



POLITECHNIKA ŚLĄSKA
WYDZIAŁ AUTOMATYKI, ELEKTRONIKI I
INFORMATYKI

Project Based Learning

*Sterowanie predykcyjne dla wybranych układów automatyki
procesowej*

Skład zespołu:

Adamczyk Mateusz
Hołda Kacper
Porwoł Adam
Śliwa Daniel
Trzęsimiech Oskar

St. II, sem. 2

Gliwice, październik 2021 – luty 2022

Spis treści

Wstęp	3
Etapy projektu oraz ich wykonanie	4
Zebranie danych badawczych z instalacji cieplnej.....	4
Implementacja prostego, lokalnego układu regulacji.....	11
Komunikacja.....	17
Konfiguracja sieci	17
Kumunikacja S7 protocol	18
Przegląd i wybór narzędzi i środowisk do własnej implementacji algorytmu MPC	19
Implementacja wybranych algorytmów z rodziny MPC w środowisku MATLAB	19
Implementacja algorytmu MPC na wybranych platformach	20
Rzeczywiste testy regulacji.....	20
Bioreaktor	23
Opis instalacji.....	23
Komunikacja OPC-UA	24
Zebranie danych badawczych z instalacji bioreaktora	26
Podsumowanie projektu	28

Wstęp

W ramach projektu zostanie zrealizowane sterowanie predykcyjne dla procesu dystrybucji ciepła. Obiektem regulacji jest przepływowy podgrzewacz ciepła. Obiekt ten charakteryzuje się silnymi nieliniowościami. Tradycyjne sterowanie oparte o regulator PID daje akceptowalną jakość regulacji tylko dla niewielkich odchyłek od punktu pracy. Poprawa jakości regulacji możliwa jest dzięki zastosowaniu zaawansowanych regulatorów takich jak regulatory z rodziny MPC (ang. Model Predictive Control). W celu podniesienia wartości merytorycznej projektu planowana jest także równoległa weryfikacja zaproponowanych rozwiązań dla reaktora biologicznego z osadem czynnym. Natomiast głównym celem projektu jest stworzenie rozproszonego systemu sterowania opartego o regulator predykcyjny z zapewnieniem mechanizmu bez uderzeniowego przełączania pomiędzy algorytmami.

Etapy projektu oraz ich wykonanie

W tym wątku raportu zostaną przedstawione najważniejsze etapy prac, wraz z przykładami, które zostały wykonane na przestrzeni realizacji całego projektu PBL.

Zebranie danych badawczych z instalacji cieplnej

Proces zbierania danych na instalacji cieplnej, dostępnej w sali 338a, przebiega w sposób następujący:

Na początku zapoznano się z całą instalacją i wybrano dogodną jej konfigurację w postaci sterowania przepływem medium roboczego, przepływającego przez grzałkę, której wartości mocy także można było zadawać. Dodatkowymi danymi, które mogły być zdczytywane z instalacji to Wartość przepływu, temperatura medium na wejściu grzałki oraz temperatura na wyjściu.

Po zapoznaniu się z całym działaniem instalacji nastąpiła implementacja systemu sterowania na zakupionym sterowniku Siemens. Dzięki wykonanym czynnościom można było przystąpić do zbierania danych z instalacji, których surową formę przedstawiono w tabeli numer 1.

W celu zebrania danych niezbędnych do stworzenia modelu badanego układu wykonano serie pomiarów z różnymi wartościami parametrów wejściowych. W przypadku badanego obiektu parametrami wejściowymi układu są:

- Przepływ cieczy [l/m]
- Temperatura na wyjściu grzałki [°C]
- Moc grzałki [%]

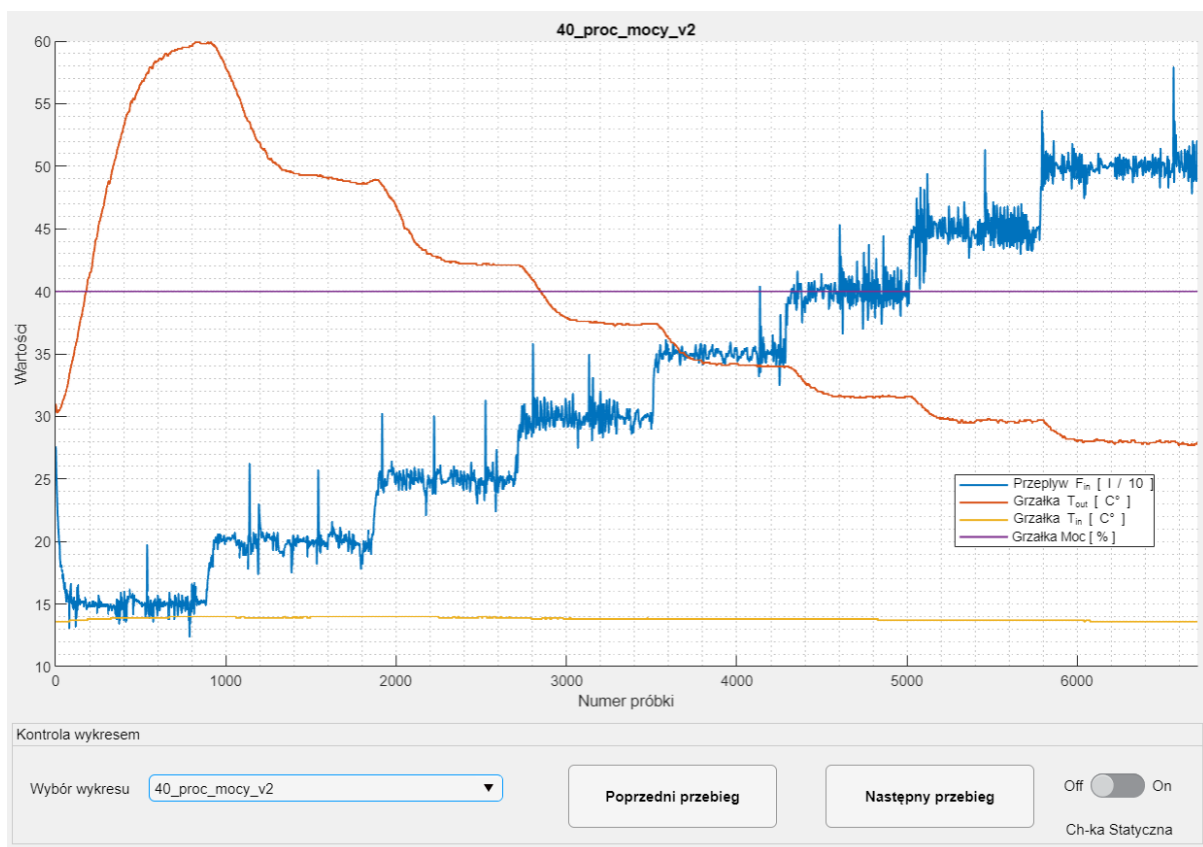
Iteration_#	"Heater_Flow_In[%]"	"Heater_Tin_In[%]"	"Heater_Tout_In[%]"	"DB.Heater_PWR"
2,	276,	136,	310,	40
3 ,	273,	136,	309,	40
4 ,	267,	136,	308,	40
5 ,	260,	136,	307,	40
6 ,	253,	136,	306,	40
7 ,	250,	136,	304,	40
8 ,	245,	136,	304,	40
9 ,	240,	136,	303,	40
10 ,	236,	136,	303,	40
11 ,	231,	136,	303,	40
12 ,	226,	136,	303,	40
13 ,	222,	136,	303,	40
14 ,	220,	136,	303,	40
15 ,	218,	136,	304,	40

16 ,	215,	136,	304,	40
17 ,	211,	136,	304,	40
18 ,	209,	136,	304,	40
19 ,	207,	136,	305,	40
20 ,	205,	136,	305,	40
21 ,	202,	136,	305,	40
22 ,	199,	136,	305,	40
23 ,	198,	136,	305,	40
24 ,	199,	136,	305,	40
25 ,	198,	136,	305,	40
26 ,	193,	136,	305,	40
27 ,	187,	136,	305,	40
28 ,	185,	136,	306,	40
29 ,	184,	136,	306,	40
30 ,	182,	136,	307,	40
31 ,	180,	136,	307,	40
32 ,	180,	136,	307,	40

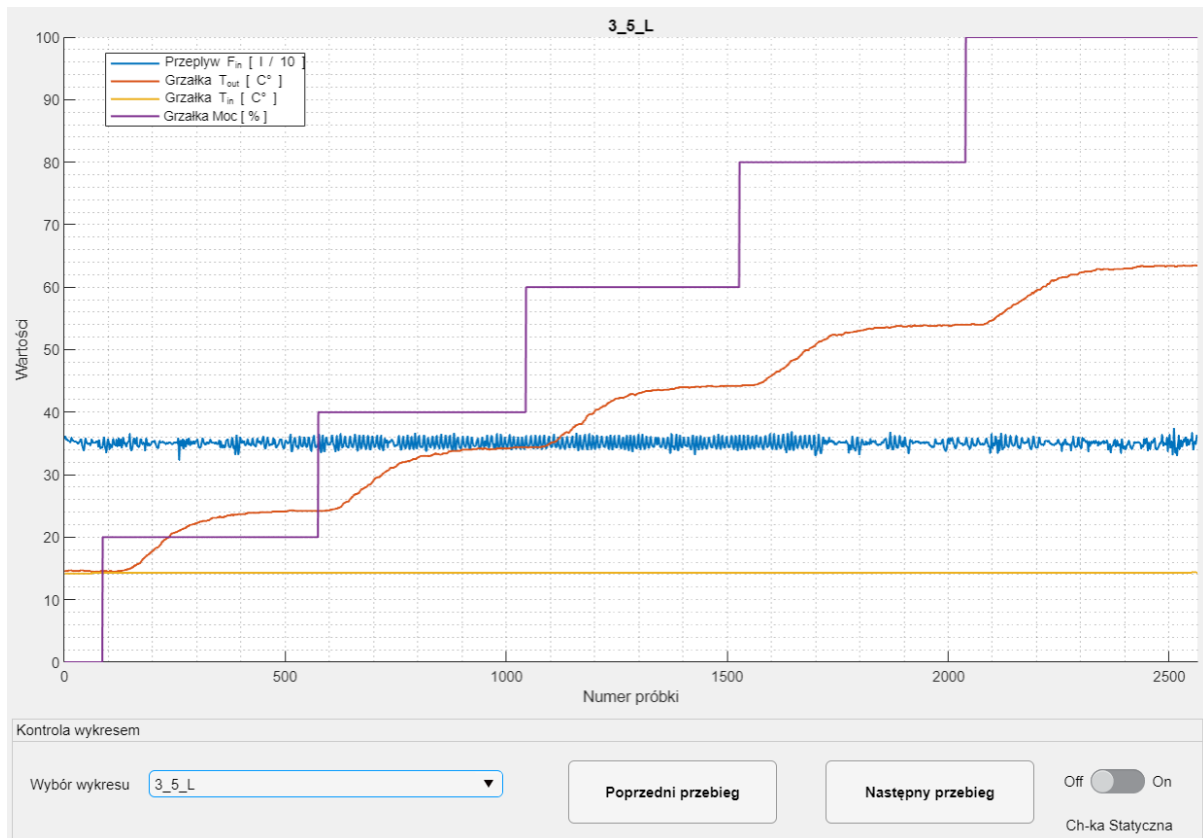
Tabela 1. Wycinek z pomiarów zebranych dla mocy grzałki = 40%

