

DCS

Laboratorium 3-5

Bartosz Gałecki

Michał Kwarciański

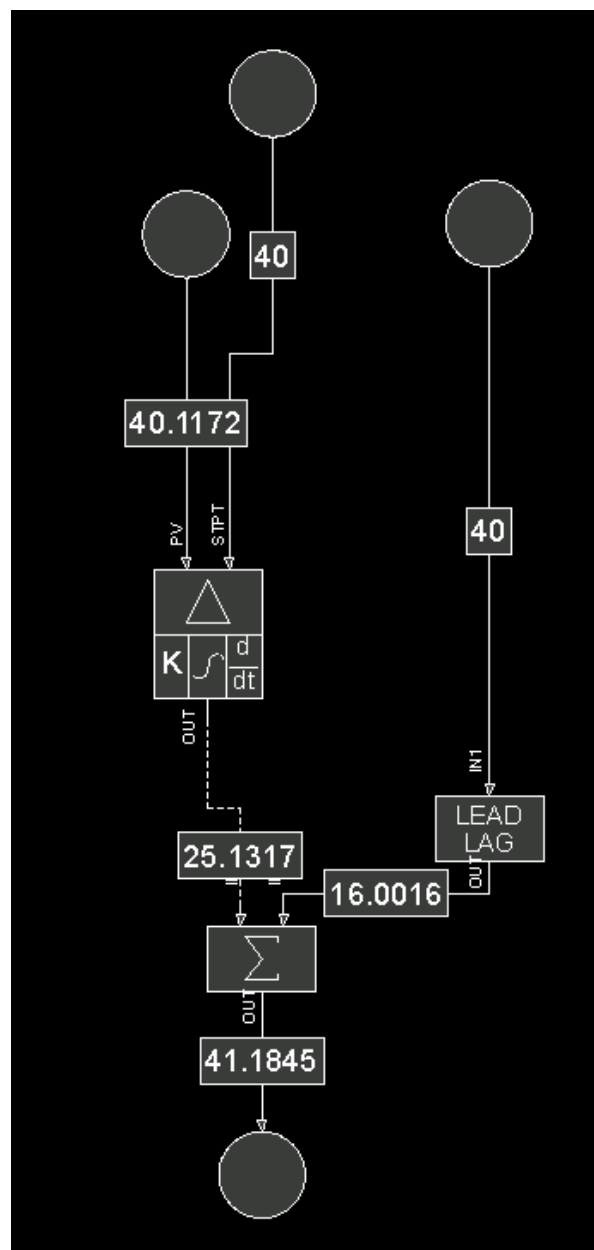
Kacper Marchlewicz

1. Cel:

Zbudowanie i zaprogramowanie w OVATION regulatora dla stanowiska grzewczo-chłodzącego, żeby regulował temperaturę i był odporny na zakłócenia.

