FAKULTETA ZA
ELEKTROTEHNIKO

DIGITALNO VODENJE

Laboratorijske vaje

GREGOR KLANČAR

Kazalo

1. Spoznavanje z mikrokrmilniškim okoljem Arduino Uno

Pri vaji se boste spoznali z mikrokrmilniško tiskanino Arduino Uno, ki vsebuje mikrokrmilnik ATmega328. Mikrokrmilniki Arduino so trenutno zelo popularni, na spletu je moč najti veliko informacij, programskih knjižnic in tudi strojne opreme za različne razširitve.

Mikrokrmilnik bomo v prihodnjih vajah uporabili za izvedbo digitalnega vodenja procesnih laboratorijskih naprav. V nadaljevanju je najprej podano nekaj osnovnih informacij in programskih primerov, ki vam bodo v pomoč pri izvedbi nalog.

1.1 Opis tiskanine Arduino Uno

Za namen laboratorijski vaj bomo uporabili tiskanino Arduino Uno, ki je prikazana na sliki 1.1. Na domači strani *www.arduino.cc* je moč najti veliko dodatnih informacij, primerov programov, knjižnic in razširitvenih vezij. Arduino Uno mikrokrmilniška tiskanina vsebuje mikrokrmilnik ATmega328 (*http://www.atmel.com/Images/doc8161.pdf*) in ima 6 analognih vhodov ter 14 digitalnih vhodno/izhodnih pinov od katerih jih lahko 6 uporabimo za PWM izhode. Deluje s 16MHz taktom in ima USB konektor preko katerega jo lahko programiramo, napajamo in komuniciramo z osebnim računalnikom.

Ostali podatki tiskanine so:

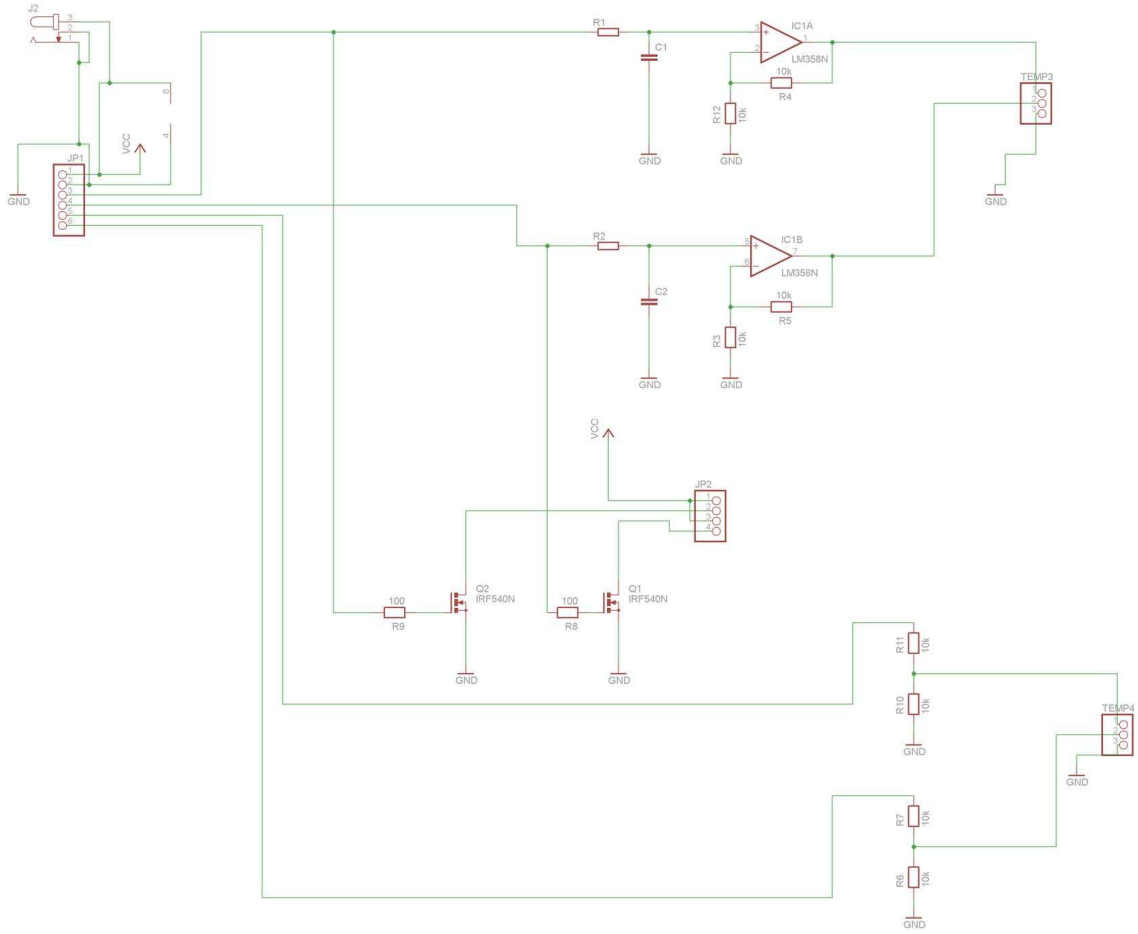
Slika 1.1: Tiskanina Arduino Uno

Microcontroller	ATmega328
Operating Voltage	5V
Input Voltage(recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
Analog Input Pins	6
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328)
EEPROM	1 KB (ATmega328)
Clock Speed	16MHz

1.2 Prilagoditveno vezje

Z mikrokrmilnikom Arduino Uno (v nadaljevanju Arduino) bomo izvedli regulacijo različnih laboratorijskih naprav. Laboratorijske naprave so opremljene z močnostno elektroniko, ki ima standardizirane napetostne vhode 0-10 V (aktuatorji) in napetostne izhode 0-10 V (senzorji). Arduino pa ima 10 bitne analogne vhode (A/D), na katere lahko priključimo signale v območju 0-5 V. Za analogne izhode pa uporabimo PWM (pulzno širinska modulacija) izhode. Določene procese lahko krmilimo neposredno z PWM signalom, če je frekvenca PWM-ja veliko večja od dinamike procesa. Proces (elektromotor) v tem primeru sam povpreči PWM signal in tako iz njega dobi le nizkofrekvenčni signal, s katerim smo mo-

dulirali nosilno frekvenco PWM-ja. V primeru, da potrebujemo čisti analogni izhod, pa moramo narediti ustrezen filter z elektronskimi elementi (upor, kondenzator, operacijski ojačevalnik). Prilagoditveno vezje, preko katerega lahko Arduino priklopimo na laboratorijske naprave prikazuje slika 1.2. Za analogne



Slika 1.2: Prilagoditveno vezje

vhode uporabimo uporovni delilnik, ki vhodno napetost 0-10 V pretvori v 0-5 V. Mikrokrmilnik ATmega328 ima 10 bitno analogno digitalno pretvorbo (A/D) za analogne vhodne signale, kar pomeni, da se vhodno območje napetosti 0-5 V diskretizira v $2^{10} = 1024$ digitalnih vrednosti.

