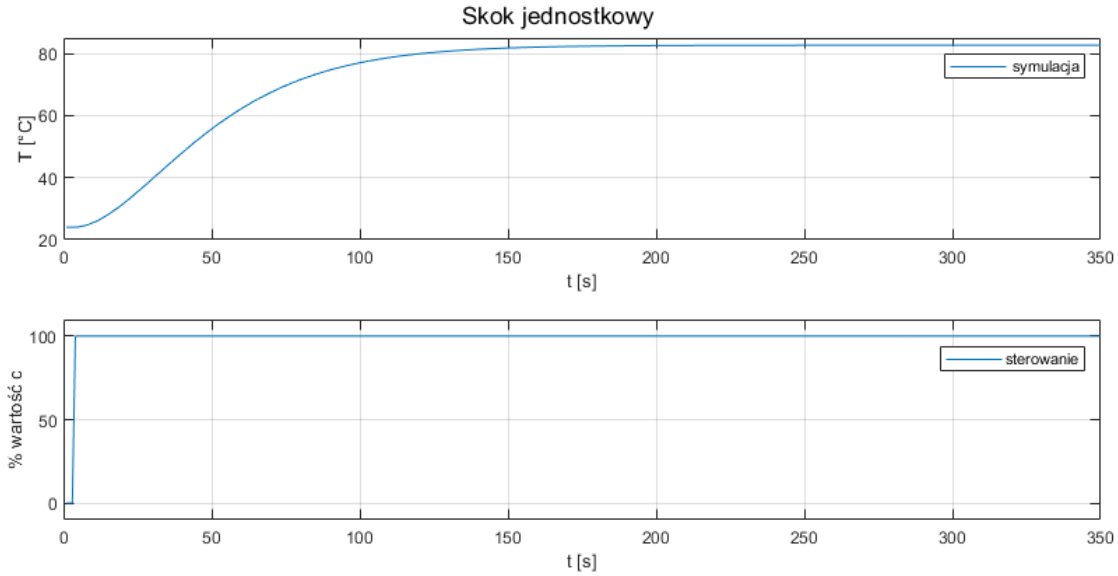
$$T_{wy} = T_4$$

Obliczenia numeryczne wykonane metodą Eulera wymagają podziału równań różniczkowych na strony oraz wykonywanie kalkulacji z krokiem całkowania i zapis ich jak przedstawiono poniżej 4.4 w zaimplementowanym algorytmie języka S7-SCL.

```
1  //symulacja modelu
2  FOR #m := 1 TO "DBc".NIS DO
3      "DBc".h := "DBc".Dt / "DBc".NIS;
4
5      "DBc".V / 4))) * ("DBc".x4 - "DBc".Tot);
6      "DBc".f4 := ("DBc".F * ("DBc".Thin - "DBc".x4)) / (1 *
           "DBc".V / 4);
7      "DBc".x4 := "DBc".x4 + "DBc".f4 * "DBc".h;
8
9      ("DBc".V / 2)) - ("DBc".kA / ("DBc".cw * "DBc".ro * (1
           * "DBc".V / 4))) * ("DBc".x3 - "DBc".Tot);
10     "DBc".f3 := ("DBc".F * ("DBc".x4 - "DBc".x3)) / (1 * "
           DBc".V / 4) + "DBc".Pg / ("DBc".cw * "DBc".ro * (2 *
            "DBc".V / 4));
11     "DBc".x3 := "DBc".x3 + "DBc".f3 * "DBc".h;
12
13     ("DBc".V / 2)) - ("DBc".kA / ("DBc".cw * "DBc".ro * (1
           * "DBc".V / 4))) * ("DBc".x2 - "DBc".Tot);
14     "DBc".f2 := ("DBc".F * ("DBc".x3 - "DBc".x2)) / (1 * "
           DBc".V / 4) + "DBc".Pg / ("DBc".cw * "DBc".ro * (2 *
            "DBc".V / 4));
15     "DBc".x2 := "DBc".x2 + "DBc".f2 * "DBc".h;
16
17     "DBc".V / 4))) * ("DBc".x1 - "DBc".Tot);
18     "DBc".f1 := ("DBc".F * ("DBc".x2 - "DBc".x1)) / (1 * "
           DBc".V / 4);
19     "DBc".x1 := "DBc".x1 + "DBc".f1 * "DBc".h;
20     "DBc".t := "DBc".t + "DBc".h;
21 END_FOR;
```

Rysunek 4.4: Symulacja metodą Eulera w języku S7-SCL.

Realizacja takich obliczeń prowadzi do powstania symulacji układu, który może podlegać regulacji. Rysunek 4.5 przedstawia odpowiedź układu symulowanego na skok jednostkowy.



Rysunek 4.5: Odpowiedź symulacji układu na skok jednostkowy.

4.3.1 Wykorzystanie idei UDYN

Aby błąd w obliczeniach ograniczyć do minimum, należy zastosować jak najmniejszy krok całkowania h. Aby prowadzenie obliczeń było dokładniejsze, bez uruchamiania części kodu odpowiedzialnych za komunikację lub kodu aktualizacji zmiennych, definiuje się krok całkowania w zależności od kroku obserwacji.W tym projekcie jest to czas cyklu w bloku cyklicznym wykonującym obliczenia symulacji.

$$h = \frac{Dt}{NIS}$$

gdzie NIS to liczba kroków całkowania h w kroku obserwacji Dt.

Obliczenia symulacji są wykonywane z krokiem całkowania tzn. NIS razy częściej niż krok obserwacji odpowiadający cyklowi bloku cyklicznego.

Rozdział 5

Komunikacja

5.1 Internet Rzeczy oraz przetwarzanie brzegowe

5.1.1 Internet Rzeczy – IoT

Internet Rzeczy (ang. *IoT – Internet of Things*) to koncept wykorzystywania urządzeń elektronicznych przez sieć. Urządzenia te mogą również komunikować się pomiędzy sobą i wymieniać informacjami za pośrednictwem sieci bez interwencji człowieka. Jest to szerokie pojęcie obejmujące wiele różnorodnych urządzeń, takich jak urządzenia inteligentnych domów, smartfonów czy samochodów z funkcjami smart.

Wraz z wkraczaniem w Przemysł 4.0 mający na celu integrację inteligentnych maszyn, systemów oraz modyfikowanie procesów produkcyjnych, które zwiększają ich wydajność, wiele rozwiązań z idei IoT wkracza do produkcji, energetyki czy urządzeń medycznych.