Następnie stworzono aplikację, która w przejrzysty sposób pokazywałaby wszystkie zgromadzone dane i umożliwia łatwą interpretację wyników. Interfejs tej aplikacji wraz z przykładowymi pomiarami przedstawiono na rysunkach numer 2, 3, 4 oraz 5.

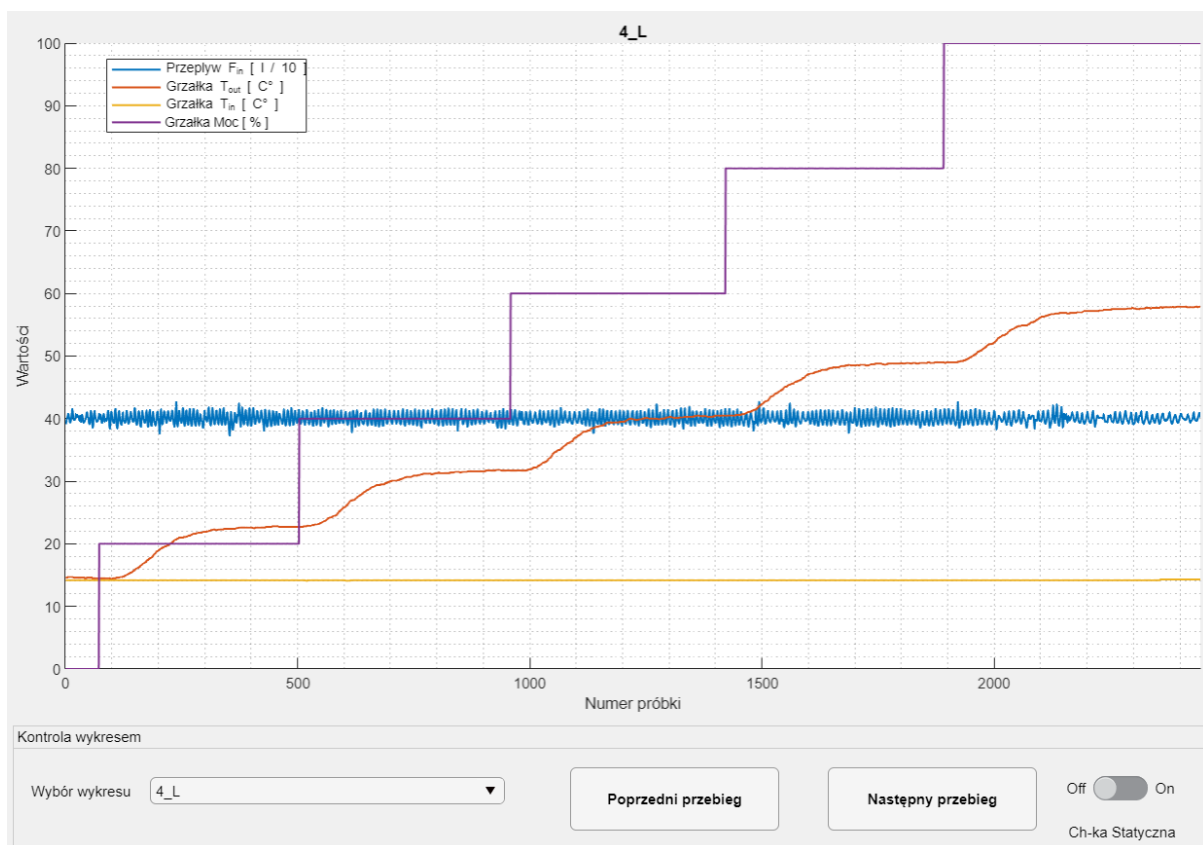
Ponadto warto zaobserwować, że na rysunku numer 3, 4 oraz 5 widnieją pomiary dla pierwszego rodzaju zbierania danych, które zostały przez nas wykonane, 3 dni wtedy, gdy przepływ z instalacji był stały (dla wszystkich pomiarów w zakresie od pół litra do 5 litrów, co pół litra), a zmieniano tylko moc grzałki. Natomiast druga seria pomiarów została wykonana dla stałych mocy grzałki w zakresie od 20% do 100% - jak pokazano na przykładowym przebiegu na rysunku numer 2.



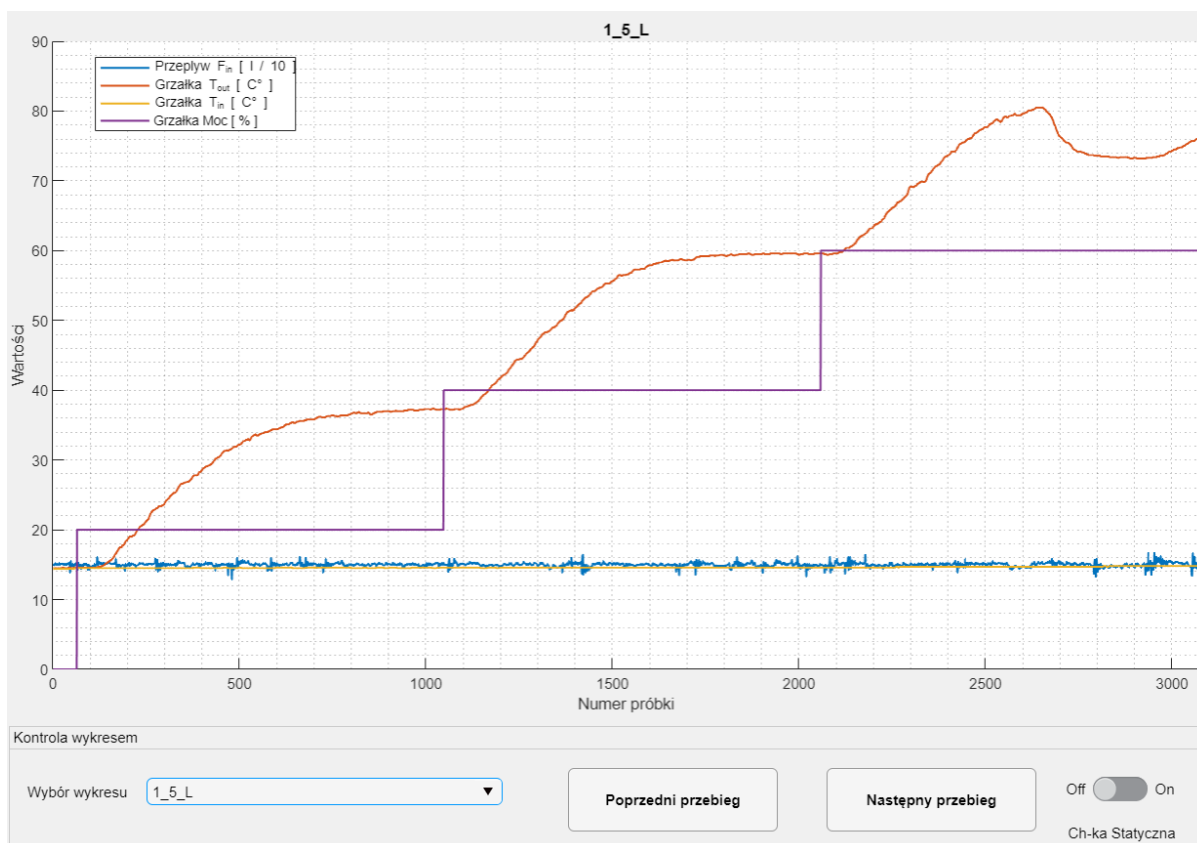
Rysunek 1. Przykładowe pomiary dla stałego poziomu mocy grzałki