Analogne izhode pa lahko posredujemo preko PWM izhodov. Lahko uporabimo MOS FET tranzistor, v kolikor želimo močnostni PWM izhod oziroma filter prvega reda in operacijski ojačevalnik z ojačenjem 2, če želimo analogni (filtriran PWM signal) izhod 0-10V.

Za namen laboratorijskih vaj izberemo frekvenco PWM signalov $f_{PWM} = 33$

kHz, kjer predpostavimo, da je f_{PWM} precej višja od maksimalne koristne frekvence signala f_S s katero moduliramo PWM signal $f_{PWM} \geq 20 f_S$. Pri motornih pogonih je dostikrat pomembno tudi, da je f_{PWM} izven slušnega območja ($f_{PWM} > 18$ kHz), da se izognemo motečega piskanju motornega pogona. Izbira f_{PWM} je navzgor omejena z stikalnimi elementi (hitrosti MOS FET tranzistorjev), navzdol pa je omejena z uporabljenim procesom (njegovim frekvenčnim pasom), ki mora PWM signal gladiti (povprečiti).

Da iz PWM signala izluščimo modulirano frekvenco signala f_S uporabimo filter prvega reda (RC člen) z lomno frekvenco $f_{FILT} = \frac{1}{2\pi RC}$. Pri lomni frekvenci f_{FILT} je ojačenje filtra -3 dB oziroma $\frac{1}{\sqrt{2}}$, kar pomeni, da se amplituda signala pri lomni frekvenci zniža na 70% prvotne. Pri nižjih frekvencah pa filter slabi manj pri večjih pa več (recimo pri $f = 10 f_{FILT}$ se amplituda signala zmanjša na 10% prvotne). Iz povedanega sledi, da lomno frekvenco filtra prvega reda lahko določimo kot

$$f_{FILT} \simeq f_S$$

1.3 Digitalno vodenje

Arduino je v osnovi digitalni računalnik, zato najbolj naravna implementacija regulatorja v diskretni obliki (po času in amplitudi). Najprej je potrebno določiti potrebno frekvenco vzorčenja f_{VZ} (oz. čas vzorčenja T_{VZ}), s katero bomo zajemali podatke iz senzorjev preko A/D pretvorbe in posredovali izhode regulatorja na PWM izhod. Glede na Shannonov teorem mora biti frekvenca vzorčenja vsaj dvakrat večja (v praksi pa več 5-10 krat) od koristne frekvence signala f_S (da lahko rekonstruiramo sinusni signal).

$$f_{VZ} > 2 f_S$$

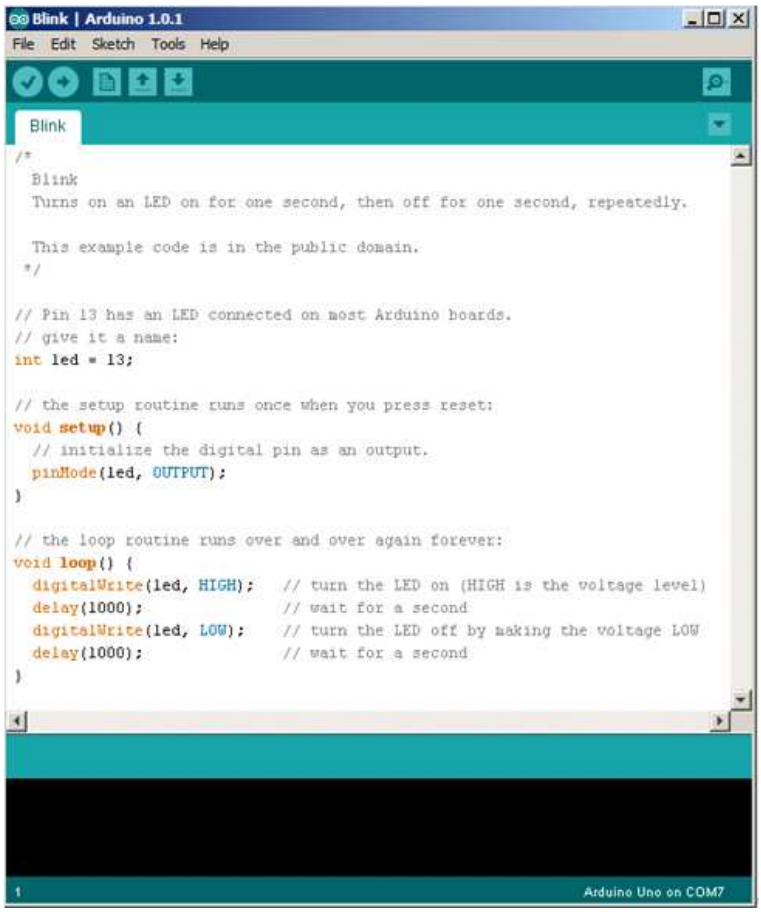
Koristno frekvenco f_S procesnega signala definira najmanjša časovna konstanta procesa T_{min} z izrazom

$$f_S = \frac{1}{2\pi T_{min}}$$

1.3.1 Izvedba regulatorja v programskem okolju Arduino

Programiranje Arduino Uno tiskanine poteka v okolju Arduino (slika 1.3) v jeziku C. Podrobna navodila za namestitev okolja in gonilnikov za tiskanino Arduino Uno najdete na *http://arduino.cc/en/Guide/HomePage*.

V nadaljevanju je prikazan pristop k načrtovanju proporcionalnega regulatorja (zgolj za namen demonstracije). Recimo, da imamo proces, katerega zanimivo



Slika 1.3: Programsko okolje Arduino

frekvenčno območje procesa sega do $f_S = 160$ Hz. Uporabimo PWM signale s frekvenco $f_{PWM} = 33$ kHz za vhode v proces. Določimo časovno konstanto filtra $RC = \frac{1}{2\pi f_{FILT}} = 1,6$ ms, s katerim bomo filtrirali PWM signal, tako da dobimo na vhodu procesa zvezni signal. Želeno konstanto dosežemo z izbiro $R = 10$ kΩ in $C = 100$ nF. Določimo še frekvenco vzorčenja $f_{VZ} = 6f_S \simeq 1000$ Hz.