5.1.2 Przetwarzanie brzegowe – *edge commputing*

Przetwarzanie Brzegowe (ang. *edge commputing*) to sposób podejścia do połączenia urządzeń w ramach Internetu Rzeczy polegający na rozproszeniu środowisk obliczeniowych. Brzeg sieci oznacza umiejscowienie urządzeń zajmujących się kalkulacjami lub przechowywaniem danych bliżej miejsca, gdzie te dane są generowane (w przeciwieństwie do przechowywania w chmurze). Bliżej oznacza, że urządzenie

znajduje się w tej samej sieci lub sieci sąsiadującej, minimalizując liczbę urządzeń pośredniczących wymianie danych. Takie podejście do zagadnienia daje możliwość szybkiej analizy i reakcji na dane pobierane z układu centralnego.

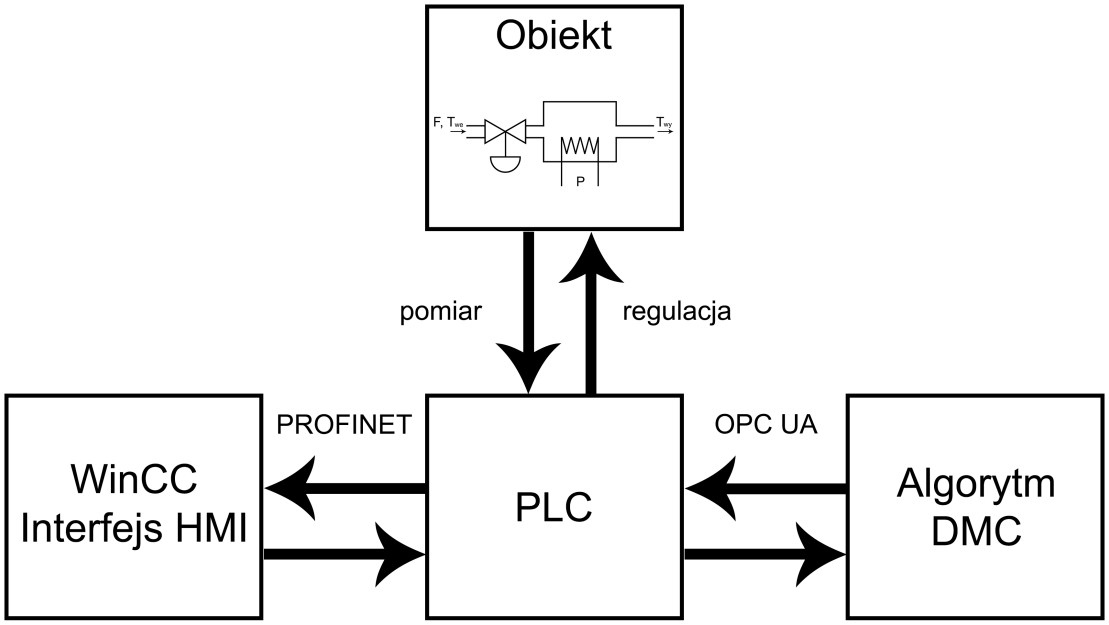
Rozwiązanie w formie przetwarzania brzegowego jest bardzo optymalne, gdy liczba obliczeń wykracza poza możliwości obliczeniowe lub bardzo obciąża obliczeniowo urządzenie centralne, takie jak sterownik posiadający ograniczone zasoby. Gdy stosuje się bardziej zaawansowane algorytmy obliczeniowe, w szczególności takie, które wymagają dużej ilości pamięci bądź krótkiego czasu obliczeń, rozwiązanie, jakim jest przetwarzanie brzegowe, oferuje najwyższy stopień elastyczności i optymalizacji wykonywania takich algorytmów.

5.2 Wykorzystanie OPC UA

Aby umożliwić wymianę danych pomiędzy urządzeniami IoT, wymagane jest zastosowanie rozwiązania technologii do komunikacji. Standard OPC UA (ang. *Open Platform Communications Unified Architecture*) jest protokołem komunikacji pomiędzy urządzeniami w automatyce przemysłowej, opartym o komunikację klient serwer, umożliwiającym komunikację pomiędzy różnymi platformami.

OPC UA koncertuje się na komunikacji pomiędzy urządzeniami przemysłowymi oraz systemami gromadzenia i kontroli danych. Działa jako standard, który wraz z rozwojem spełnia wymagania technologii IoT oraz Przemysłu 4.0, zapewniając przy tym bezpieczeństwo danych.

Aby poprawnie zaimplementować komunikację OPC UA w celu wykorzystania urządzenia do prowadzenia obliczeń, wymagane jest stosowanie sterownika PLC, który będzie działać jako urządzenie zajmujące się obiektem sterowania - tutaj jest to prowadzenie symulacji, komunikacja z HMI oraz zarządzanie danymi takimi jak parametry lub dane z pomiarów. Ten sam sterownik będzie działał jako serwer w komunikacji. Urządzenie wykonujące algorytm prowadzi komunikację ze sterownikiem jako klient. Za pośrednictwem serwera na PLC odbywa się wymiana danych pomiarów i danych regulacji procesu.



Rysunek 5.1: Schemat prowadzonej komunikacji.

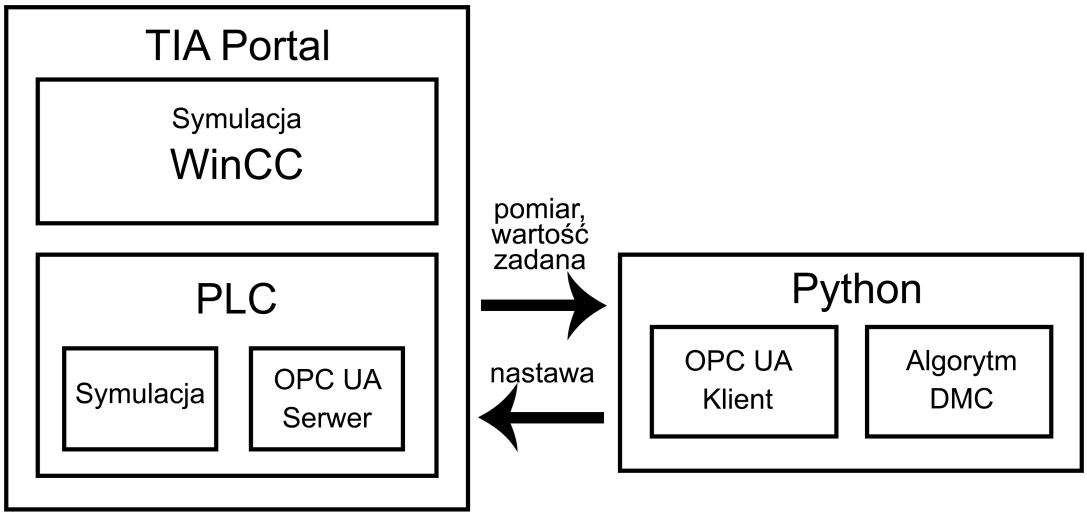
Aby zrealizować projekt, zastosowano wcześniej omówioną symulację modelu obiektu w celu prowadzenia regulacji oraz symulację interfejsu użytkownika z programu TIA Portal. Aby symulacja układu oraz przesył informacji poprzez OPC UA zostały poprawnie zrealizowane, wymagane są odpowiednie okresy przesyłu danych. Dzięki temu procesy regulacji i obserwacji będą przebiegały właściwie.