Rysunek 2. Przykładowe pomiary dla stałego poziomu przepływu dla 3,5l



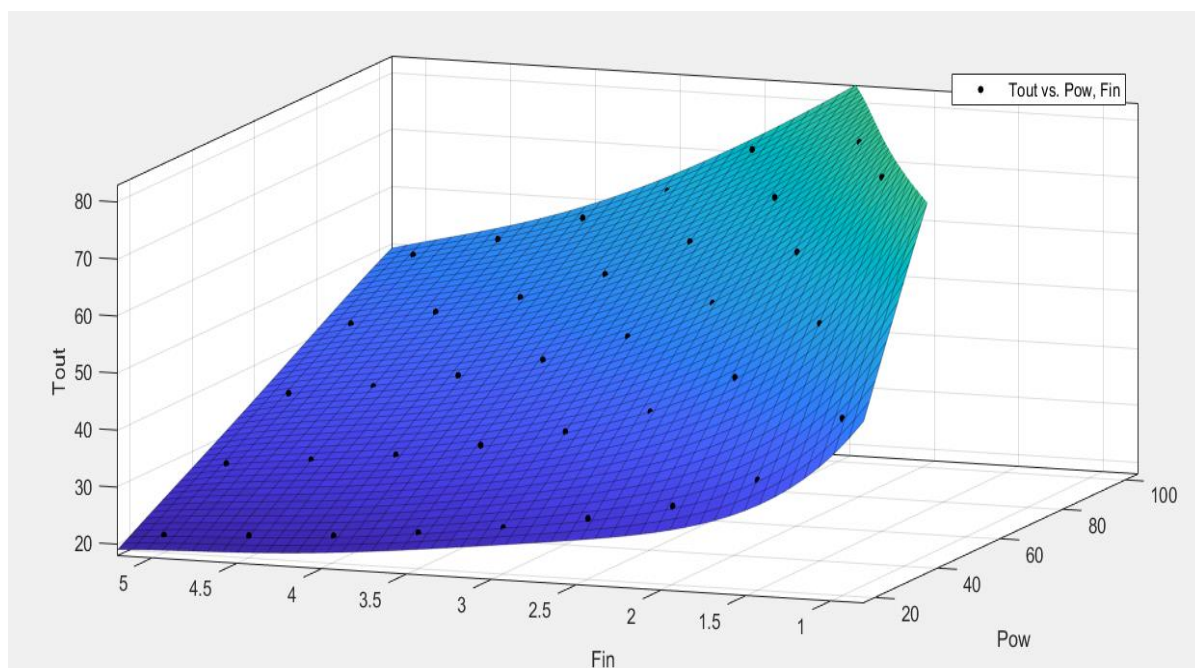
Rysunek 3. Przykładowe pomiary dla stałego poziomu przepływu dla 4l



Rysunek 4. Przykładowe pomiary dla stałego poziomu przepływu dla 1,5l z wystąpieniem alarmu temperaturowego

Dodatkowo, na rysunku numer 5 można zaobserwować, niespotykane przy większości pomiarów z występowaniem większej wartości przepływu, zachowanie. Jest ono spowodowane zbyt małym przepływem medium roboczego, tak aby uzyskać maksymalną temperaturę grzałki poniżej 80 stopni Celsjusza. Powyżej tej temperatury następuje włączenie alarmu temperaturowego, który realizuje proste wyłączenie grzałki.

Z tak zebranych pomiarów wyznaczono charakterystykę statyczną, którą aproksymacji przedstawiono na rysunku numer 6.



Rysunek 5. Charakterystyka statyczna dla instalacji cieplnej

W ten też sposób stworzono model łączący wszystkie punkty widoczne na charakterystyce. Jego model, w surowych danych, przedstawiono w tabeli, z kodem, numer 2.

Linear model Poly45:

$$f(x,y) = p_{00} + p_{10}x + p_{01}y + p_{20}x^2 + p_{11}xy + p_{02}y^2 + p_{30}x^3 + p_{21}x^2y + p_{12}xy^2 + p_{03}y^3 + p_{40}x^4 + p_{31}x^3y + p_{22}x^2y^2 + p_{13}xy^3 + p_{04}y^4 + p_{41}x^4y + p_{32}x^3y^2 + p_{23}x^2y^3 + p_{14}xy^4 + p_{05}y^5$$

where x is normalized by mean 55.14 and std 27.65

and where y is normalized by mean 3.338 and std 1.167

Coefficients (with 95% confidence bounds):

p00 =	43.08	(42.83, 43.32)
p10 =	14.39	(13.99, 14.78)
p01 =	-10.46	(-10.88, -10.04)
p20 =	-0.001136	(-0.5633, 0.561)
p11 =	-5.311	(-5.835, -4.788)
p02 =	3.882	(3.403, 4.361)
p30 =	-0.03527	(-0.2889, 0.2183)
p21 =	0.1468	(-0.5499, 0.8435)
p12 =	1.766	(1.167, 2.365)
p03 =	-0.4751	(-0.978, 0.02783)
p40 =	-0.01274	(-0.2623, 0.2368)
p31 =	0.4081	(0.1335, 0.6826)
p22 =	-0.7726	(-1.072, -0.4734)
p13 =	-0.08012	(-0.3391, 0.1788)
p04 =	0.1994	(0.001938, 0.3968)
p41 =	-0.05466	(-0.3443, 0.235)
p32 =	-0.2397	(-0.5434, 0.06401)
p23 =	0.5005	(0.2163, 0.7847)
p14 =	-0.08529	(-0.3196, 0.149)
p05 =	-0.3165	(-0.4821, -0.1509)

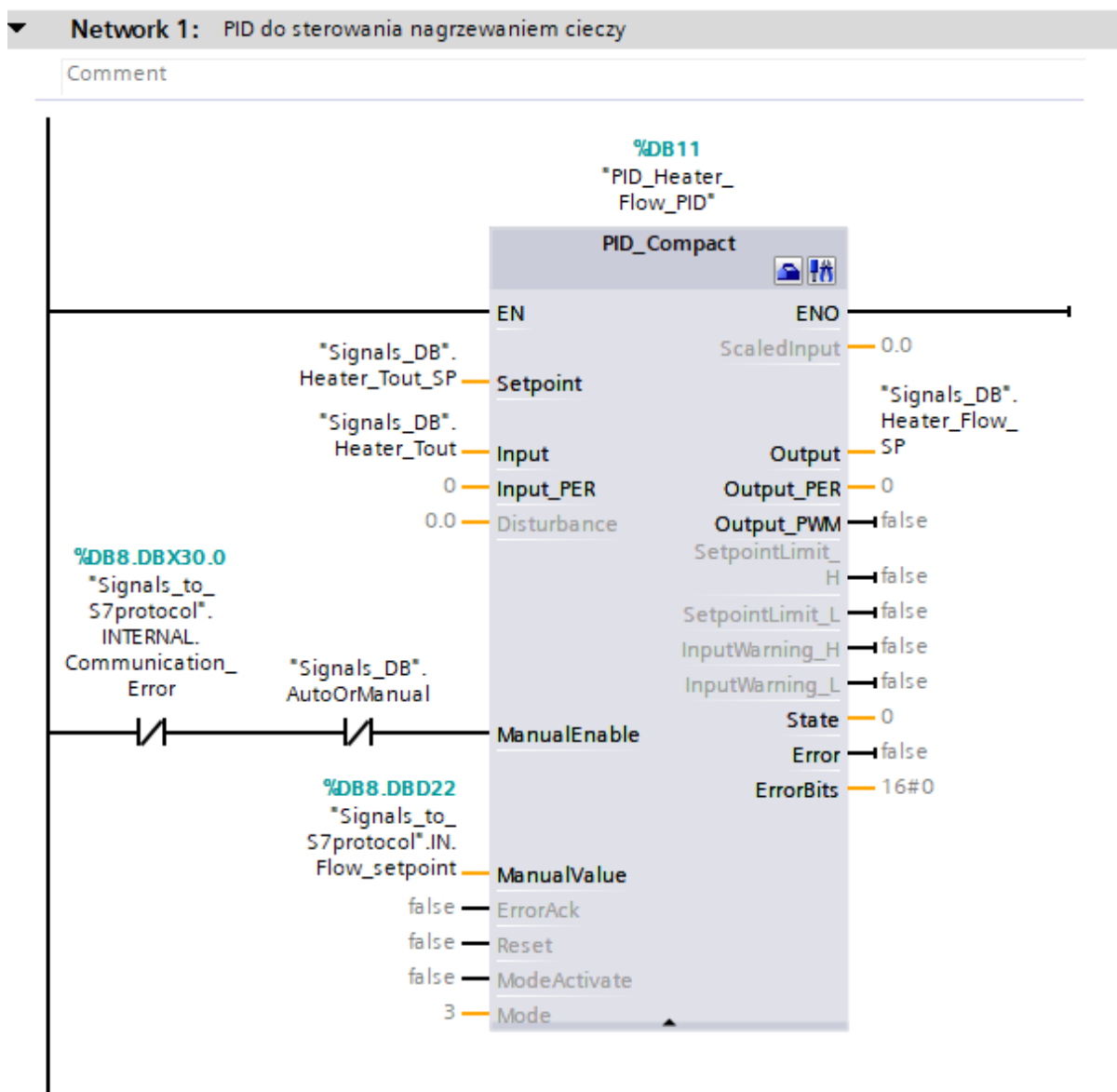
Tabela 2. Zdefiniowana funkcja stworzonego modelu

Co więcej poniżej przedstawiono poziom dopasowania wedle wybranych wskaźników:

- SSE: 0.7832
- R-square: 0.9999
- Adjusted R-square: 0.9998
- RMSE: 0.2146

Implementacja prostego, lokalnego układu regulacji

Z wykorzystaniem środowiska programistycznego TIA Portal został zaimplementowany lokalny układ regulacji. Wykorzystano do tego celu bloczek PID_Compact, który umieszczono w cyklicznym bloku danych OB. W konfiguracji bloczka ustawiono parametr „Cyclic time” na wartość 100 co oznacza, że program w bloczku będzie wykonywany co 100 ms. Całość implementacji wykonano w języku LAD. Poniżej znajduje się zrzut ekranu obrazujący bloczek PID_Compact wraz z podpiętymi sygnałami wejściowymi oraz wyjściowymi.