Periodično vzorčenje procesnih signalov najlažje dosežemo s prekinitveno rutino (timer interrupt). Za izvedbo časovnika lahko uporabimo obstoječo knjižnico *FlexiTimer2* (lahko tudi katero drugo ali pa sprogramiramo sami), ki uporabi drugi časovnik v mikrokrmilniku. Za izvedbo PWM izhodnih signalov pa uporabimo knjižnico *Timer1*, ki uporablja prvi časovnik. Analogne vrednosti beremo preko A/D pretvorbe z ukazom *analogRead()*. Celotno okolje nadgradimo še s serijsko komunikacijo, preko katere spremljamo procesne vrednosti in posredujemo ukaze za referenco. Sledi celoten izpis programa.

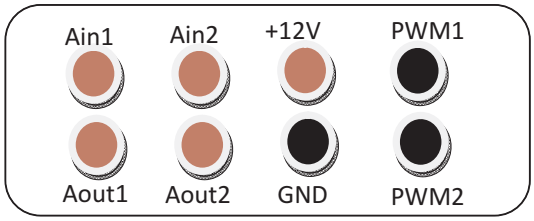
```
// example how to do a simple P controller for some proces
#include "FlexiTimer2.h"
#include "TimerOne.h"
#define SERIALc

int refPercent = 0;   // for incoming serial data
int stevcPrint=0;
bool Flag_Control;
float voltage=0.0;
float reference=6.3;
float input=0;

////////////////////////////////////////////////////////////////////////////
void setup() {
  Flag_Control=false;
  pinMode(13, OUTPUT);
  FlexiTimer2::set(2, 1.0/2000, timerISR); // unit=2, resolution 1/2000, t=unit*resolution=1ms
  FlexiTimer2::start();

  Timer1.initialize(500000);         // initialize timer1, and set a 1/2 second period
  Timer1.pwm(9, 0, 30);              // setup pwm on pin 9, 0% duty cycle, period=30us (33,3kHz)  % možen je pin9 ali pin10

  // initialize serial communication at 9600 bits per second:
  #if defined(SERIALc)
    Serial.begin(9600);
  #endif
}
////////////////////////////////////////////////////////////////////////////

void loop() {
  if (Flag_Control){
  }
  else{
      #if defined(SERIALc)
        if(stevcPrint>500) {   // do communication only each 0.5 second when we have time
           stevcPrint=0;
           Serial.print("R= ");     // print out the reference
           Serial.println(reference);
           Serial.print("Y= ");
           Serial.println(voltage); // print out the last value you read
           Serial.print("U= ");     // print out the outpur of the controller
           Serial.println(input);
        }
            // receive data for the reference
        if (Serial.available() > 0) {
                // read the incoming byte:
                refPercent = Serial.parseInt();
                if(refPercent>0&&refPercent<101){   // you can input reference in percentage
                  if (refPercent==1) refPercent=0;
                  reference=(float)refPercent/10;
                }
        }
      #endif
  }
}
////////////////////////////////////////////////////////////////////////////
void timerISR(){
  Flag_Control=true;
  control();
}
////////////////////////////////////////////////////////////////////////////
void control() {
  static boolean output = HIGH;
  static int duty=0;

  // read analog input A0
 int sensorValue = analogRead(A0); // read the input on analog pin 0
 voltage = sensorValue * (10.0 / 1023.0); // Convert the analog reading (which goes from 0 - 1023) to a voltage (0 - 10V):

  // calculate error and controller output
  float err=reference-voltage;
  float U_DT=2.1; // input voltage for the process in the working point
  float Kp=0.4;   // controller gain

  float Uout=err*Kp+U_DT;
  if (Uout>10)
    Uout=10;
  input= Uout;

// write PWM on pin9 for the Uout in range 0-10V
 duty=1023*Uout/10;
 Timer1.setPwmDuty(9, duty);   // pin 9 duty 50%

  stevcPrint++;
  Flag_Control=false;
}
```

1.3.2 Vezalni načrt

Mikrokrmilnik Arduino Uno in prilagoditveno vezje sta zapakirana v škatli, ki je prikazana na sliki 1.4. Na čelni plošči škatle priključimo procesne signale, na zadnji strani pa konektor za USB komunikacijo in dodatno napajanje. V kolikor uporabljamo prilagoditveno vezje, ki nam da na izhodu 0-10 V signale, moramo priključiti še napajanje 12 V (plus je notranji kontakt). Če uporabljamo le mikrokrmilnik Arduino Uno, se le ta napaja preko USB kabla in dodatno napajanje +12 V ni potrebno.

Slika 1.4: Mikrokrmilnik Arduino Uno s prilagoditvenim vezjem

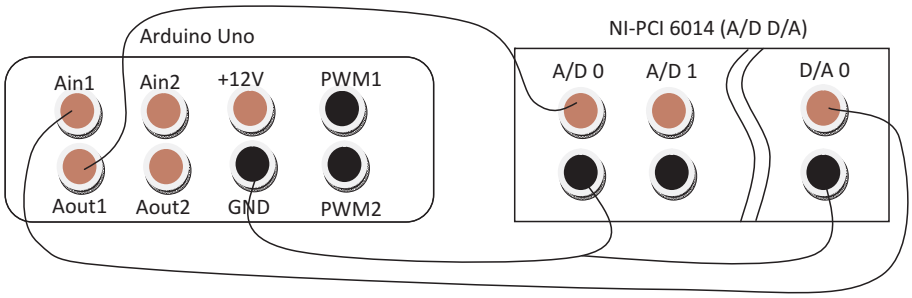
Oznake vhodov in izhodov na škatli so prikazane na sliki 1.5. Na analogne vhode škatle (Ain1 in Ain2) priljučimo analogne signale (iz senzorjev) v območju 0-10 V. Na analogne izhode škatle (Aout1 in Aout2) pa priključimo vhode v proces (signali so šibkotokovni !). Analogni izhodi in vhodi imajo skupno maso - sponka GND. Lahko uporabimo tudi PWM močnostne analogne izhode, kjer porabnik oz. aktuator (motor) priključimo med sponko +12 V in na PWM (PWM1 ali PWM2) sponko.