Tablica 5.1: Czas cyklu poszczególnych elementów w komunikacji.

element	czas cyklu	komentarz
PLC Cycle Interrupt	100 ms	
symulacja z UDYN	10 ms	wynikająca z 4.3.1
PROFINET HMI Tags	100 ms	minimalne odświeżanie
OPC UA	min. 100 ms	w zależności od zmian wartości zmiennej
DMC sampling time	1 s	wnika z doboru nastaw 6.2

Prowadzenie obliczeń algorytmu DMC odbywa się w środowisku Python na osobnej maszynie, komunikując się za pośrednictwem OPC UA ze sterownikiem PLC. Wymiana danych w tym projekcie może zostać przedstawiona tak jak na

rysunku 5.1. Dokładniej opisując, można przedstawić to tak, że zewnętrzne urządzenie prowadzące komunikację z serwerem wykonuje zapytania o zmienne oraz zmienia wartość innych zmiennych tak jak przedstawiono na schemacie 6.1, utrzymując ciągłe połączenie, prowadząc odczyt i zmiany cyklicznie w cyklu prowadzenia obliczeń algorytmu DMC jak ustalono w tabeli 5.1.



Rysunek 5.2: Schemat ideowy implementacji.

5.2.1 Uzasadnienie rozwiązania

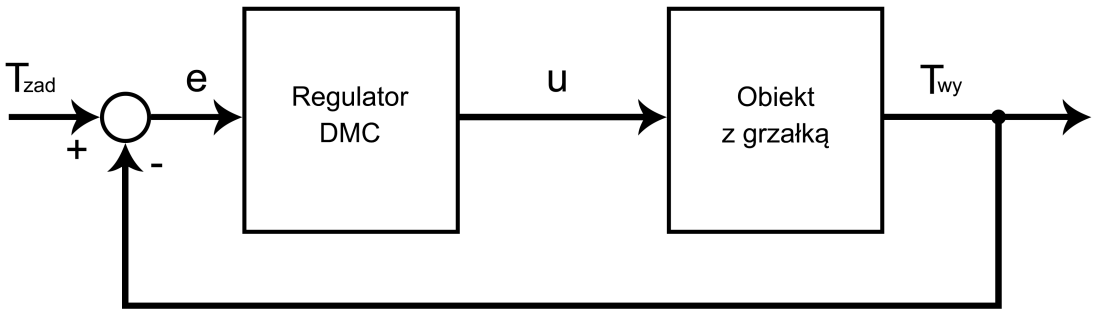
Zastosowanie rozwiązania, jakim jest *edge computing* może wydawać się stratne dla wyniku regulacji, przez istnienie opóźnień przesyłu danych w sieci. Dla wolno reagujących układów problem ten staje się znikomy. Dla symulacji układu działającej w cyklu 100 ms oraz regulacji DMC działającej co 1 sekundę czas transportu danych nie wpływa znacząco na wynik regulacji, w szczególności gdy czas przepływu pakietów w tej samej sieci lokalnej zazwyczaj nie przekracza 20 ms, a w poprawnie skonfigurowanych sieciach nie jest większy niż 1 ms.

Rozdział 6

Algorytm MPC – DMC

6.1 Wykorzystanie algorytmu DMC

Algorytm DMC należy do algorytmów typu MPC, jest to technika regulacji należąca do zaawansowanych. Skrót MPC rozwija się jako *Model (Model-based) predictive control*, co oznacza algorytm regulacji predykcyjnej.



Rysunek 6.1: Schemat układu regulacji.

Algorytmy MPC polegają na liczeniu optymalizacji pewnej funkcji celu z określonymi horyzontami. W metodach MPC wykorzystuje się zagadnienie regulacji predykcyjnej z przesuwnym horyzontem, oznacza to, że dla danej chwili są przewidywane przyszłe wartości zadane w wektorze określonym horyzontem predykcji,

ten horyzont nie zmienia rozmiarów, lecz w kolejnych chwilach czasu służy do przewidywań kolejnych N wartości, działając jako horyzont przesuwny.

W przypadku niedokładnego modelu bądź występujących niemierzalnych zakłóceń, predykcje mogą różnić się od wartości mierzonych, dlatego w układzie stosuje się również sprzężenie zwrotne. Jednym z pierwszych opracowanych algorytmów MPC był DMC czyli *Dynamic Matrix Control*, który polega na wykorzystaniu liniowego modelu obiektu w postaci odpowiedzi skokowych oraz funkcji z karami za zmiany sterowań. Podejście do stosowania odpowiedzi wyjścia obiektu na skok jednostkowy jest bardzo wygodnym sposobem modelowania dynamiki obiektu. Dlatego też do regulatora jest używana odpowiedź z układu jak przedstawiono na rysunku 4.5.

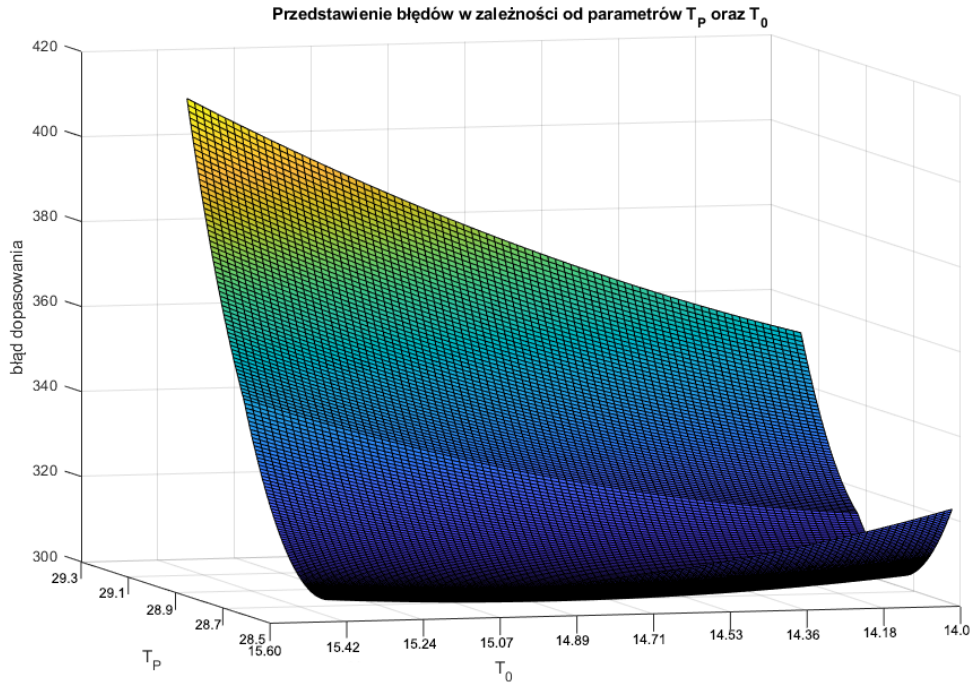
6.2 Dobieranie nastaw DMC

Dobieranie nastaw dla regulatora odbywa się na podstawie analizy modelu w postaci FOPDT (ang. *first-order plus deadtime*). Aby wyznaczyć model FOPDT dla rzeczywistego układu (w tym wypadku symulacji), należy pobrać skok jednostkowy i wyznaczyć model FOPDT najbliższy odpowiedzi układu rzeczywistego, a na podstawie FOPDT prowadzić wyliczenia parametrów regulatora DMC.