Rysunek 6 Bloczek PID odpowiedzialny za kontrolę procesu nagrzewania cieczy

Ze względu na specyfikę obiektu zdecydowano się na wykorzystanie odwróconej logiki. Natomiast tryb działania regulatora ustawiono na automatyczny

Controller type

Flow l/min

☒ Invert control logic

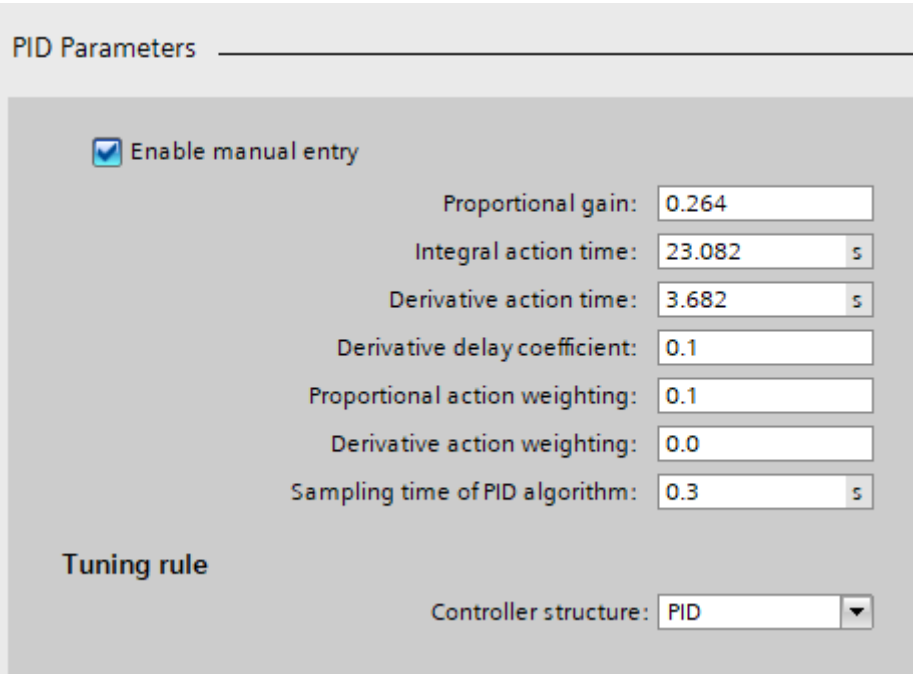
☒ Activate Mode after CPU restart

Set Mode to: Automatic mode

Rysunek 7 Konfiguracja regulatora PID - typ regulatora

Należy również wspomnieć, że korzystaliśmy z wejść oraz wyjść uprzednio przeskalowanych. Całość procesu sygnałów analogowych przeprowadzono w bloku OB1 „Main”. W konfiguracji bloczka PID postanowiono również ograniczyć wartość sygnału wyjściowego, aby uniknąć nadmiernego narastania wartości wyjściowej (setpoint przepływu). Po przeprowadzeniu testów zdecydowano przyjąć górny limit wartości wejściowej jako 15, ponieważ maksymalna wartość przepływu uzyskana na instalacji wynosiła w pikie nieco ponad 14 litrów. Taki zabieg miał na celu ograniczenie narastania setpointu do bardzo dużych wartości.

Następnie przystąpiono do nastrojenia regulatora PID. Poniżej zostały zaprezentowane aktualne nastawy regulatora wykorzystywane do przeprowadzenia testów.



PID Parameters

☒ Enable manual entry

Proportional gain: 0.264

Integral action time: 23.082 s

Derivative action time: 3.682 s

Derivative delay coefficient: 0.1

Proportional action weighting: 0.1

Derivative action weighting: 0.0

Sampling time of PID algorithm: 0.3 s

Tuning rule

Controller structure: PID

Rysunek 8 Nastawy regulatora PID

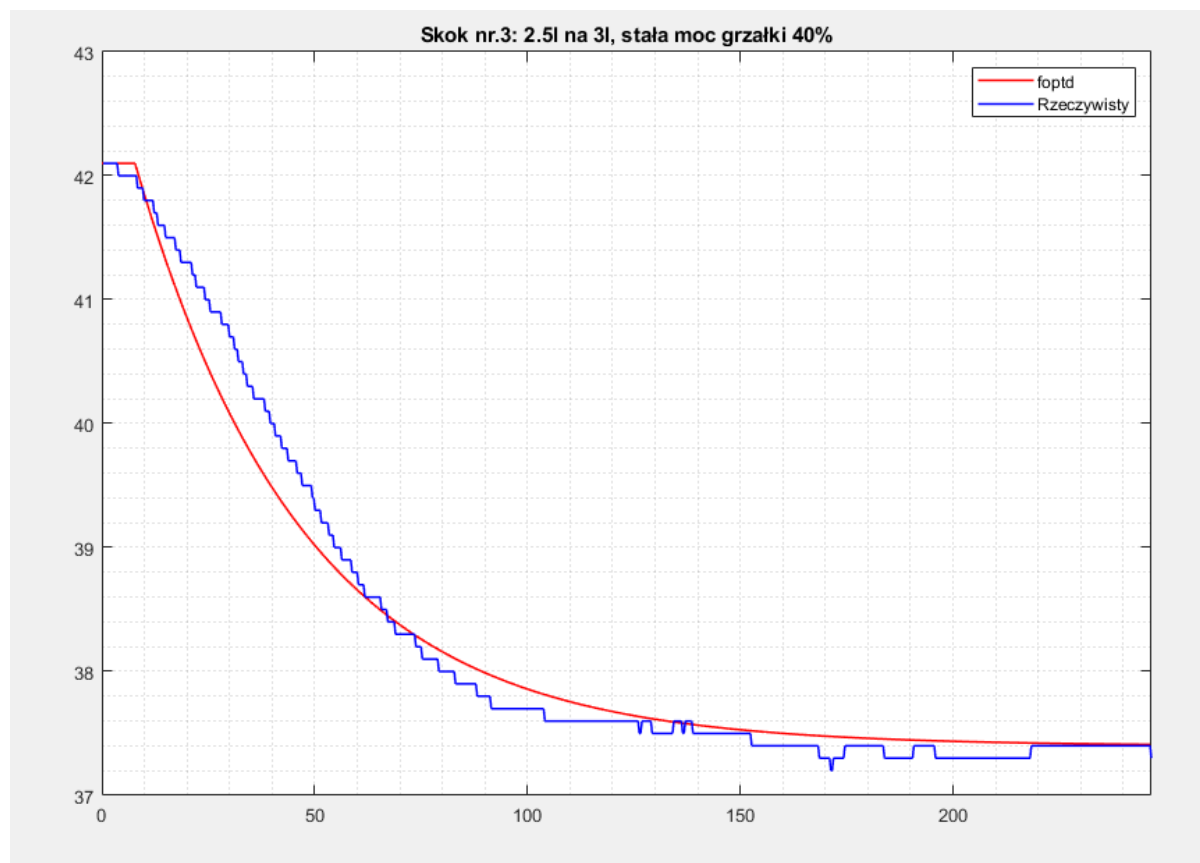
Wyznaczając nastawy regulatora PID należało przeprowadzić eksperyment polegający na ustawieniu stałej mocy grzałki a następnie dokonywać skokowej zmiany przepływu w momencie osiągnięcia przez temperaturę wyjściową stanu ustalonego. Szereg takich pomiarów opisanych w punkcie „Zebranie danych badawczych z instalacji cieplnej” umożliwił dobranie nastaw regulatora PID. Następnie analizując poszczególne skoki przepływu oraz przebiegi wartości wyjściowej (temperatury wyjściowej) wykorzystano metodę 2 punktów do wyznaczenia stałej czasowej oraz czasu opóźnienia. Parametry te posłużyły do skonstruowania modelu FOPDT. Następnie bazując na najlepszym dopasowaniu modelu do rzeczywistych przebiegów temperatury wyjściowej wybrano zestaw nastaw regulatora PID.

Do wyznaczenia nastaw posłużyła metoda AMIGO. Poniżej została zaprezentowana tabela przedstawiająca obliczone nastawy regulatora

l.p.	kp	t1	t2	tał p	delta p	Kp	Ti	Td
1.	24	49,8	90,3	-60,75	-10,95	0,112357	34,07524	5,194132
2.	13,8	36,3	64,5	-42,3	-6	0,244384	21,25513	2,877551
3.	9,4	31,8	58,2	-39,6	-7,8	0,264321	23,08163	3,682403
4.	6,6	22,2	50,1	-41,85	-19,65	0,175515	34,08143	8,611923
5.	4,8	21,6	47,7	-39,15	-17,55	0,250801	31,34717	7,734802
6.	3,8	17,4	30,9	-20,25	-2,85	0,894044	10,13723	1,367271
7.	3,8	12,9	33,3	-30,6	-17,7	0,25736	26,90809	7,541353

Tabela 3 Parametry FOPDT oraz nastawy regulatora PID

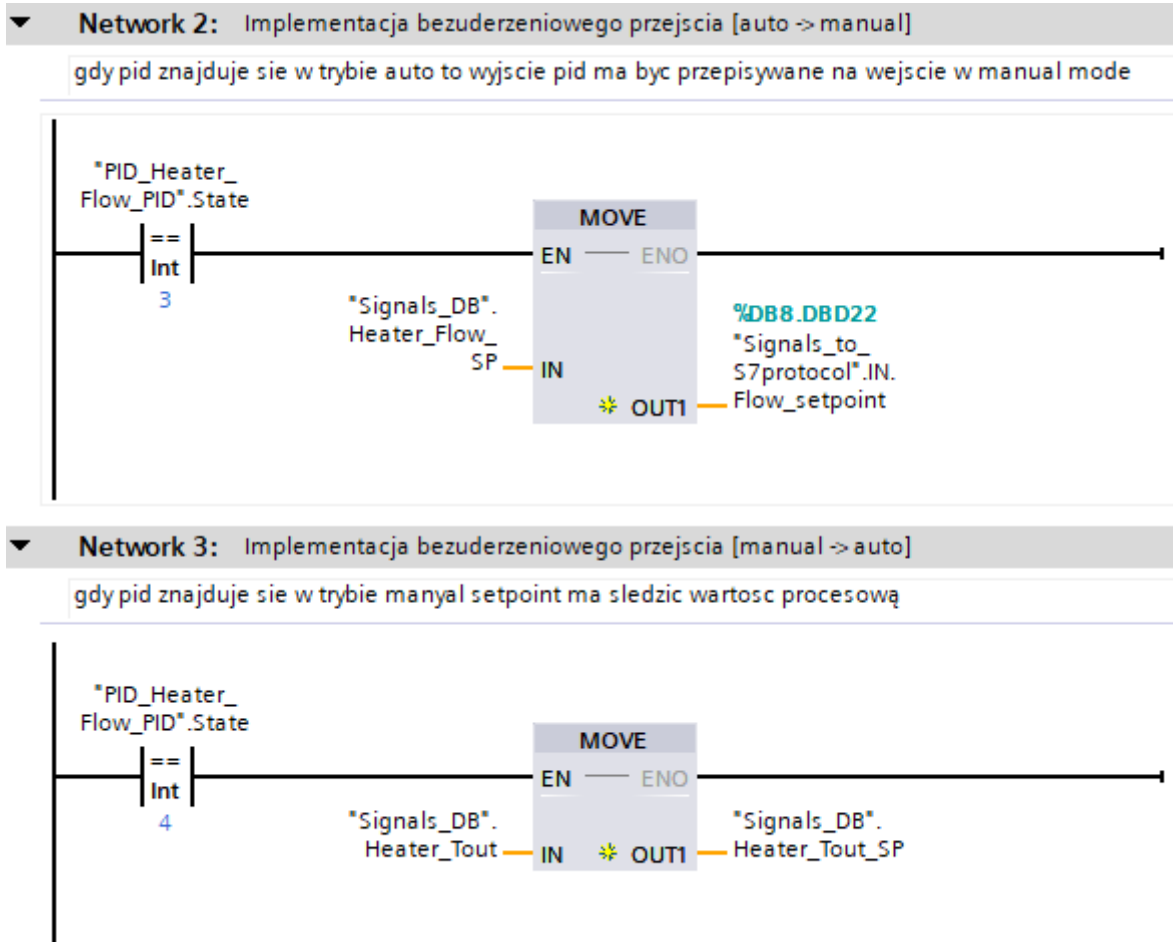
Przeprowadzając testy dla modelu FOPDT wykorzystano moduł stałej czasowej oraz czasu opóźnienia. Kolorem zielonym wskazano nastawy dla których uzyskano najlepsze dopasowanie modelu FOPDT do rzeczywistych przebiegów.