Slika 1.5: Vhodi in izhodi

Vodenje simuliranega procesa v okolju Matlab-Simulink

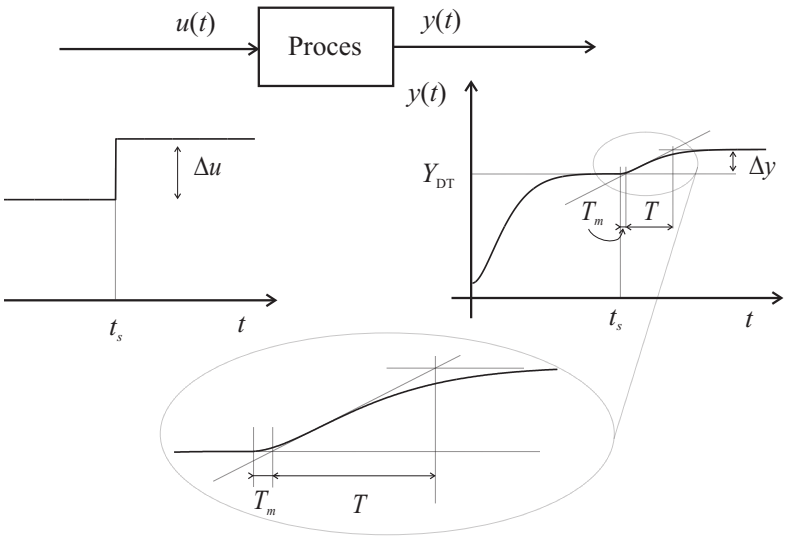
Če želimo voditi simuliran proces v okolju Simulink lahko uporabimo NI-PCI 6014 (A/D in D/A) vmesnik. Analogne izhode (filtriran PWM signal) prilagoditvenega vezja Arduino priključimo na želene analogne vhode (A/D) NI-PCI 6014 vmesnika. Analogne vhode (A/D) prilagoditvenega vezja Arduino pa priključimo na analogne izhode (D/A) NI-PCI 6014 vmesnika. Primer vezave za en analogni izhod in en analogni izhod prikazuje slika 1.6.



Slika 1.6: Priklop Arduino Uno vmesnika na NI-PCI 6014 za vodenje simuliranega procesa v okolju Simulink

1.4 Ocena linearne prenosne funkcije v delovni točki

Gre za praktičen pristop, kjer ocenimo linearni model procesa v neki delovni točki s pomočjo eksperimenta. Večinoma je to odziv procesa na stopničast signal iz delovne točke na vhodu procesa. Amplituda stopnice mora biti primerno izbrana, torej dovolj majhna, da ne preseže meja linearnega področja in hkrati dovolj velika, da je vpliv šuma in motenj procesa zanemarljiv. Postopek določitve lineariziranega modela prvega reda z zakasnitvijo ponazarja slika 1.7. Preden naredimo stopničasto spremembo vhoda, moramo počakati, da se proces ustali v



Slika 1.7: *Določitev linearnega modela 1. reda z zakasnitvijo v delovni točki U_{DT}, Y_{DT}, s pomočjo odziva na stopnico.*

delovni točki. Iz dobljenega odziva v delovni točki določimo časovno konstanto T, mrtvi čas procesa T_m in ojačenje $K = \frac{\Delta y}{\Delta u}$.

V večini primerov lahko dobljeni odziv opišemo s prenosno funkcijo 1. reda brez zakasnitve ($T_m = 0$)

$$G(s) = \frac{K}{Ts + 1}$$

kar je dober približek za procese 1. reda oz. procese z eno dominantno časovno konstanto.

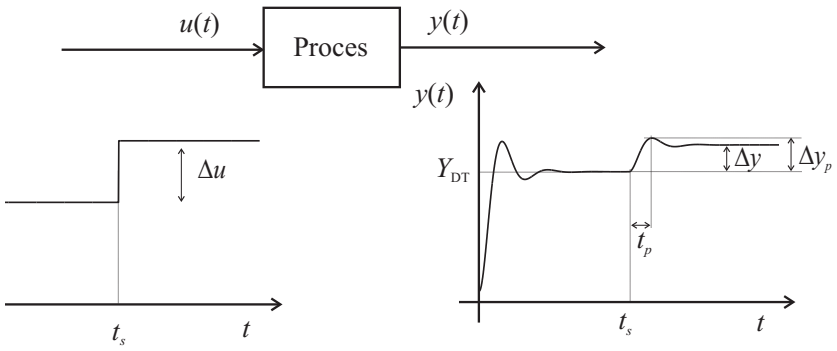
Procese z zakasnitvijo in procese višjega reda z eno dominantno časovno konstanto in nadkritično dušenim odzivom ($\zeta \geq 1$) lahko aproksimiramo z modelom 1. reda z zakasnitvijo

$$G(s) = \frac{K}{Ts + 1} e^{-sT_m}$$

Procese s podkritično dušenim odzivom ($0 \leq \zeta < 1$) na stopničasto vzbujanje lahko aproksimiramo z 2. redom. Odziv nelinearnega procesa 2. reda ponazarja slika 1.8. Model procesa v delovni točki (U_{DT}, Y_{DT}) zapišemo v obliki

$$G(s) = \frac{K{\omega_n}^2}{s^2 + 2\zeta\omega_n s + {\omega_n}^2}$$

kjer je K ojačenje, ω_n lastna frekvenca in ζ koeficient dušenja. Pola tega procesa sta konjugirano kompleksna $s_{1,2} = \zeta\omega_n \pm j\omega_n\sqrt{1 - \zeta^2}$. S pomočjo odziva iz