6.2.1 Wyznaczanie FOPDT

FOPDT to *first-order plus deadtime*, co można przetłumaczyć jako model pierwszego rzędu z opóźnieniem. Jest on używany do przybliżenia dynamicznej odpowiedzi procesu o wyższych rzędach.

Aby otrzymać FOPDT, zastosowano metodę najmniejszych kwadratów w celu wyznaczenia parametrów posiadających parametry najbliższe odpowiedzi skokowej układu rzeczywistego. Wyniki zostały przedstawione w formie trójwymiarowej, przedstawiającej błąd obliczony między odpowiedzią na skok jednostkowy modelu FOPDT z parametrami T_P oraz T_0 a odpowiedzią na skok jednostkowy układu.



Rysunek 6.2: Obliczanie błędów metodą najmniejszych kwadratów.

Na podstawie obliczeń z danych przedstawionych na rysunku 6.2 wyznaczono parametry, dla których błąd jest najmniejszy i otrzymano parametry do modelu FOPDT:

$$k_0 = 58.75$$

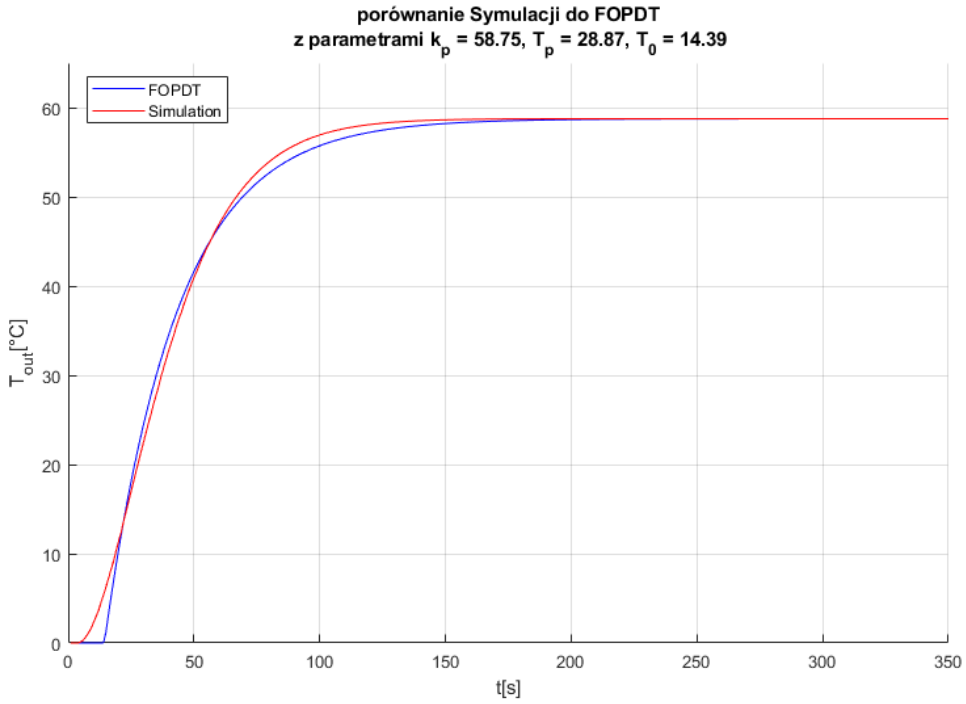
$$T_0 = 14.93$$

$$T = 28.87$$

Transmitancja na podstawie dobranych parametrów:

$$K(s) = \frac{58.75e^{-14.93s}}{28.87s + 1}$$

Otrzymując transmitancję, możliwe jest przedstawienie porównania otrzymanego modelu FOPDT do rzeczywistej odpowiedzi układu wyższego rzędu na wykresie jak na rysunku 6.3.



Rysunek 6.3: Dopasowanie FOPDT do odpowiedzi skokowej.

6.2.2 Obliczenia parametrów regulatora DMC na podstawie otrzymanego modelu FOPDT

Aby dobrać najlepsze parametry, a co za tym idzie, wielkości wektorów i macierzy w algorytmie, należy dobrać możliwie jak najmniejsze rozmiary, aby oszczędzić moc obliczeniową i czas potrzebny na kalkulację, ale jednocześnie trzeba dobrać je wystarczająco duże, aby jakość regulacji nie spadła znacząco. Dla algorytmów DMC i instalacji SISO [1] (*single input single output* tzn. z pojedynczym wejściem pojedynczym wyjściem) zostały rozwiązane jako optymalizacja poprzez dobranie parametrów następująco:

Na podstawie modelu FOPDT

$$K(s) = \frac{k_0 e^{-T_0 s}}{Ts + 1} = \frac{58.75 e^{-17.37 s}}{28.87 s + 1}$$

zostaje przeprowadzona procedura strojenia, gdzie parametry zaokrąglane są do najbliższej liczby całkowitej.

Czas próbkowania (*sampling time*):

$$T_C \leq 0.1T$$

$$T_C = 1$$

Horyzont kontroli (*control horizon*):

$$H_C = 2$$

Horyzont predykcji (*prediction horizon*):

$$H_P = \frac{T}{T_C} + \frac{T_0}{T_C} = 43$$

Dynamiczny horyzont (*dynamic horizon*):

$$H_D = \frac{3T}{T_C} + \frac{T_0}{T_C} = 101$$

Współczynnik tłumienia (*supression coefficient*):