Rysunek 9 Porównanie przebiegu odpowiedzi rzeczywistej i fopdt.

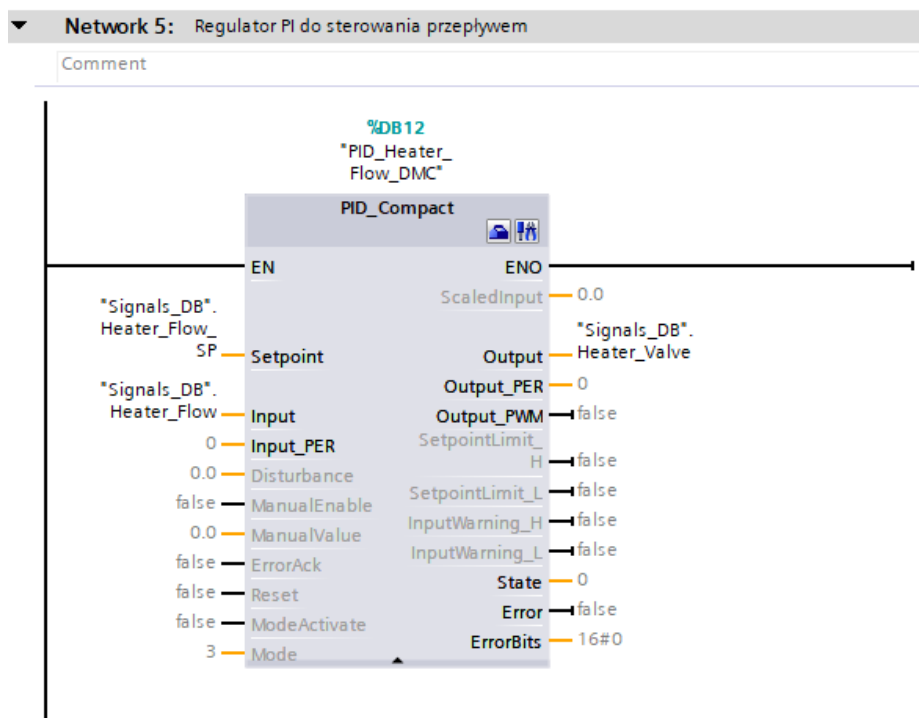
Dokonano również implementacji bezuderzeniowego przełączania pomiędzy regulacją lokalną oraz zdalną. Wykorzystano do tego celu wejścia bločku PID_Compact umożliwiające wprowadzenie regulatora w tryb regulacji manualnej. Oprócz tego stworzono 2 dodatkowe networki umożliwiające przepisywanie wyjścia regulatora na wartość setpointu regulacji

zdalnej oraz śledzenie przez setpoint bloczka PID wartości procesowej odpowiednio w trybie automatycznym oraz w trybie manualnym. Rozwiązanie zostało zaprezentowane poniżej.



Rysunek 10 Implementacja bezuderzeniowego przełączania pomiędzy lokalnym oraz zdalnym układem regulacji

Należy również wspomnieć o sposobie regulacji przepływu. Tutaj również wykorzystano bloczek PID_Compact. Jednak regulator wykorzystany w tym zagadnieniu różni się od prezentowanego powyżej brakiem części różniczkującej. Zastosowano regulator PI.



Rysunek 11 Błoczek PID odpowiedzialny za kontrolę przepływu cieczy przez piec

Regulator PI został wykorzystany na samym początku realizacji projektu do zbierania pomiarów z instalacji. W tym wypadku posłużono się Auto-Tunningiem do wyznaczenia nastaw regulatora PI. Wykonano kolejno pretunning oraz finetunning co skutkowało uzyskaniem nastaw zaprezentowanych poniżej. Ze względu na zadowalające wyniki zaprzestano wyznaczania nastaw regulatora PI z wykorzystaniem innych metod strojenia.

PID Parameters

☒ Enable manual entry

Proportional gain: 5.20221

Integral action time: 2.821944 s

Derivative action time: 0.0 s

Derivative delay coefficient: 0.1

Proportional action weighting: 0.8

Derivative action weighting: 0.0

Sampling time of PID algorithm: 0.099998 s

Tuning rule

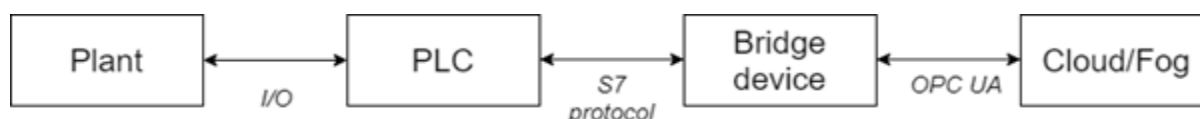
Controller structure: PI

Rysunek 12 Nastawy regulatora PI

Komunikacja

W celu wykorzystania funkcjonalności przetwarzania danych na urządzeniu zewnętrznym (Fog computing) stanęliśmy przed wyzwaniem wymiany danych pomiędzy dwoma urządzeniami (PLC oraz jednostka o dużej mocy obliczeniowej). W tym celu wykorzystaliśmy jeszcze jedno dodatkowe urządzenie jednopłytkowe nazwane przez nas "Bridge device". Do komunikacji bezpośredniej pomiędzy PLC, a bridge device wykorzystaliśmy protokół oferowany przez firmę Siemens dla swoich urządzeń S7 Protocol. Komunikacja między urządzeniem pośredniczącym, a jednostką dużej mocy była zaimplementowana za pomocą OPC UA.

Wprowadzenie "Bridge device" pozwoliło na zwiększenie abstrakcji w całym układzie oraz odizolowanie Mgły od reszty systemu.



Rysunek 13 Schemat blokowy komunikacji pomiędzy urządzeniami w części projektu z instalacją ciepłą

Konfiguracja sieci

W celu umożliwienia poprawnej komunikacji pomiędzy urządzeniami przedstawionymi na rysunku numer 1 należało skonfigurować prostą sieć. Do dalszych celów takich jak konfigurowanie komunikacji oraz aktualizowanie urządzeń użyto przewodowego połączenia, aby dzięki zaimplementowaniu Switch-a uzyskać wcześniej wspomnianą prostą topologię sieci.

Ponadto w celu lepszego funkcjonowania tak stworzonego układu urządzeń, sterownik ovi oraz komputerowi jednopłytkowemu nadano lokalny statyczny adres IP. Na komputerze jedno płytowym z serii Raspberry Pi należało odpowiednio zmodyfikować plik `/etc/dhcpd.conf`.

W taki sposób zdefiniowano adres statyczny urządzenia, adres routera, podstawowy interfejs, czyli interfejs kablowy oraz adres serwera, dzięki któremu będzie można sprawdzić, czy jest dostępne połączenie z siecią WAN, a nie tylko LAN.

Wszystkie wprowadzone modyfikacje na urządzeniu przedstawiono w tabeli, z kodem, numer 3.

```
interface wlan0
static ip_address=192.168.0.15/24
static routers=192.168.1.1
static domain_name_servers=192.168.1.1
static ip_address=192.168.1.15/24
static routers=192.168.1.1
static domain_name_servers=192.168.1.1
ping_name_web 8.8.8.8
```

Tabela 4. Konfiguracja komputera jednopłytkowego

Komunikacja S7 protocol

S7 Protocol jest to protokół komunikacyjny pozwalający na połączenie urządzeń z rodziny S7 z jakimkolwiek partnerem komunikacyjnym. Protokół ten pozwala urządzeniom na bezpośredni dostęp do pamięci np. danego sterownika PLC. Urządzenia z serii Simatic S7 wspierają natywnie protokół ISO-on-TCP np. jako serwer w architekturze klient-serwerⁱ.

W naszej pracy zdecydowaliśmy się na wybór tego protokołu ze względu na łatwość jego użycia wynikającą z obecności biblioteki Snap7 oraz bindingu dla języka Python. W celu skorzystania z tej biblioteki poczyniliśmy następujące kroki:

1. Pobraliśmy źródła biblioteki Snap7 z strony autoraⁱⁱ
2. Zbudowaliśmy bibliotekę za pomocą narzędzie "Make"
3. Zainstalowaliśmy bibliotekę przy pomocy gotowego już pliku makefile
Biblioteka została umieszczona w katalogu /usr/lib.
4. Pobraliśmy oraz zainstalowaliśmy moduł python-snap7 za pomocą menadżera pakietów o nazwie "pip"

Na potrzeby komunikacji stworzony został DB w sterowniku PLC odpowiedzialnym za sterowanie w obiekcie. Posiada on struktury, które w prosty sposób oddzielają części do których zapisuje nasz "bridge", a do których sterownik PLC. W trakcie projektowania komunikacji pomiędzy urządzeniami zaplanowany został mechanizm sprawdzający stan komunikacji pomiędzy urządzeniami. Został on zrealizowany za pomocą tzw. Heart beat'u. W naszym komunikacyjnym DB znajduje się zmienna typu Bool, której stan wysoki jest ustawiany przez sterownik PLC, a kasowany przez urządzenie "bridge". Jeżeli któreś z urządzeń nie

spełni swojego zadania w trakcie zadanego czasu bezpieczeństwa komunikacji realizowane są scenariusze awaryjne.