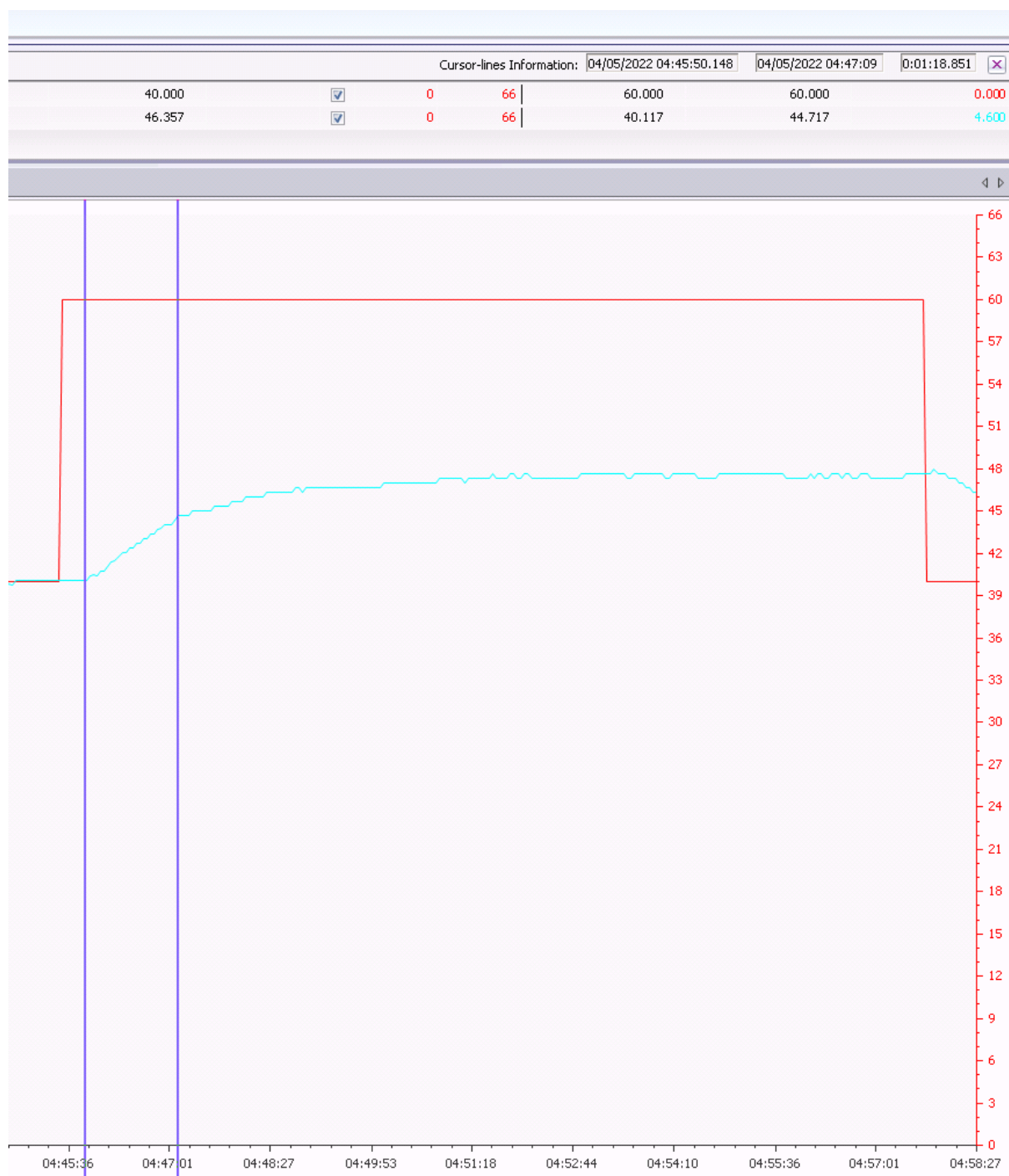
2. Budowa

W systemie Ovation stworzyliśmy następujący model regulatora PID z sprzężeniem w przód:



3. Transmitancje układu

Następnym krokiem było wyznaczenie transmitancji grzałki i wentylatora. Za punkt pracy przyjęliśmy temperaturę 40 jednostek i 40% mocy wentylatora. Wykonaliśmy dwa skoki o 20 jednostek kolejno dla grzałki i wiatraczka.



Na ich podstawie wyznaczyliśmy następujące transmitancje:

$$G_g(s) = \frac{0,37}{78,5s + 1} e^{-24,3s}$$

$$G_w(s) = -\frac{0,15}{80,8s + 1} e^{-10s}$$

Zmodyfikowaliśmy otrzymany plik do symulacji PIDa:

```
%Ten przykład pokazuje jak użyć klasę regulatora PID
%Regulator PID jest realizowany równolegle jako człony
% Proporcja: K
% Całka: 1/Ti s
% Różniczka: Kd - Kd/(Td * s + 1) - po skoku err sterowanie
ustawiane jest na wartość Kd i zmniejszane po LAGu
clear all;

%classPID(K, Ti, Kd, Td, Tp, Hlim, Llim, Dir, AutoMan, ManVal)
%DIR: 1 - direct (SP-PV), 0 - indirect (PV-SP)
% AutoMan = 0 regulator na wyjściu podaje wartość sterowania
ręcznego ManVal
% AutoMan = 1 regulator na wyjściu podaje wartość wyliczoną z prawa
regulacji
%Przykład: deklaracja regulatora D
p=classPID(1, 0, 1, 10, 1, 100, -100, 1, 1, 0);
heat=classLEADLAG(0.375, 0, 78.5, 1, 100, -100);
fan=classLEADLAG(-0.15, 0, 80.8, 1, 100, -100);
ff = classLEADLAG(0.405, 78.5, 80.8, 1, 100, -100);
%PID
%reTune(obj, K, Ti, Kd, Td)
% funkcja umożliwia zmianę nastaw regulatora
p.reTune(5, 20, 1, 0.5)

dv=40;
stpt = 15;
temp=0;
u=0;

y = [];
stero = [];
l = [];
j = [];
y_zad = [];
for i=1:1:1000
    %metoda wyliczająca prawo PID w oparciu o PV i STPT
    u = p.calc(temp,stpt);
    stero(i) = u+ff.calc(40);
    j(i) = u;
    if i<=15
        temp = heat.calc(0) + fan.calc(40);
    else
        temp = heat.calc(stero(i-14))+ fan.calc(40);
    end

    y(i) = temp;
    %tutaj należy symulować obiekt
    %wyliczenie wyjścia obiektu na następny krok wykorzystując u

end
stpt = 15;
for i=1:1:1000
```

```

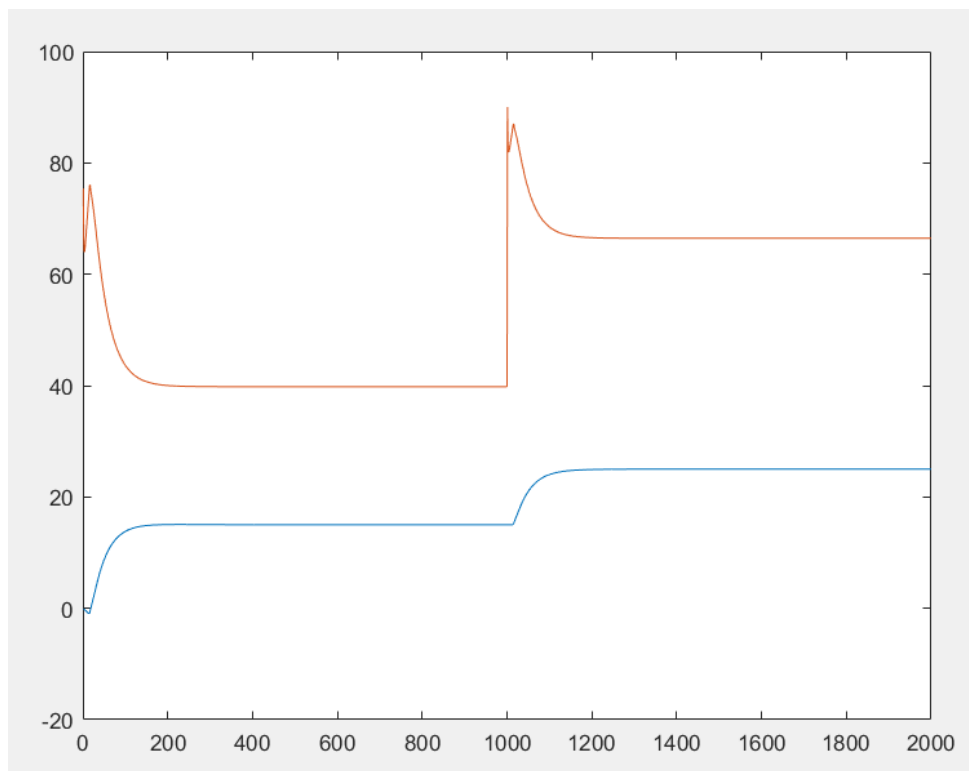
y_zad(i) = stpt;
u = p.calc(temp,stpt);
stero(1000+i) = u+ff.calc(50);
j(1000+i) = u;
if i < 11
    temp = heat.calc(stero(1000+i-14))+fan.calc(40);
else
    temp = heat.calc(stero(1000+i-14))+fan.calc(50);
end

y(1000+i) = temp;
l(i) = temp;

end
plot(y)
hold on
plot(j)
hold off
f= y_zad-l;
k=0;
for i = f
    k=k+1;
    if f(k) < 0.5 && f(k)>-0.5
        f(k) = 0;
    end
end
norm(f)

```

Otrzymaliśmy wykres:



Pierwszy skok jest skokiem do początkowej wartości zadanej.

3. Regulacja PID

Po wielu testach parametry regulatora dostroiliśmy na wartości poniżej, nie odbiegają dużo od wyznaczonych w matlabie.