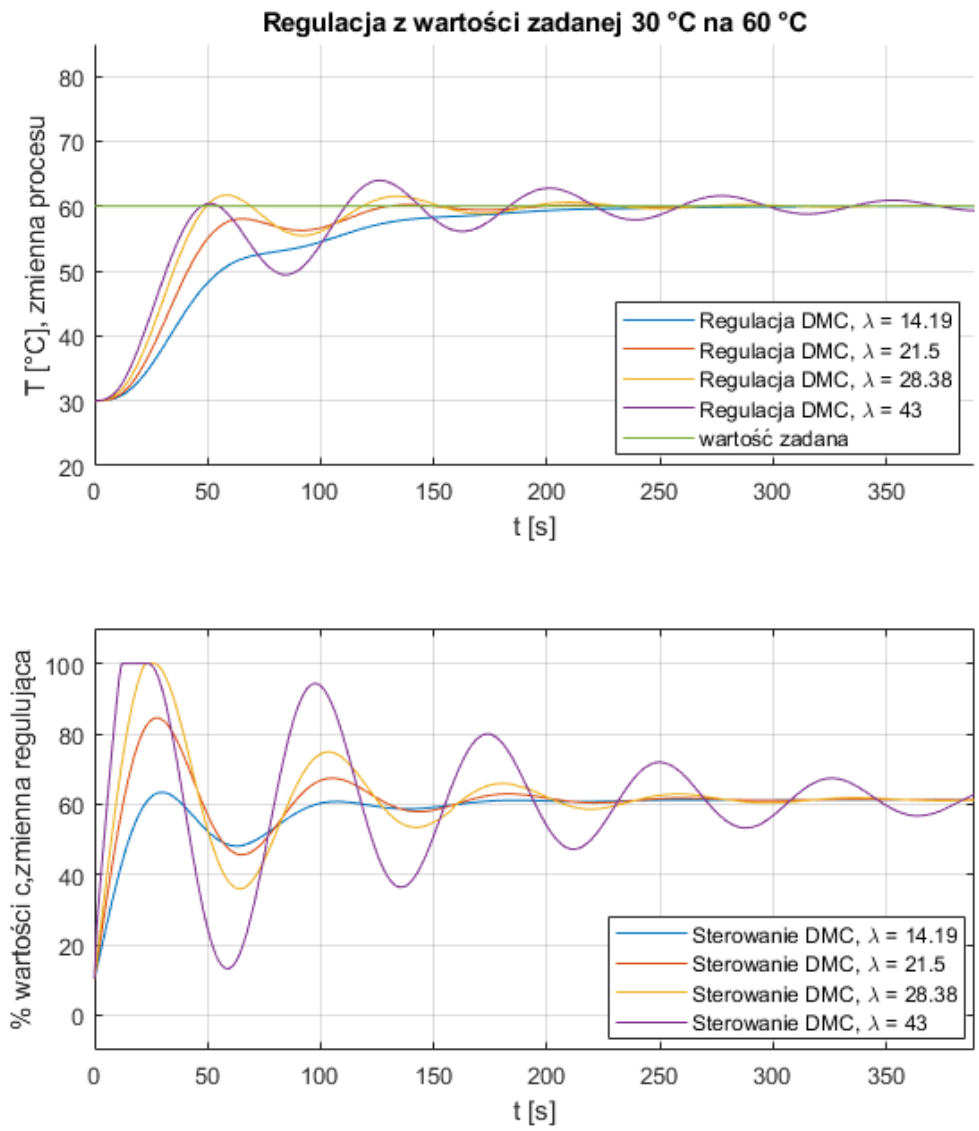
$$\lambda = x * k^2 * H_p$$

$$x_{min} = \frac{73}{5000} * \frac{1}{1 + \frac{T_0}{T}} = 0.0097 \leq x$$

$$x \geq 0$$

Tablica 6.1: Proponowane parametry x dla współczynnika tłumienia λ.

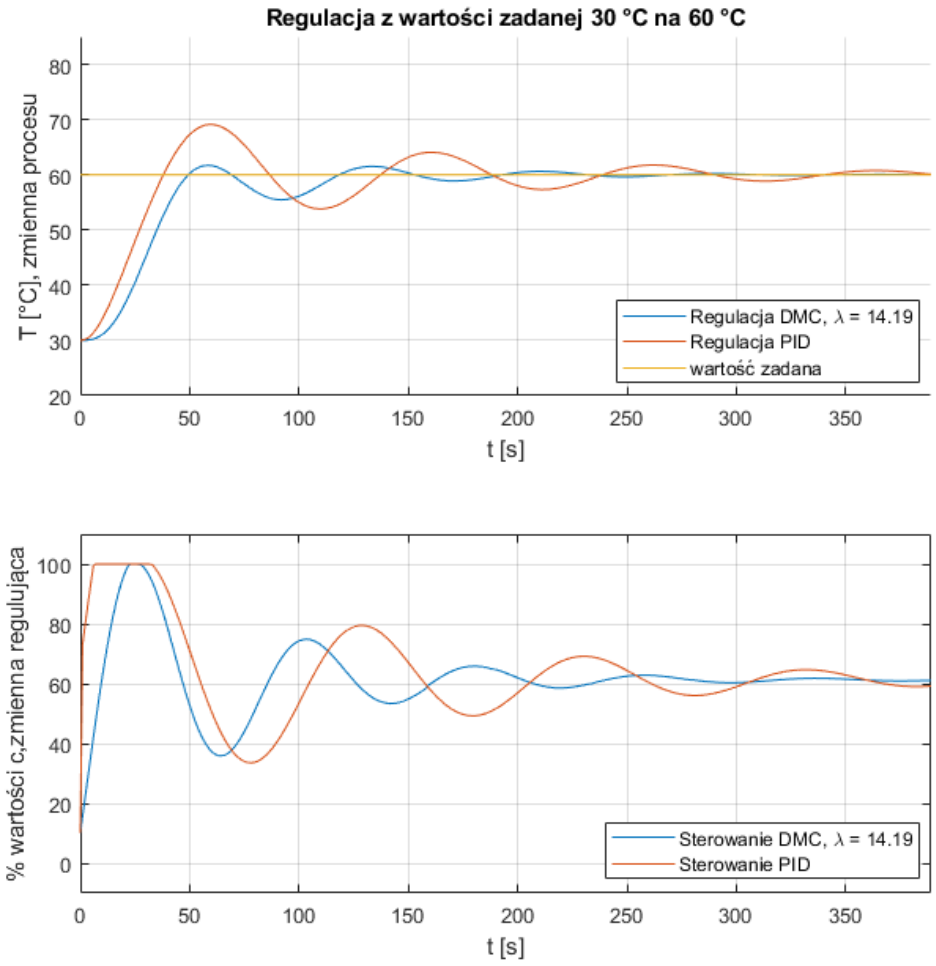
x	λ
0.1	0.4171
0.33	14.19
0.5	21.5
0.66	28.38
1.0	43



Rysunek 6.4: Regulacja w zależności od parametru λ.

6.3 Porównanie działania algorytmu DMC do konkurencyjnych rozwiązań

Do porównania zaproponowano popularny PID, tutaj PID Compact został zaimplementowany przez TIA Portal jako obiekt technologiczny. PID został nastrojony, wykorzystując wbudowaną funkcję *commissioning*, w której na podstawie przebiegów można przeprowadzić strojenie (ang. *tuning*). Posiadając oba regulatory nastrojone, można przedstawić porównanie ich działania na rysunku 6.5.