	Name	Offset	Data type	Accessible f...	Writable fro..	Comment
[-]	OUT	0.0	Struct	True	True	PLC out goi...
[-]	Heater_Tout_setpoint	0.0	Real	True	True	Setpoint ou...
[-]	Heater_Flow_actual	4.0	Real	True	True	Actual flow ...
[-]	Heater_Flow_setpoint	8.0	Real	True	True	Flow setpoi...
[-]	Auto_Or_Manual	12.0	Bool	True	True	Auto - true ...
[-]	Heater_Tout_actual	14.0	Real	True	True	Actual outp...
[-]	Heater_Power_setpo..	18.0	Real	True	True	Percentage ...
[-]	Counter_OUT	22.0	Int	True	True	Output cou...
[-]	IN	24.0	Struct	True	True	PLC in from ...
[-]	Flow_setpoint	24.0	Real	True	True	Flow setpoi...
[-]	Counter_IN	28.0	Int	True	True	Input ciunte...
[-]	IN_OUT	30.0	Struct	True	True	
[-]	INTERNAL	32.0	Struct	False	False	

Rysunek 14 Struktura DB

Przegląd i wybór narzędzi i środowisk do własnej implementacji algorytmu MPC

Przed rozpoczęciem rzeczywistej implementacji podjęto kluczowe decyzje odnośnie struktury, implementacji oraz środowisk wybranych przez nas do realizacji projektu. Głównym źródłem informacji odnośnie regulatorów MPC były książki: Sterowanie zaawansowane obiektów przemysłowych. Struktury i algorytmy (Piotr Tatjewski) oraz Model Predictive Control (E.F. Camacho, C. Bordons). Z książek tych wyciągnięte niezbędne informacje dotyczące regulatora DMC takie jak:

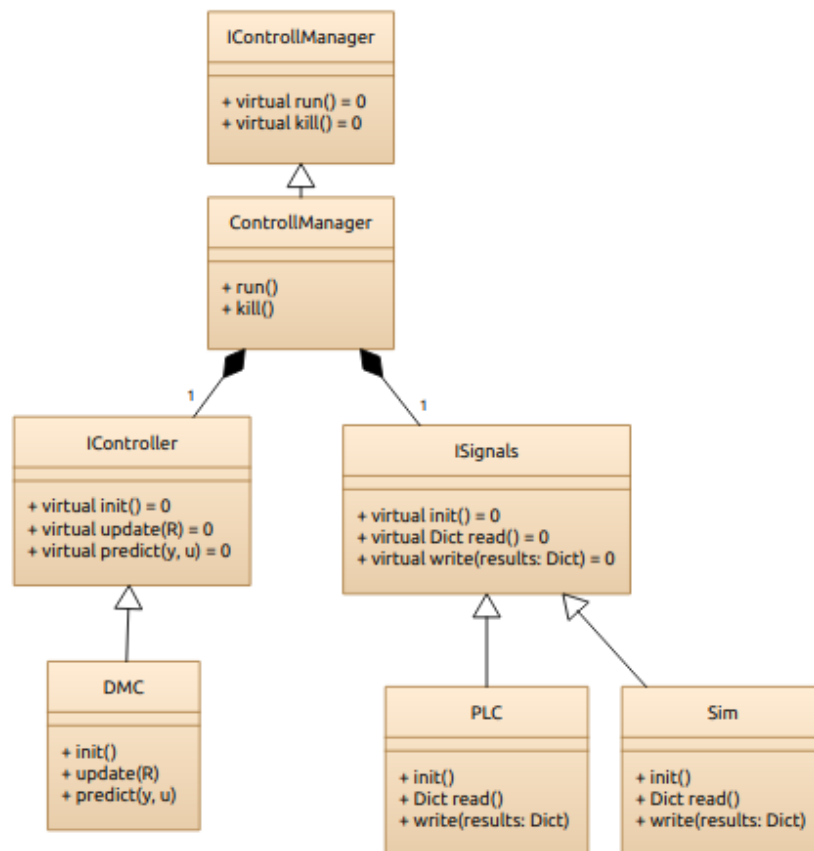
- Postać oraz wytłumaczenie zagadnienia modelu odpowiedzi skokowej układu
- Równania regulatorów w postaci macierzowej (w naszym przypadku dla obiektu SISO)
- Sposób implementacji ograniczeń na regulator – ograniczenie sterowania oraz ograniczenie przyrostu sterowania

Implementacja wybranych algorytmów z rodziny MPC w środowisku MATLAB

Pierwszym wykonanym krokiem realizacji sterowania była implementacja regulatora MPC w środowisku MATLAB. Środowisko MATLAB posiada wiele cech, które stanęły za decyzją wykorzystania go do pierwszej próby implementacji regulatora. Najważniejszymi cechami środowiska były łatwe operacje na macierzach oraz możliwość prostego podglądu wartości wszystkich zmiennych, co było niezwykle przydatne w procesie debugowania.

Implementacja algorytmu MPC na wybranych platformach

Kolejnym krokiem realizacji projektu było przeniesienie implementacji DMC na język Python. Język ten wybrano ze względu na prostotę, szybkość implementacji, dużą ilość zewnętrznych bibliotek oraz niezawodność. Wykorzystano Pythona w wersji 3.6. Jest to domyślna wersja tego środowiska w wykorzystywanym przez nas systemie operacyjnym Ubuntu Bionic. Diagram klas jest widoczny na Rys. 15.



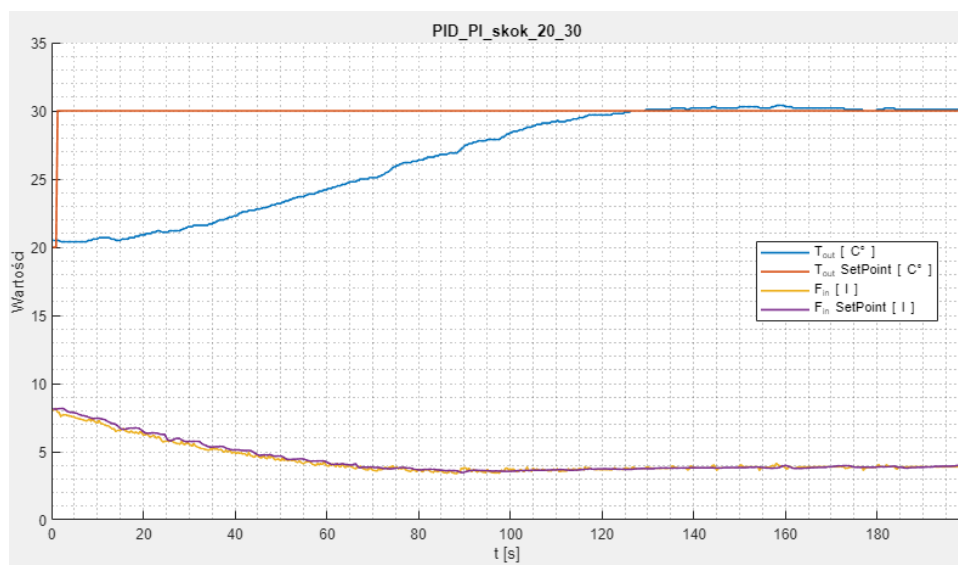
Rysunek 15 Diagram klas

Rzeczywiste testy regulacji

Po etapie implementacji projekt przeszedł do etapu testów na rzeczywistym obiekcie. Do testów używano następujących urządzeń:

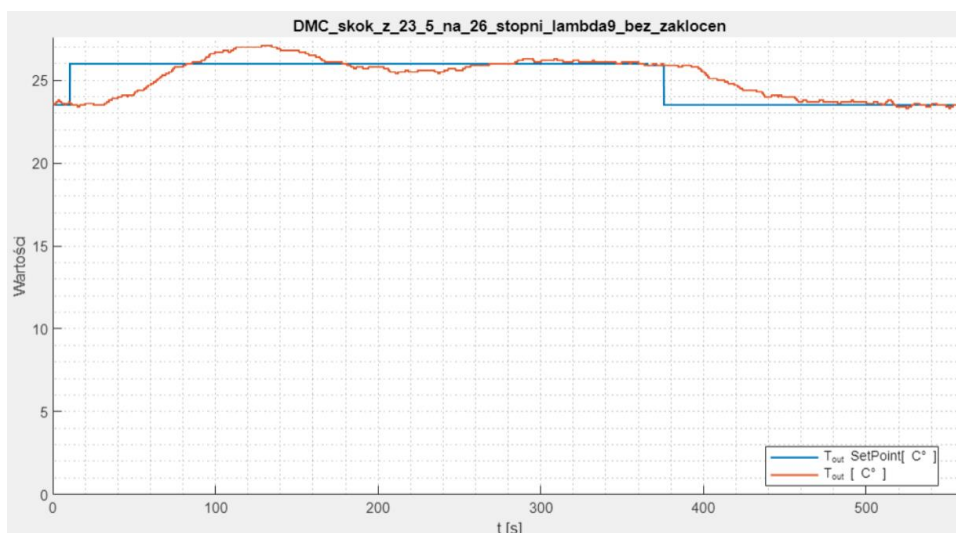
- Komputer Lenovo L580, Ubuntu Bionic, Intel(R) Core(TM) i5-8250U CPU @ 1.60GHz - używany w systemie jako serwer przetwarzający o dużej mocy obliczeniowej
- Komputer jednopłytkowy Raspberry Pi 4, Raspberry Pi OS, ARMv7 Processor rev 3 (v7l) - używany w systemie jako "bridge"
- Sterownik Siemens S7-1200

Pierwszym etapem testów były eksperymenty na lokalnym układzie regulacji. W tym przypadku użyto regulatora PID tak jak wspomniano w rozdziale: Implementacja prostego, lokalnego układu regulacji. Na Rys. X widać skok wartości zadanej z wartości 20 [°C] do 30 [°C]. Jak widać regulator PID bardzo dobrze poradził sobie z doprowadzeniem układu do stanu ustalonego, zajęło to w przybliżeniu 160 sekund.