Slika 1.8: *Določitev linearnega modela 2. reda z ($0 \leq \zeta < 1$) v delovni točki U_{DT}, Y_{DT}, s pomočjo odziva na stopnico.*

delovne točke na sliki 1.8 in upoštevanjem relacij

$$M_p = \frac{\Delta y_p - \Delta y}{\Delta y} = e^{\frac{-\pi\zeta}{\sqrt{1-\zeta^2}}}$$

$$t_p = \frac{\pi}{\omega_d}$$

$$\omega_d = \omega_n\sqrt{1-\zeta^2}$$

lahko določimo parametre modela 1.4 kot

$$\zeta = \sqrt{\frac{(ln(M_p))^2}{\pi^2 + (ln(M_p))^2}}$$

$$\omega_n = \frac{\pi}{t_p\sqrt{1-\zeta^2}}$$

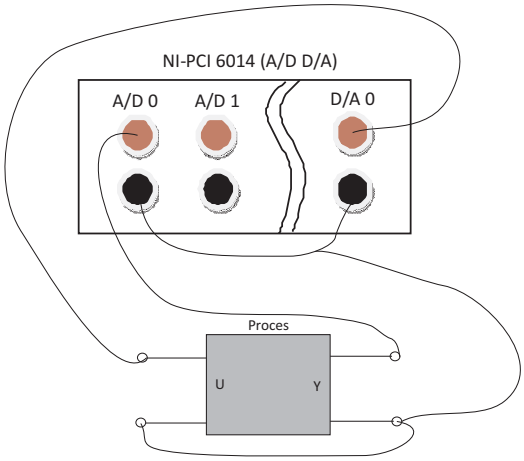
$$K = \frac{\Delta y}{\Delta u}$$

Tako ocenjeni modeli procesa veljajo le v okolici delovne točke (velikost okolice je odvisna od nelinearnosti procesa).

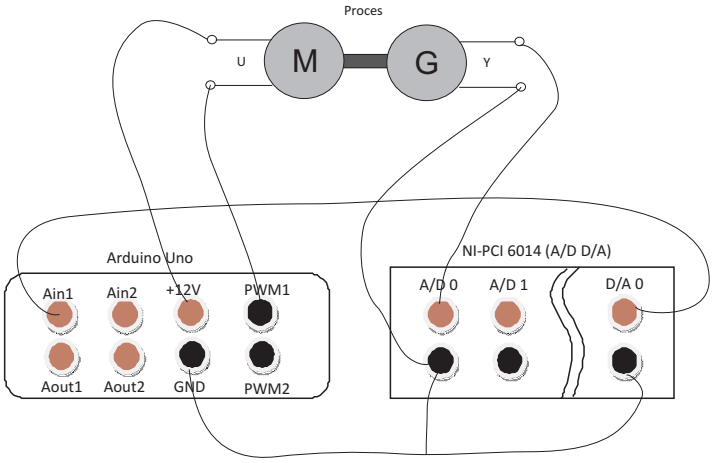
Vezalno shemo prikazujeta slika 1.9 za procese, ki zahtevajo analogni vhod in imajo na vhodu močnostni ojačevalnik (večina procesov v laboratoriju) in 1.10 za procese, ki jih lahko vzbujamo z močnostnim PWM signalom na vhodu.

1.4.1 Vrednotenje linearnega modela

V okolici delovne točke dobljeni linearni model procesa (teoretično z linearizacijo ali eksperimentalno) želimo primerjati s procesom, da ugotovimo njegovo ustreznost. Pravilna shema za vrednotenje, ki upošteva delovno točko je prikazana na sliki 1.11.



Slika 1.9: *Vezalna shema za procese, ki zahtevajo analogni vhod in imajo na vhodu močnostni ojačevalnik.*

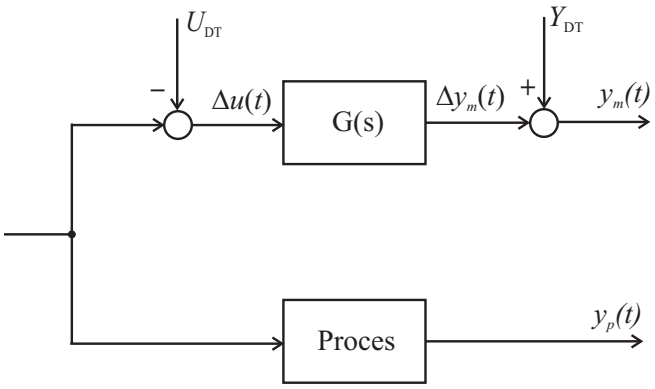


Slika 1.10: *Vezalna shema za procese, ki jih lahko vzbujamo z močnostnim PWM signalom na vhodu.*

Ker linearni model velja le za spremembe iz delovne točke (deviacijski model), moramo torej vhodu v linearni model odšteti delovno točko vhoda U_{DT}, dobljenemu izhodu modela pa prišteti vrednost izhoda v delovni točki Y_{DT}.

1.5 Naloge

1. Za Arduino Uno napišite program za utripanje led diode (vgrajena led dioda na pinu 13) s taktom 0,1 s (0,1 s gori in 0,1 s ugasnjena). Program naj



Slika 1.11: *Shema za vrednotenje modela procesa, kjer je vključena informacija o delovni točki (U_{DT}, Y_{DT}).*

vsebuje časovnik implementiran s prekinitveno rutino.

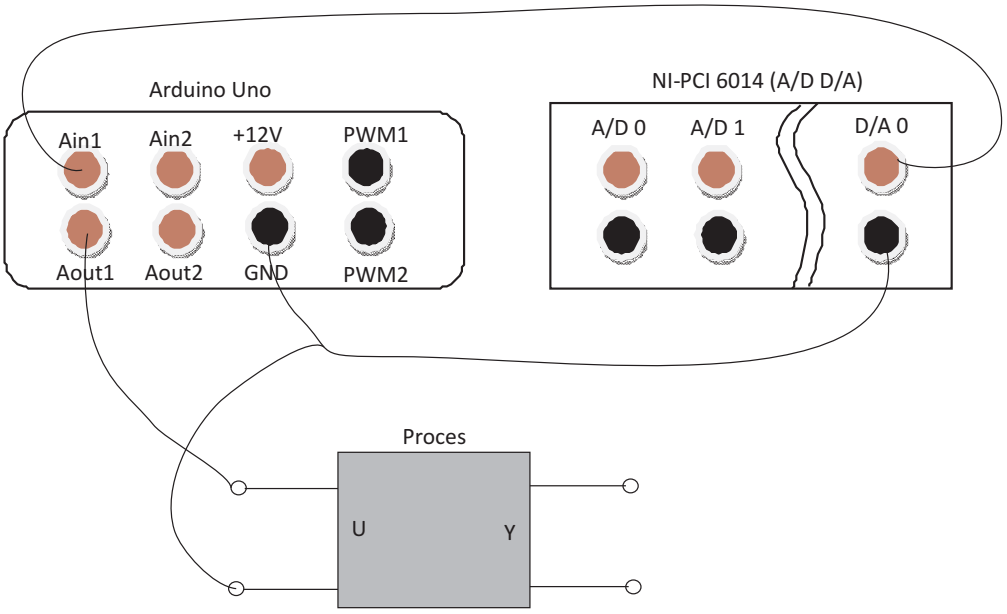
S pomočjo pulznoširinske modulacije PWM ob vsakem taktu časovnika na pinu 9 (ki naj bo PWM izhod) spreminjajte napetost od 0 do 10 V po korakih 0,1 V in nato v obratno smer. Dobljeni izhod Aout1 si lahko ogledate v Simulinku preko bloka NI-PCI 6014.