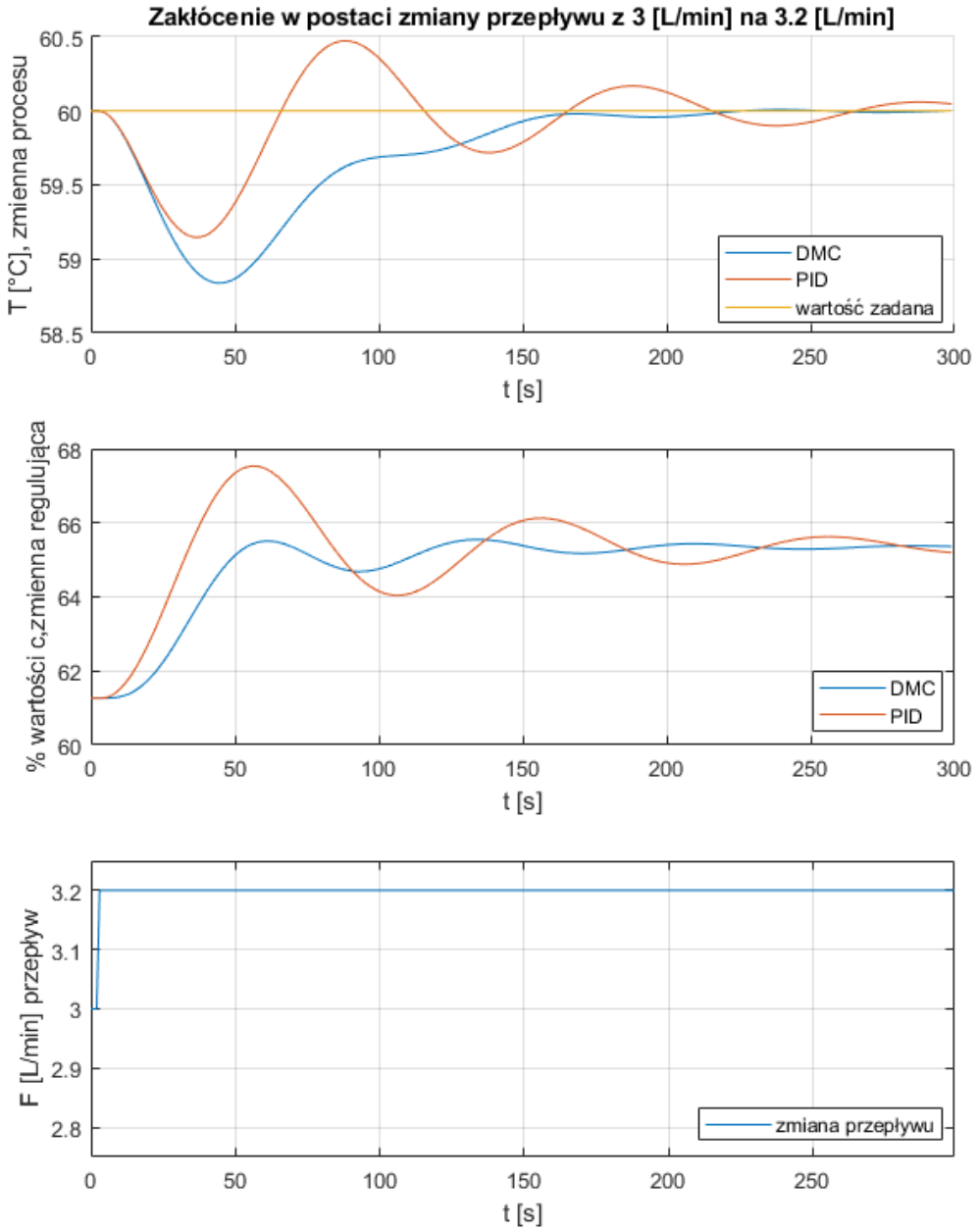
Rysunek 6.5: Regulacja w zależności od algorytmu.

Na porównaniu można zaobserwować, jak PID bardziej agresywnie zwiększa wartość zmiennej regulacji c, powodując szybsze osiągnięcie zadanej temperatury, niestety jednocześnie powoduje to, że dochodzi do przeregulowania, które występuje w regulacji DMC w znacznie mniejszym stopniu.

Obserwując algorytm DMC dla innych parametrów λ, jak na rysunku 6.4, mimo wyższego parametru λ i mniej korzystnej regulacji, wciąż nie dochodzi do takich przeregulowań ponad zadaną temperaturę, jak to ma miejsce w przypadku PID. Taka właściwość jest kluczowa w wielu procesach produkcyjnych, w szczególności cieplnych, aby nie uszkodzić obiektów bądź produktów w procesie.

Kolejnym ważnym aspektem są oscylacje wokół wartości zadanej, oscylacje PID są widocznie rzadsze, a jednocześnie bardziej odbiegające od wartości zadanej. Porównując przebiegi zmiennej procesowej, bardzo widoczne jest jak oscylacje ustają znacząco dla algorytmu DMC, w przeciwieństwie do PID, gdzie są one o wiele bardziej nieustające i utrzymują się dłużej.

W rzeczywistych przypadkach nie istnieją idealne zmienne wejściowe, co powoduje zakłócenia. Dla obiektu na którym działają regulatory, takie zakłócenia w procesie mogą występować jako fluktuacje temperatury otoczenia bądź różny przepływ wejściowy z sieci wodociągowej. W takich wypadkach regulator powinien prowadzić korekcję zmiennej regulującej, aby odpowiedziało to zakłóceniom.



Rysunek 6.6: Regulacja podczas zakłóceń.

Dla analizy wpływu zakłóceń widoczne jest, jak znacząco różnią się w działaniu porównywane algorytmy. Zakłócenia w PID powodują oscylacje, jak w przypadku wartości zadanej, występujące dla regulatora DMC, lecz w znacznie mniejszym stopniu. Odpowiedź DMC na zakłócenie oferuje o wiele mniejszą zmienność wartości regulującej, a co za tym idzie, mniej wzburzony powrót do wartości zadanej.