PGAIN = 4

INTG = 20

DGAIN = 1

DRAT = 1

	Current Value	Tuned Value
AUSC	NONE	
AUSCP	INT	
TYPE	NORMAL	
ACTN	INDIRECT	
CASC	NORMAL	
DACT	NORMAL	
DBND	0	
ODBND	0	
DOPT	SINGLE	
INHB	ENABLE	
ERRD	0	
PGAIN	4	
INTG	20	
DGAIN	1	
DRAT	1	
TRAT	2.5	
PVG	1	
PVB	0	
PVT5	100	
PVB5	0	
SPTG	1	
SPTB	0	



4. Feedforward:

Transmitancje zakłócenia obliczyliśmy z odpowiedzi skokowej i na jej podstawie oraz transmitancji obiektu ustaliliśmy parametry bloku feedforward:

	Current Value	Tuned Value
TYPE	LEADLAG	
GAIN	0.4	
LEAD	78.5	
LAG	80.8	
TRAT	2.5	
TP5C	50	
BT5C	-50	

Wykresy z testów na zajęciach:

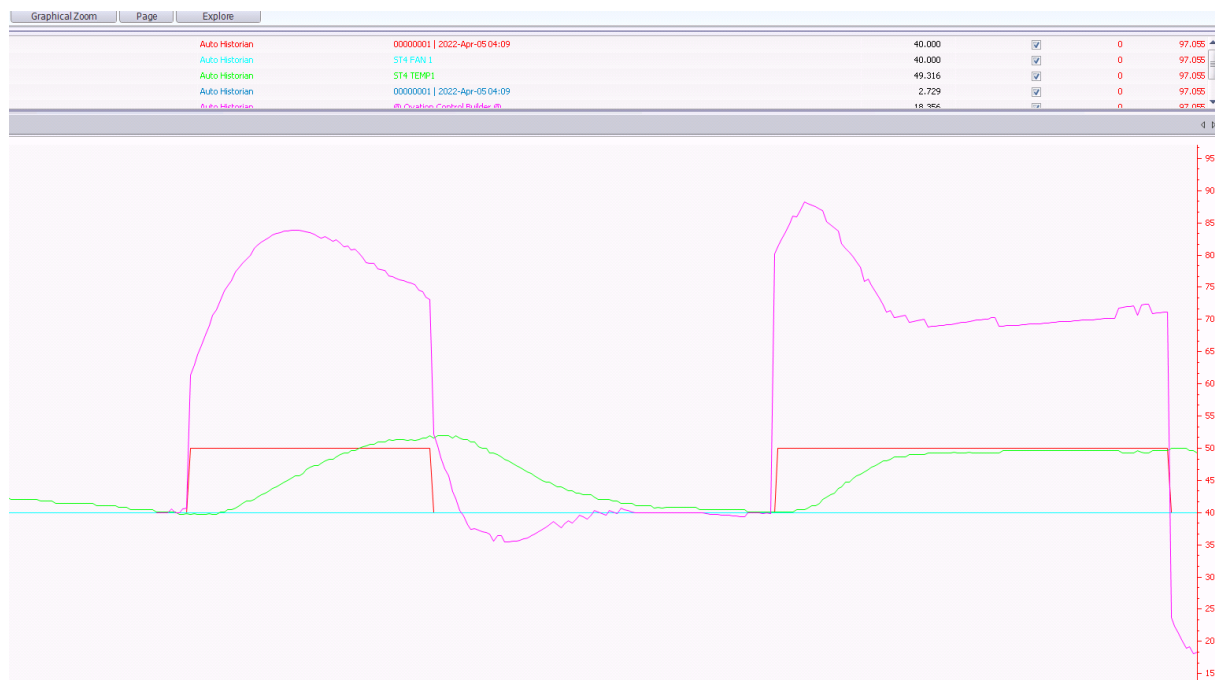
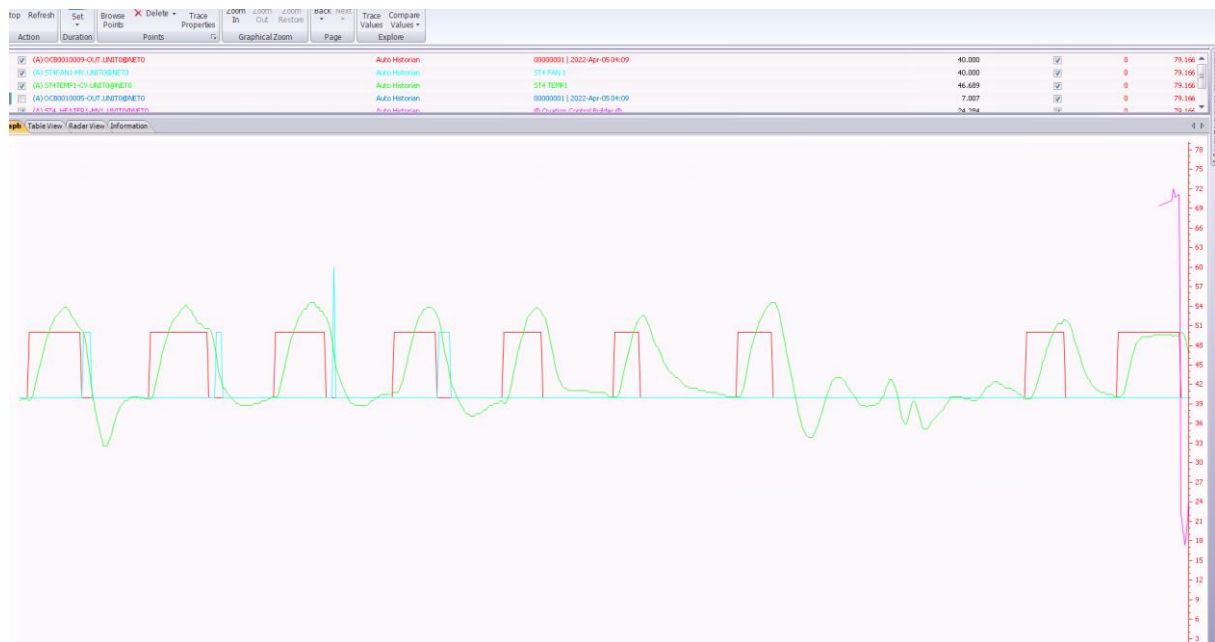
Legenda:

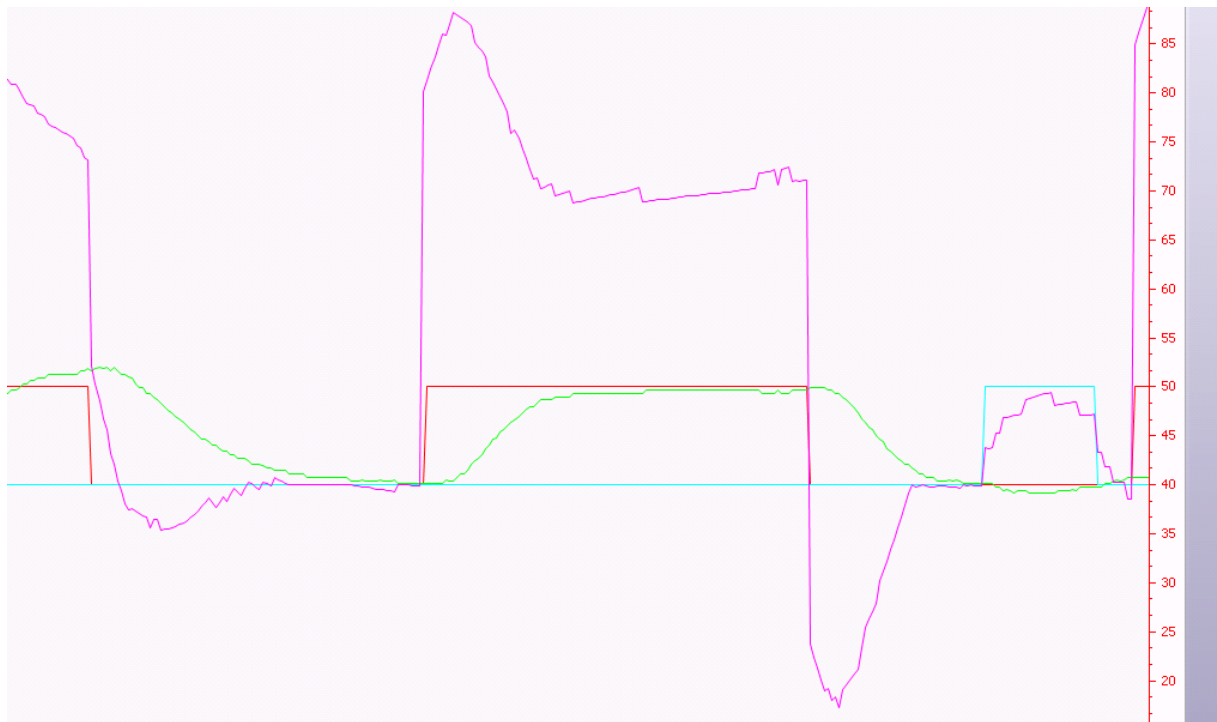
Czerwony – temperatura zadana

Zielony – temperatura grzałki

Niebieski – wentylator

Fioletowy – sygnał sterujący





5. Podsumowanie

Nasz regulator wygrał konkurs ze zmianą zadawanej temperatury, lecz niestety nie okazał się najlepszy przy odprężaniu. Mogło to wynikać z niewystarczającej ilości czasu poświęconej na tą część zadania.