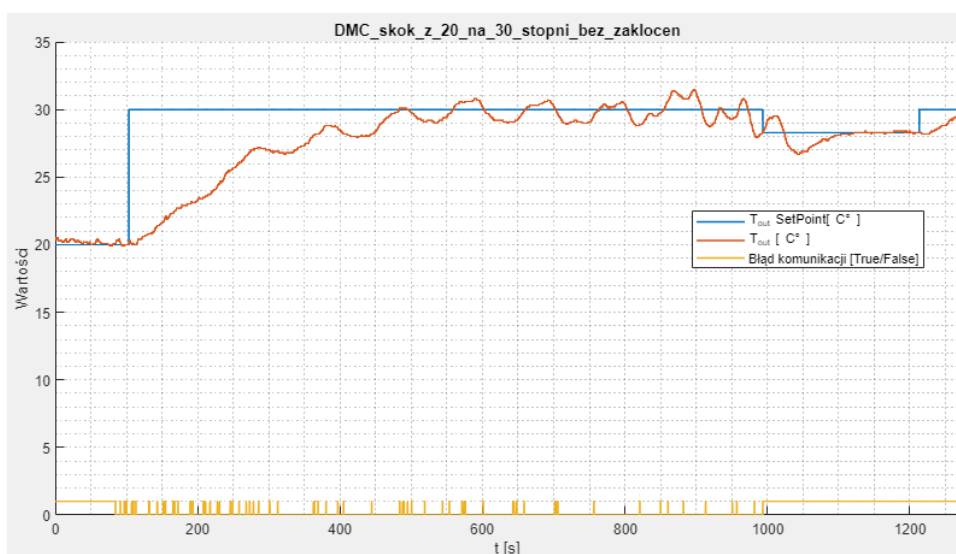
Rysunek 16 Skok wartości zadanej - regulator PID

Kolejnym krokiem było użycie regulatora DMC. Po skonfigurowaniu systemu wykonano skok wartości zadanej z wartości 23.5 [°C] do 26 [°C]. Czas regulacji w tym przypadku wyniósł 130 sekund. Wartość wyjściowa układu po 130 sekundach utrzymywała się w zakresie 26 [°C] +/- 0.78 [°C]. Wartość otoczenia wartości zadanej w tym przypadku ustawiliśmy na poziomie 3% wartości zadanej.



Rysunek 17 Skok wartości zadanej - regulator DMC

Ostatnim krokiem wykonanym podczas testów była ocena zachowania systemu w trakcie błędów komunikacji z serwerem, na którym zainstalowany został program z regulatorem DMC. Zachowanie systemu widoczne jest na Rys. 18. Jak widać, układ zachowywał się stabilnie nawet w przypadku częstych błędów komunikacji. Główną zasługą tego jest prawidłowo zaimplementowane bezuderzeniowe przejście oraz zastosowanie tzw. Heart beat'u.

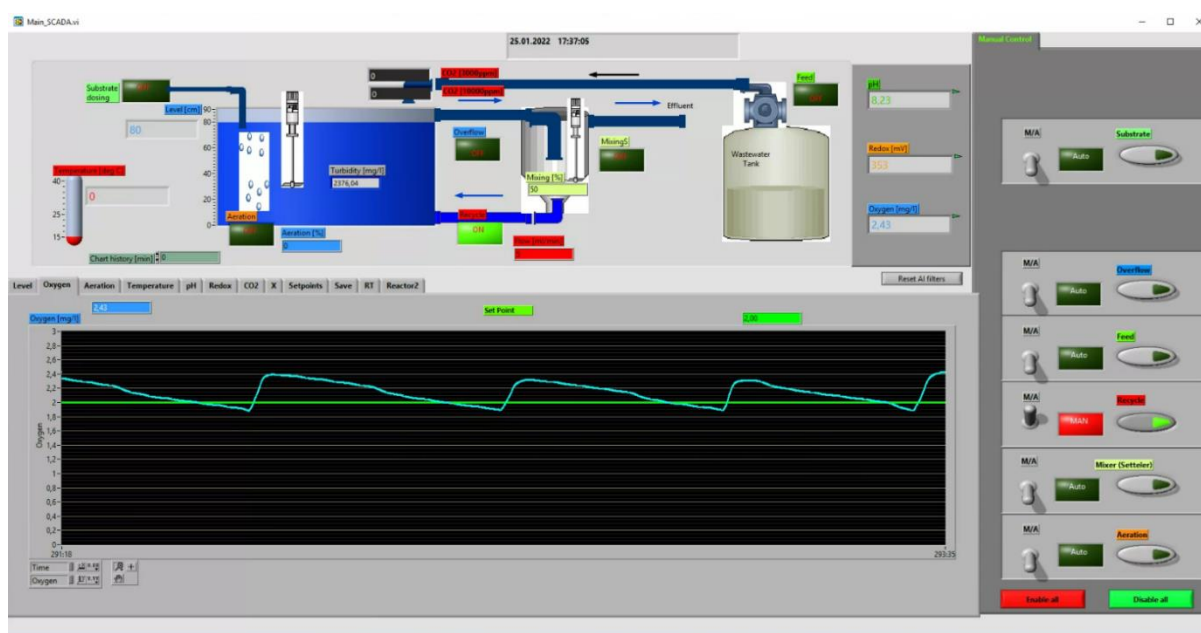


Rysunek 18 Eksperyment z częstymi błędami komunikacji

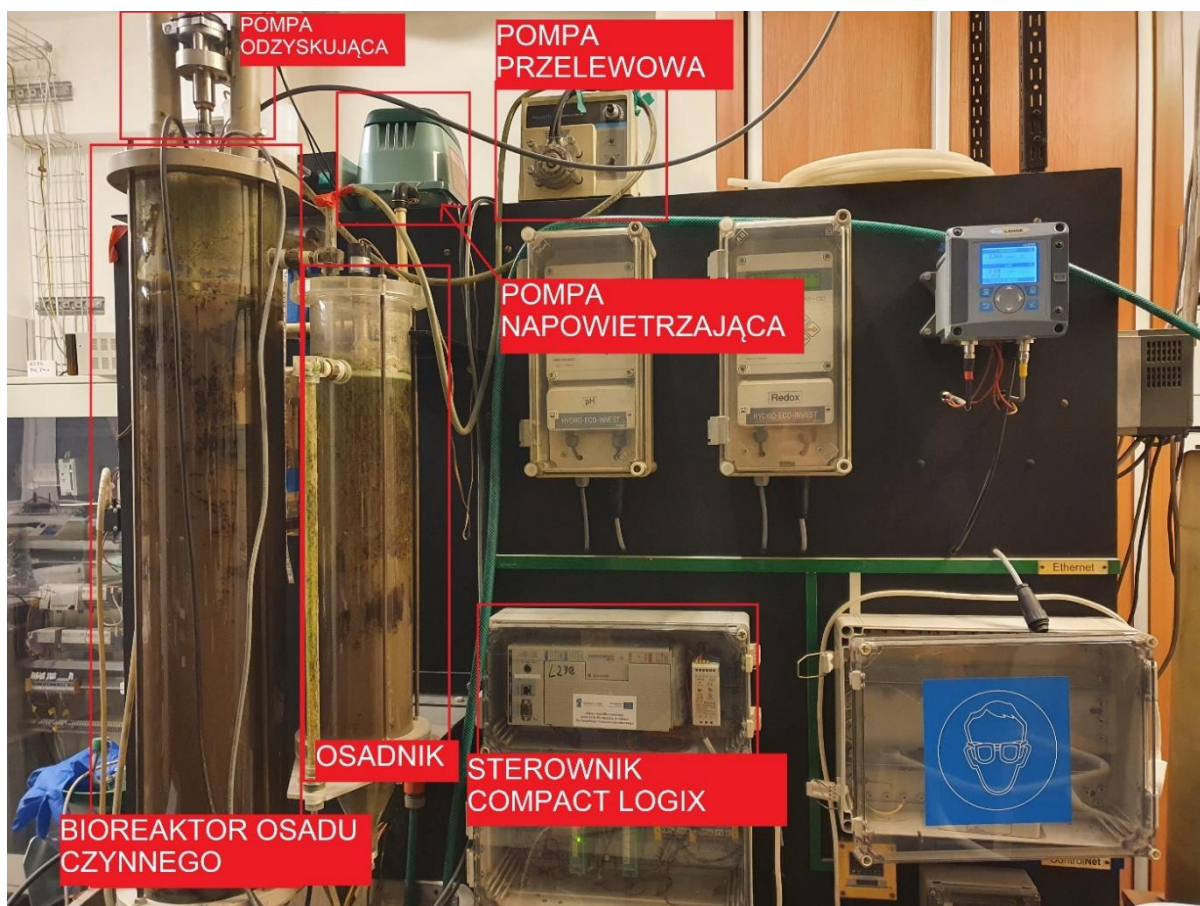
Bioreaktor

Opis instalacji

Druga przetestowana w projekcie instalacja to instalacja bioreaktora osadu czynnego. Sterowanie w niej polega na załączaniu/wyłączaniu napowietrzania osadu, a wartością mierzoną jest zawartość rozpuszczonego tlenu w osadzie. Do sterowania pompą napowietrzającą reaktor został zastosowany regulator BBPC (Boundary Based Predictive Controller) - czyli sterownik predykcyjny, pozwalający na precyzyjną kontrolę zmiennej procesowej. Zastępuje on tradycyjny sterownik on/off z histerezą. Program sterujący i nadzorujący jest uruchomiony na komputerze, a sam sterownik jest zaimplementowany na PLC.



Rysunek 19 Program sterujący i nadzorujący



Rysunek 20 Instalacja bioreaktora

Pompa przelewowa kontroluje poziom płynów w reaktorze, pompując je do osadnika, gdzie następuje rozdzielenie biomasy oraz oczyszczonej wody. Następnie osad jest pompowany do bioreaktora za pomocą pompy odzyskującej. Do osadu dostarczany jest substrat, a także powietrze z pompy napowietrzającej. Napowietrzane jest sterowane za pomocą sterownika Compact Logix, który zawiera implementację wspomnianego wcześniej regulatora BBPC.