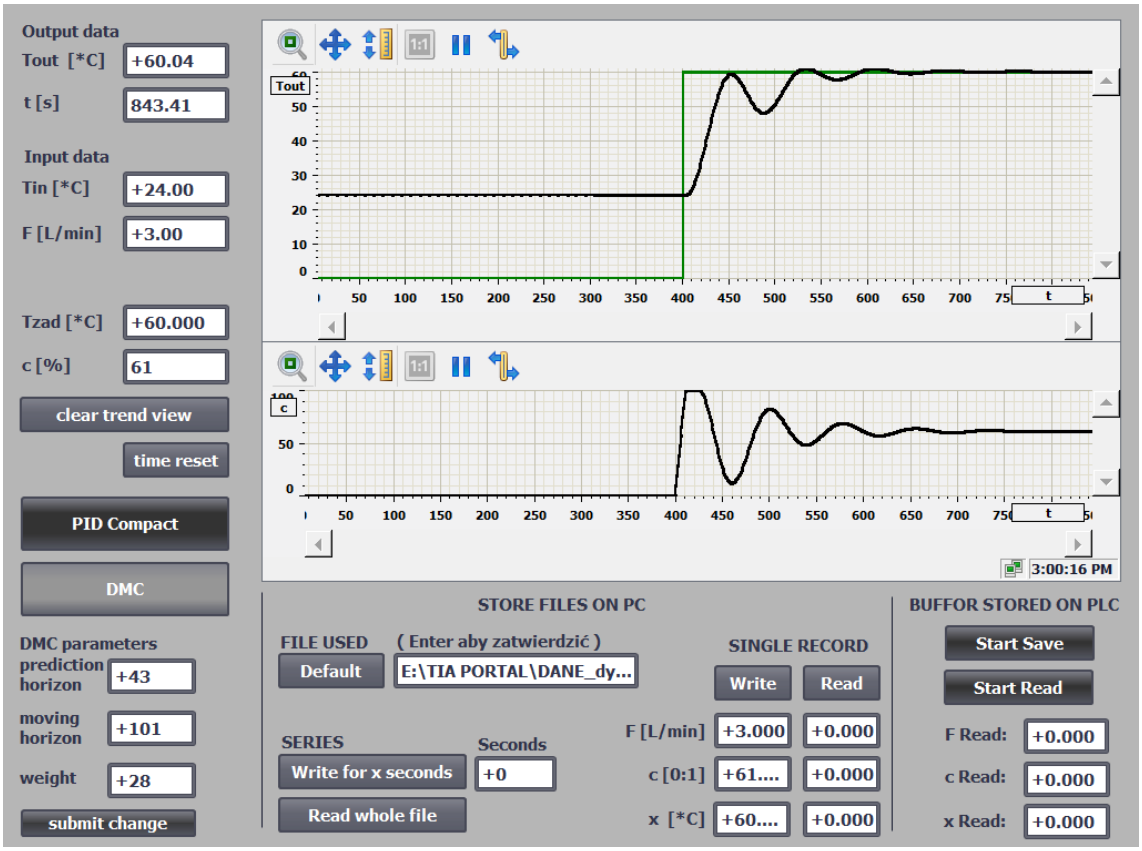
Rozdział 7

Obsługa i działanie

7.1 Interfejs użytkownika

Aby zapewnić możliwość obserwacji i wywierania wpływu na proces, została stworzona aplikacja, mająca za zadanie umożliwić wizualizację, zmiany parametrów obiektu, zmiany pomiędzy algorytmami regulacji, jak i zmiany parametrów regulatora DMC.

Interfejs użytkownika to HMI działające na zasadzie połączenia się ze sterownikiem za pośrednictwem sieci Profinet dzięki której następuje wymiana danych. HMI operuje na zmiennych Tag, które są cyklicznie synchronizowane ze sterownikiem.



Rysunek 7.1: Przedstawienie wizualizacji w WinCC.

Dolna część interfejsu oferuje również zapis i odczyt danych na dwa sposoby, pierwszy po stronie sterownika, a drugi w formie pliku na komputerze obsługującym WinCC.

7.2 Rozruch regulatora DMC

Aby uruchomić regulację na maszynie zewnętrznej, uruchamia się kod 7.2. Do jego uruchomienia wymagane jest posiadanie zainstalowanego interpretera języka Python 3 na maszynie oraz połączenie z siecią, w której znajduje się sterownik. Uruchomiony program nawiązuje łączność ze sterownikiem i prowadzi proces wymiany danych oraz regulacji obiektu.

```
if __name__ == "__main__":
    print("MAIN -> uruchamiam regulator DMC")
    D=D_struct(
        np.transpose(Step_response['y'])[0] # D.sr - unit step response data
        ,43 # D.p  - prediction horizon 20 to 70
        ,101 # D.m  - dynamic (moving) horizon
        ,28.38 # D.la - performance criterion weight
        ,0  # D.y  - current mrasurement
        ,[] # D.v  - past input, initially empty
        ,[]  # D.r  - reference (set point)
        ,[] # D.F  - matrix to calculate free response, set by the initial call
        ,[] # D.M  - dynamic matrix, set by the initial call
        ,[] # D.k  - DMC gain, set by the initial call
        ,None # D.u  - current input, initially 0
        )

    print("len(D.sr)")
    print(len(D.sr))
    print(D.sr)

    client = Client("opc.tcp://192.168.1.5:4840")
    try:
        client.connect()

        # Client has a few methods to get proxy to UA nodes that should always be in address space such as Root or Objects
        root = client.get_root_node()
        print("Objects node is: ", root)

        # Node objects have methods to read and write node attributes as well as browse or populate address space
        print("Children of root are: ", root.get_children())

        s.enter(0, 1, dmc_cycle,  (s,client,D,))
        s.run()

    finally:
        client.disconnect()
```

Rysunek 7.2: Uruchomienie regulatora DMC.

Rozdział 8

Biblioteki i moduły użyte w języku Python

8.1 Sys

Moduł zapewniający dostęp do obiektów używanych lub utrzymywanych przez interpreter oraz do funkcji, które współdziałają z interpreterem. Używany do wyświetlania informacji w konsoli jak i operacji na danych.

8.2 NumPy

NumPy (ang. *Numerical Python*) to biblioteka stworzona, aby umożliwić obliczenia numeryczne. Największą część blibioteki obejmują działania na tablicach i macierzach. Zawiera również funkcje potrzebne do algebry liniowej oraz transformat.

8.3 SciPy

SciPy (ang. *Science Python*) to biblioteka używana do obliczeń naukowych oraz technicznych. Zawiera ona moduły do działań w dziedzinie algebry liniowej, interpolacji oraz przetwarzania macierzy.

8.4 Opcua

Opcua to biblioteka implementująca funkcjonalności OPC UA dla programów napisanych w języku Python. Oferuje ona interfejs zarówno niskopoziomowy do wysyłania i odbierania struktur, jak i wysokopoziomowy do pisania serwerów oraz klientów.

8.5 Sched oraz time

Sched działa jako moduł do programowania zajęć w harmonogramie, dla której przydatnym modułem jest moduł time implementujący funkcjonalności do operowania na danych w formie dat oraz czasu.