V pomoč vam je lahko program v poglavju 1.3.

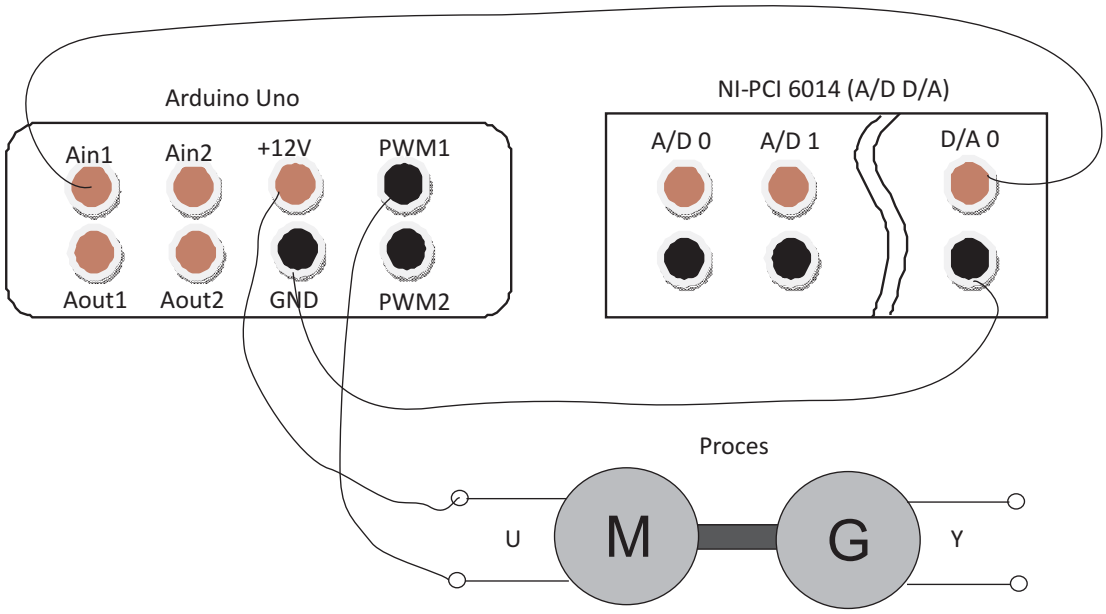
2. S pomočjo okolja Matlab Simulink ocenite prenosno funkcijo $G(s)$ med izhodom in vhodom procesne naprave za izbrano delovno točko. Delavna točka naj bo izbrana približno v sredini področja delovanja naprave. Prenosno funkcijo ocenite s pomočjo odziva na stopnico v delovni točki. Veljavnost modela preverite s simulacijo.

3. Ocenite primeren čas vzorčenja za ocenjen model iz točke 2) in določite diskretno prenosno funkcijo $H(z)$ z uporabo funkcije *c2d* (metoda *zoh*). Na simulaciji primerjajte delovanje dobljenega zveznega in diskretnega modela.

4. S pomočjo Arduino Uno vezja izvedite krmiljenje (odprtozančno) naprave, tako da v okolici delovne točke spreminjate vhod naprave. Vrednost vhoda spreminjate s pomočjo PWM signala na pinu 9 (Aout1 na sliki 1.5). Ukaz za vrednost PWM na pinu 9 pa naj bo napetost na vhodu Ain1 (slika 1.5). Ena od možnosti generiranje analognega signala, ki je vhod na A/D mikrokrmilnika, je uporaba bloka NI-PCI 6014 (glej sliko 1.12 oz. 1.13).



Slika 1.12: *Vezalna shema za proces z analognim vhodom (proces mora imeti močnostni ojačevalnik na vhodu).*



Slika 1.13: *Vezalna shema za proces s PWM vhodom v proces.*

2. Vodenje procesov z digitalnim regulatorjem

Vodenje izbranega procesa, za katerega ste v prejšnji vaji že določili zvezni in diskretni model, bomo izvedli s pomočjo diskretnega PID regulatorja implementiranega na mikrokrmilniškem vezju Arduino Uno.

2.1 PID-regulator

Prenosno funkcija idealnega PID-regulatorja podaja enačba

$$G_{PID}(s) = \frac{U(s)}{E(s)} = K_P \left(1 + \frac{1}{T_I s} + T_D s\right)$$

kjer so K_P proporcionalno ojačenje, T_I integrirna časovna konstanta in T_D časovna konstanta diferenciatorja.

Pri izvedbi PID-regulatorja v praksi se uporablja zakasnjeni D člen, kar dosežemo z uvedbo filtra prvega reda na odvodu

$$G_{PID_{real}}(s) = K_P \left(1 + \frac{1}{T_I s} + \frac{T_D s}{T^* s + 1}\right)$$

kjer je T^* filterska konstanta D člena in jo lahko določimo z $T^* = 0,2T_D$.