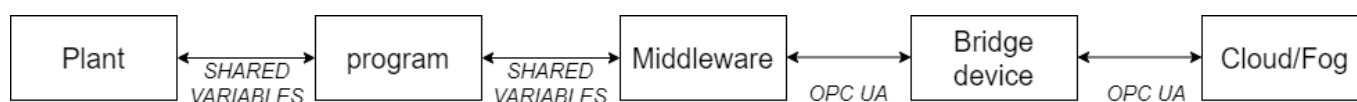
Komunikacja OPC-UA

W celu umożliwienia komunikacji pomiędzy zdalnym regulatorem oraz programem kontrolującym i nadzorującym instalację, został wykorzystany zdalny serwer OPC, zhostowany na Raspberry Pi 3.


```
pi@raspberrypi: ~/PBL-OPC
Oxygen: 1.9183673469387754
Aeration: False
Oxygen: 2.0816326530612246
Aeration: False
Oxygen: 1.9183673469387754
Aeration: False
Oxygen: 1.663265306122449
Aeration: False
Oxygen: 1.9591836734693877
Aeration: False
Oxygen: 2.5
Aeration: False
Oxygen: 2.1122448979591835
Aeration: False
Oxygen: 2.0408163265306123
Aeration: False
Control: False
Talkback: False
Watchdog: 1
Oxygen: 2.0408163265306123
Aeration: False
Oxygen: 2.0408163265306123
Aeration: False
```

Rysunek 21 Komunikaty wyjściowe z serwera

Serwer OPC udostępnia przestrzeń nazw, w której zdefiniowane są zmienne, które mogą być odczytywane i zapisywane zarówno przez “middleware”, jak i program zdalnego regulatora. Middleware to program znajdujący się na komputerze, gdzie uruchomiony jest program nadzorujący instalację, komunikujący się za pomocą OPC z serwerem i ustawiający/odczytujący wartości zmiennych dzielonych LV.



Rysunek 22 Schemat blokowy komunikacji pomiędzy urządzeniami w części projektu z bioreaktorem

Na komputerze “symulującym fog” znajduje się zdalny regulator BBPC, który również ustawia i odczytuje odpowiednie wartości zmiennych na serwerze OPC – w tym przypadku odczytuje stężenie tlenu rozpuszczonego, a zapisuje wartość sterowania napowietrzaniem.

Middleware oraz zdalny regulator zaimplementowano w języku LabView. Do komunikacji z serwerem OPC została wykorzystany moduł OPC-UA toolkit, będący submoduleń DMC.

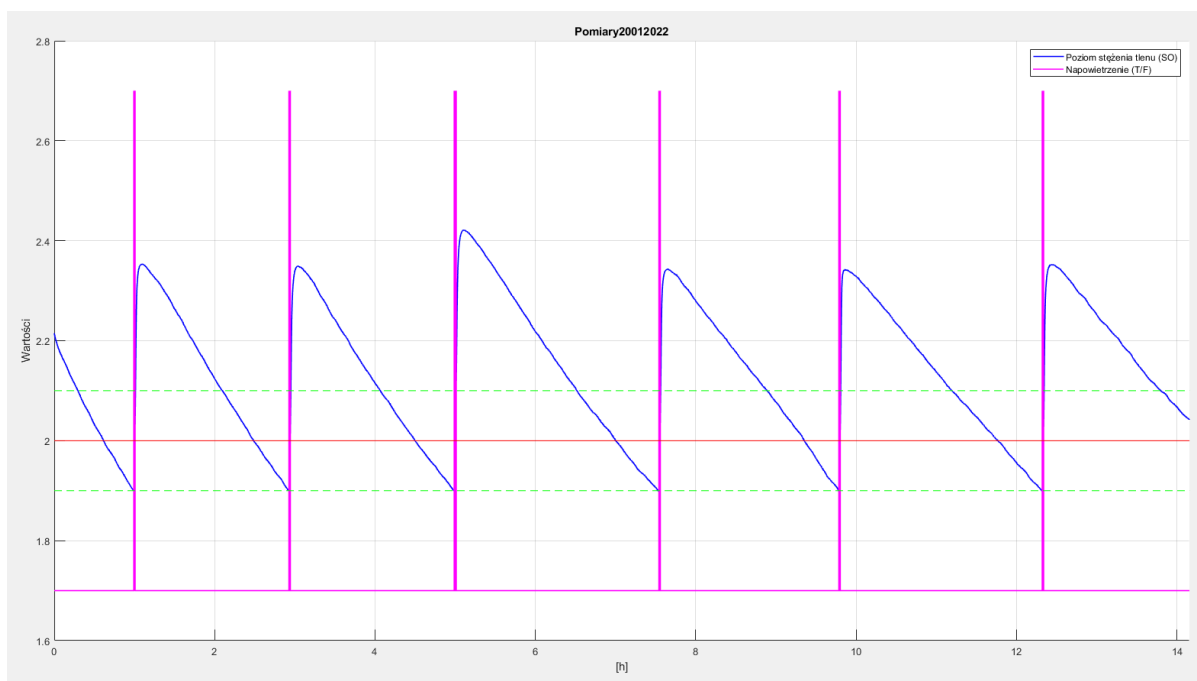
Serwer został napisany w języku Python z wykorzystaniem biblioteki asyncua (<https://pypi.org/project/asyncua/>) i został oparty o architekturę asynchroniczną, żeby działające równolegle funkcje monitorujące wartości nie blokowały siebie nawzajem.

W celu zapewnienia, że połączenie instalacja - serwer jest utrzymywane, został zaimplementowany software'owy watchdog.

Zebranie danych badawczych z instalacji bioreaktora

Wstępne nastawy parametrów modelu regulatora zostały obliczone na podstawie pomiarów zebranych podczas eksperymentu na instalacji bioreaktora.

Eksperyment polegał na wyłączeniu dopływu substratu oraz feedu, a następnie zebraniu przebiegu stężenia rozpuszczonego tlenu (S_O) w czasie.



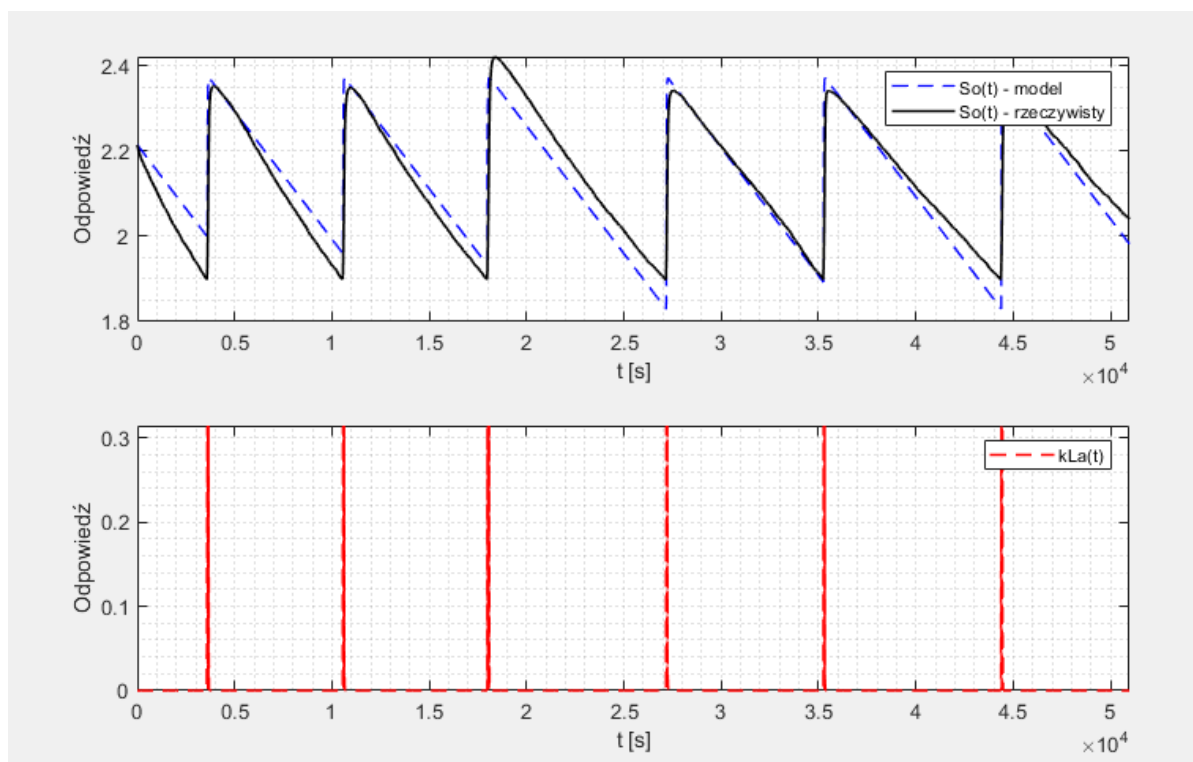
Rysunek 23 Przebiegi pomiarów z instalacji bioreaktora

Następnie został zamodelowany układ równań:

$$\frac{dS_O(t)}{dt} = k_L a(t) (S_{O_{sat}} - S_O(t)) - OUR(t),$$

$$T_{k_L a} \frac{dk_L a(t)}{dt} = -k_L a(t) + u_{air}(t - T_0).$$

na podstawie odpowiedzi którego w porównaniu z zebranymi danymi zostały wyznaczone brakujące parametry.



Rysunek 24 Dopasowywanie parametrów modelu regulatora BBPC

Wejściem regulatora jest właśnie So , a wyjściem - załączenie bądź wyłączenie pompy napowietrzającej osad. Zdalny regulator za pomocą protokołu OPC pobiera dane o stężeniu rozpuszczonego tlenu z serwera, a także wpisuje wartość sterowania napowietrzeniem do odpowiedniej zmiennej na tymże serwerze.

Podsumowanie projektu

Przedstawiono możliwości implementacji zaawansowanego algorytmu regulacji w układzie automatyki procesowej. Uzyskane w projekcie wyniki mogą stać się podstawą publikacji naukowej a także bazą do kolejnych projektów PBL.

ⁱ <https://www.ipcomm.de/protocol/S7ISOTCP/en/sheet.html>

ⁱⁱ <https://sourceforge.net/projects/snap7/files/>