Rozdział 9

Podsumowanie i wnioski

Rozwój technologii pozwala na wdrażanie zaawansowanych algorytmów regulacji. Częścią przemysłu 4.0 jest komunikacja pomiędzy urządzeniami w ramach wymiany danych. W myśl tej idei implementacja algorytmów na zewnątrz, połączonych poprzez sieć, oferuje możliwość elastycznego rozwoju zakładów przemysłowych. *Edge computing* oferuje idealne rozwiązanie dla zaawansowanych algorytmów regulacji, działając w ramach tej samej sieci co sterownik. Dzięki takim rozwiązaniom sterowanie procesami zyskuje na jakości, jednocześnie odciążając wymagania obliczeniowe sterownika.

Jak można zaobserwować w sekcji 6.3, zaawansowane algorytmy regulacji jak DMC, który należy do grupy MPC, oferują znaczące zyski w kontroli procesu. Implementacje z wykorzystaniem komunikacji sieciowej oferują znacząco większe zasoby obliczeniowe do wykorzystania i wraz z rozwojem technologii będą stawać się coraz częściej implementowanym rozwiązaniem.

Rozwój

- Rozwinięcie algorytmu o automatyczne strojenie.
- Wprowadzenie redundancji oraz komunikacji pomiędzy obiektami regulującymi.
- Implementacja innych algorytmów regulacji.

Bibliografia

[1] Tomasz Klopot, Piotr Skupin, Mieczysław Metzger, Patryk Grelewicz. Tuning strategy for dynamic matrix control with reduced horizons. *Elsevier*, 76:145–154, 2017.

[2] Piotr Skupin. *Materiały dydaktyczne dla przedmiotu „DYNAMIKA PROCESÓW", ćw. Obiekt cieplny, kierunek Automatyka i Robotyka.*

[3] Piotr Tatjewski. *Sterowanie zaawansowane obiektów przemysłowych.* EXIT, Warszawa, 2002.

Dodatki

Spis skrótów i symboli

Skróty

DMC (ang. *Dynamic Matrix Control*)

FOPDT model pierwszego rzędu z opóźnieniem (ang. *first-order plus deadtime*)

IoT Internet rzeczy (ang. *Internet of Things*)

MPC algorytm regulacji predykcyjnej (ang. *Model Predictive Control Model-based Predictive Control*)

OPC UA Zunifikowana architektura komunikacji otwartej platformy (ang. *Open Platform Communications Unified Architecture*)

PID regulator proporcjonalno-całkująco-różniczkujący (ang. *proportional–integral–derivative controller*)

PLC Programowalny sterownik logiczny (ang. *Programmable Logic Controller*)

SISO Pojedyncze wejście pojedyncze wyjście (ang. *Single Input Single Output*)

SCL (ang. *Structured text*)

Symbole

F przepływ

Q ilość ciepła zakumulowanego w zbiorniku [J]

V objętość

P_{max} maksymalna moc grzałki

c wypełnienie przebiegu periodycznych zmian mocy grzałki

K transmitancja

T_{ot} Temperatura otoczenia

T_{out} Temperatura na wyjściu układu

T_{in} Temperatura na wejściu układu

T_{zad} Temperatura zadana na regulator

c_w ciepło właściwe

t czas

Dt krok obserwacji

h krok całkowania

NIS liczba kroków h w kroku Dt

SISO Single input single output – system z pojedynczym wejściem pojedynczym wyjściem

regulacja DMC

T_C (ang. *Sampling Time*)

H_C (ang. *Control Horizon*)

H_P (ang. *Prediction Horizon*)

H_D (ang. *Dynamic Horizon*)

λ (ang. *Supression Coefficient*)

Kod źródłowy

Kod algorytmu DMC zaimplementowanego w języku Python

```
1  def dmc(D):
2
3      #size of step response
4      #ilosc danych w step response ukladu
5      N=np.size(D.sr)
6
7      # predition horizon parameter
8      # parametr horyzontu przewidywania
9      Dc=D.p
10
11     # first run definition
12     # definicje pierwszego uruchomienia
13     if not D.v:
14         # number of past inputs to keep
15         # liczba poprzednich wej
16         n=N-Dc
17         # storage for past input
18         # wektor poprzednich wej
19         D.v=np.zeros(n)
20         # matrix to calculate free response from past input
21         # macierz do liczenia skladowej swobodnej z
               poprzednich wej
22         x=D.sr[0:n]
```

```
23          # tutaj ze wzoru (10) Ku
24          D.F=hankel(D.sr[1:Dc+1],D.sr[Dc:N])-np.tile(x,[Dc
                ,1])
25
26          # dynamic matrix
27          # macierz dynamiczna M
28          first_row = D.sr[1]*np.eye(1,D.m)
29          D.M=linalg.toeplitz(D.sr[0:Dc],first_row)
30
31          # calculate DMC gain
32          # liczenie wzmocnienia DMC
33          #print(D.la*np.eye(D.m))
34          R=np.linalg.cholesky(np.dot(np.transpose(D.M),D.M)
                + D.la*np.eye(D.m))
35          # print(R) z jakiego  powodu wychodzi tranponowane
                R
36          R=np.transpose(R)
37          rightRnaDG = np.linalg.lstsq(np.transpose(R),np.
                transpose(D.M),rcond=None)
38          #macierz wsp  czynnik w K jest obliczana raz w
                fazie obliczania regulatora
39          K=np.linalg.lstsq(R,rightRnaDG[0],rcond=None)
40
41          # only the first input will be used
42          # tylko pierwsze u ywane
43          #D.k=K[1,:]
44          #tutaj ze wzoru (10) Ke
45          D.k=K[0][0]
46          D.u=0
47      #end of first run definition
48      #koniec definicji pierwszego uruchomienia
49
50      # free response
```

```
51      # skadowa swobodna
52      f=np.dot(D.F,D.v)
53
54      # reference sp
55      #referencja warto ci zadanej
56      nr=np.size(D.r)
57      if nr>=Dc:
58          ref=D.r[0:Dc]
59      else:
60          ref=[D.r]
61          ref.extend(D.r[-1]+np.zeros(Dc-nr))
62
63      ref[0]=D.y
64
65      # DMC input change
66      # obliczenie nowego du z
67      du=np.dot(D.k,(ref-f-D.y))
68
69      # past input change for next step
70      # przesuni cie wektora poprzednich wej
71      prevDvWOlast = D.v[:-1]
72      prevDvWOlast=np.append(du,prevDvWOlast)
73      D.v = prevDvWOlast.tolist()
74
75      # next input
76      # nast pne u
77      D.u=D.u+du
78
79      # so that the output is in <0,1>
80      # ograniczenie parametr w do przedzia u <0,1>
81      # anti-wind-up
82      if D.u > 1 :
83          D.u = 1
```

```
84      if D.u < 0 :
85          D.u = 0
86
87      return D
```

Lista dodatkowych plików, uzupełniających tekst pracy

Do pracy dołączono dodatkowe pliki zawierające:

- projekt programu sterownika PLC z TIA Portal,
- kod programu do cyklicznego liczenia algorytmu DMC oraz łączności w języku Python

Spis rysunków

Spis tablic