2.1.1 Diskretizacija PID regulatorja

Če izvedemo diskretizacijo idealnega PID regulatorja $G_{PID}(s)$ pri času vzorčenja z uporabo metode zadnjih diferenc pri operaciji odvajanja in integracije. Pri izpeljavi vstavimo v $G_{PID}(s)$ izraz $s = \frac{1}{T}\frac{z-1}{z}$, kjer je T čas vzorčenja. Po izpeljavi dobimo diskretni PID-regulator v rekurzivni obliki

$$u(k) = u(k-1) + q_0 e(k) + q_1 e(k-1) + q_2 e(k-2)$$

kjer so parametri

$$\begin{aligned} q_0 &= K_P(1 + \tfrac{T}{T_I} + \tfrac{T_D}{T}) \\ q_1 &= -K_P(1 + 2\tfrac{T_D}{T}) \\ q_2 &= K_P \tfrac{T_D}{T} \end{aligned}$$

Izpeljavo lahko izvedemo v okolju Matlab s sledečo kodo

```
\verbatim{
syms  z Kp Ti Td T
s=1/T*(z-1)/z
Gpid=Kp*(1+1/(Ti*s)+Td*s)     % zvezna prenosna funkcija PID-a
[n,d]=numden(Gpid)
Gpid_z=(collect(n,z)/collect(d,z))
pretty(Gpid_z) }
```

ki nam določi diskretno prenosno funkcijo regulatorja

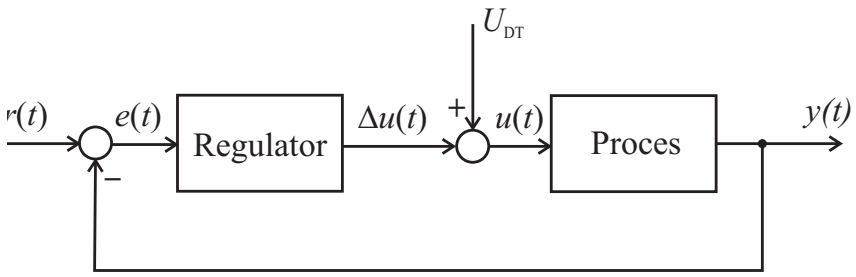
$$G_{PID}(z) = \frac{U(z)}{E(z)} = \frac{K_P\,(T^2 + T_D\,T_I + T_I\,T)\,z^2 - K_P\,(2\,T_D\,T_I + T_I\,T)\,z + K_P\,T_D\,T_I}{T_I\,T z^2 - T_I\,T z}$$

od koder lahko določimo gornjo rekurzivno obliko regulatorja v časovnem prostoru.

2.1.2 Regulacijska shema s kompenzacijo delovne točke

Proces želimo regulirati v okolici delovne točke (U_{DT}, Y_{DT}) z ustreznim regulatorjem, ki zagotavlja sledilno ali regulacijsko delovanje zaprte zanke.

Najenostavnejši linearni regulator je P-regulator, kjer regulirno veličino dobimo tako, da regulacijski pogrešek množimo s konstanto. Glavna pomanjkljivost P-regulatorja je pogrešek v ustaljenem stanju pri konstantni referenci oz. konstantni motnji. To pomanjkljivost lahko vsaj v delovni točki odpravimo, če uporabimo regulacijsko shemo iz slike 2.1. Ko je referenca $r(t) = Y_{DT}$, se proces ustali



Slika 2.1: *Kompenzacija delovne točke v regulacijski zanki.*

v delovni točki, zaradi česar je regulacijski pogrešek $e(t) = 0$, $\Delta u = Ke(t) = 0$ in je zato vhod v proces $u(t) = U_{DT}$.

V kolikor uporabimo PI-regulator, I del poskrbi za izničenje pogreška v ustaljenem stanju saj integrira pogrešek, dokler regulirna veličina ne zagotovi ničnega

pogreška. Čas, ki je potreben, da I del regulatorja odpravi regulacijski pogrešek, je odvisen od integracijske časovne konstante I-regulatorja (T_I). V kolikor uporabimo regulacijsko shemo iz slike 2.1, lahko v okolici delovne točke ta integracijski čas bistveno skrajšamo, s čemer dosežemo hitrejši odziv regulacijske zanke.

2.2 Uglaševanje regulatorja PID z uporabo nastavitvenih pravil

V nadaljevanju so podana nekatera nastavitvena pravila za uglaševanje PID-Regulatorjev in sicer za vodenje procesov prvega reda, prvega reda z mrtvim časom in procesov višjega reda.

2.2.1 Nastavitvena pravila za proporcionalne procese 1. reda

Za vodenje proporcionalnega prvega reda lahko uporabimo P-regulator (v tem primeru moramo predpisati želeno ojačenje zaprtozančnega sistema K_{zel} ali želeno časovno konstanto zaprtozančnega sistema T_{zel}) ali PI-regulator (predpisati moramo T_{zel}, ojačenje zaprtozančnega sistema je namreč vedno enako 1 – zakaj?), medtem ko je T_s časovna konstanta procesa 1. reda, K_s pa njegovo ojačenje.

regulator	K_P	T_I	T_D
P	$\frac{T_s - T_{zel}}{K_s T_{zel}}$ ali $\frac{K_{zel}}{K_s(1-K_{zel})}$	/	/
PI	$\frac{T_s}{T_{zel}K_s}$	T_s	/

Gornja nastavitvena pravila za P-regulator v tabeli lahko izpeljemo, če primerjamo želeno zaprtozančno prenosno funkcijo $G_{zel} = \frac{K_{zel}}{T_{zel}s+1}$ z izračunano zaprtozančno prenosno funkcijo $G_z = \frac{K_P G}{1+K_P G}$, kjer je K_p proporcionalno ojačenje regulatorja in $G = \frac{K_s}{T_s s+1}$ prenosna funkcija procesa. Iskano ojačenje K_P lahko dobimo s primerjavo časovnih konstant G_{zel} in G_z ali s primerjavo ojačenj G_{zel} in G_z. Ker ima P-regulator na P-procesu prvega reda vedno pogrešek v ustaljenem stanju, je smiselna izbira $0 < K_{zel} < 1$.

Nastavitvena pravila za PI-regulator izpeljemo tako, da izberemo želeno zaprtozančno prenosno funkcijo $G_{zel} = \frac{1}{T_{zel}s+1}$ (I-del regulatorja kompenzira

pogrešek v ustaljenem stanju, torej $K_{zel} = 1$). Iz zaprtozančne prenosne funkcije $G_z = \frac{G_{PI}G}{1+G_{PI}G}$ izrazimo $G_{PI} = \frac{G_z}{G-G_zG}$, kjer za G_z vstavimo G_{zel}, in dobimo $G_{PI} = \frac{T_s}{T_{zel}K_s}(1 + \frac{1}{T_s s})$.

2.2.2 Nastavitvena pravila za proporcionalne procese 1. reda z mrtvim časom

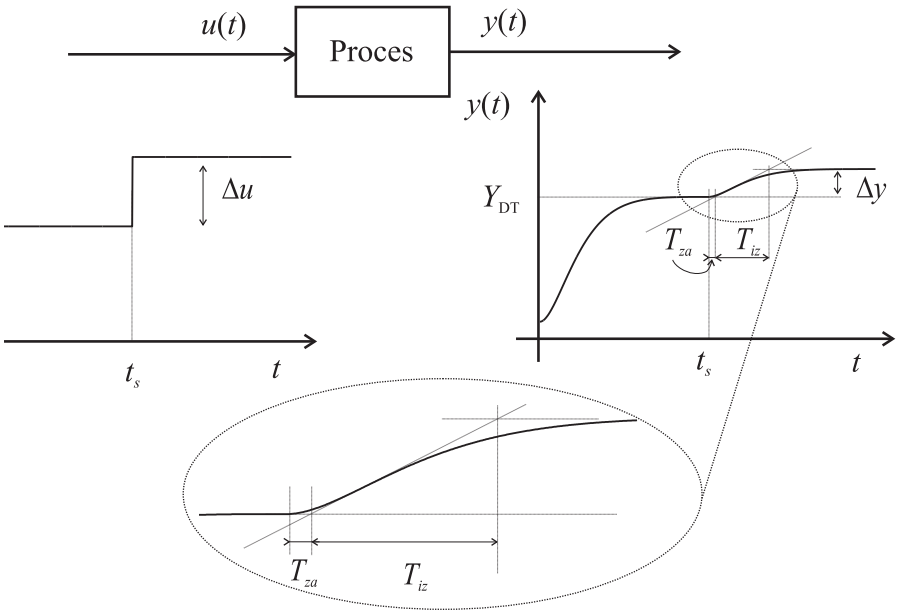
Potrebujemo ojačenje procesa K_s, časovno konstanto T_s in mrtvi čas T_m. Iz nastavitvenih pravil Ziegler-Nichols določimo regulator.

regulator	K_P	T_I	T_D
P	$\frac{T_s}{K_s T_m}$	/	/
PI	0,9 $\frac{T_s}{K_s T_m}$	3,3 T_m	/
PID	1,2 $\frac{T_s}{K_s T_m}$	2 T_m	0,5 T_m

2.2.3 Nastavitvena pravila za P-procese višjega reda (Ziegler-Nichols)

Podobno kot pri naslednji skupini nastavitvenih pravil (Chien-Hrones-Reswick), tudi ta pravila temeljijo na odprtozančnem preizkusu s stopničastim vhodnim signalom. V prevojni točki (za P-procese višjega reda) odziva narišemo tangento in ocenimo čas zakasnitve T_{za} in čas izravnave T_{iz} ter ojačenje procesa K_s (za ilustracijo glejte sliko 2.2). Iz nastavitvenih pravil Ziegler-Nichols določimo regulator.

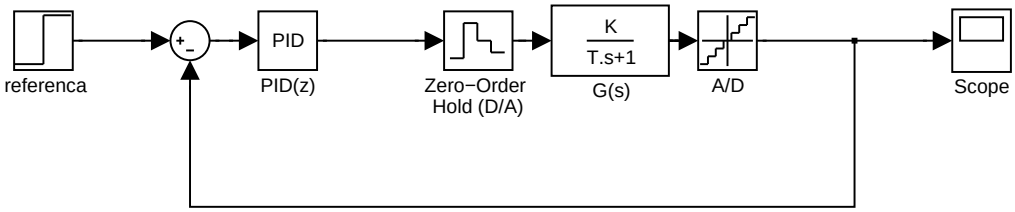
regulator	K_P	T_I	T_D
P	$\frac{T_{iz}}{K_s T_{za}}$	/	/
PI	0,9 $\frac{T_{iz}}{K_s T_{za}}$	3,3 T_{za}	/
PID	1,2 $\frac{T_{iz}}{K_s T_{za}}$	2 T_{za}	0,5 T_{za}



Slika 2.2: *Odprtozančni odziv procesa na stopničasto vzbujanje in ocena parametrov* T_{za}, T_{iz} *in* $K_s = \frac{\Delta y}{\Delta u}$.

2.3 Naloge

1. Za ocenjeni model procesa (iz 1. vaje) v izbrani delovni točki določite parametre zveznega regulatorja PID z uporabo ustreznih nastavitvenih pravil. Delovanje regulatorja preverite s simulacijsko shemo, ki vsebuje zvezni model procesa in diskretni regulator, kot prikazuje slika 2.3.



Slika 2.3: *Izgled simulacijske sheme z zveznim modelom procesa in diskretnim regulatorjem.*

   Če z odzivom niste zadovoljni popravite parametre regulatorja.

2. Dobljeni diskretni regulator preverite tudi na procesu, tako da v simulink shemo namesto modela vstavite blok NI-PCI 6014. Pri izvedbi regulacije upoštevajte delovno točko procesa (slika 2.1).

3. Implementirajte dobljeni regulator na mikrokrmilniškem vezju Arduino Uno in preverite njegovo delovanje. Uporabite prvi A/D kanal (Ain1) za branje signala iz senzorja, izhod regulatorja pa naj bo PWM signal na pinu 9 (Aout1). Najprej preverite regulacijsko delovanje, kjer je referenca nastavljena na delovno točko, regulator pa odpravlja motnje, ki se zgodijo med delovanjem procesa. Motnje lahko simulirate z vplivanjem na proces (npr. previdno z roko nekoliko zadržite vrtenje motorja).

4. Izvedite regulator še za sledilno delovanje, kjer referenca ne bo konstantno nastavljena na delovno točko, ampak jo bo možno spreminjati med delovanjem. Za nastavljanje reference lahko uporabite serijsko komunikacijo - v pomoč vam je lahko program v poglavju 1.3. Referenco lahko nastavljate tudi preko dodatnega analognega vhoda (Ain2) na Arduinu.

3. Načrtovanje regulatorja stanj in observatorja stanj v simulacijskem okolju

Glavni cilj vaje je načrtovanje diskretnega regulatorja stanj za podani zvezni proces

$$G(s) = \frac{4}{(s+1)(s+2)(s+3)}$$

Načrtati želimo regulator stanj. Merimo lahko le izhod procesa, zato moramo stanja procesa oceniti s pomočjo diskretnega observatorja. Regulator (in observator) sta diskretna, kar omogoča enostavno aplikacijo na računalniku oziroma mikroprocesorju.

V nadaljevanju je najprej podano nekaj teorije, ki vam bo poleg literature s predavanj v pomoč pri načrtovanju, nato sledijo naloge.

3.1 Spoznavna kanonična oblika

Pri vaji obravnavamo sistem 3. reda, za katerega bomo tudi izpeljali zapis v prostoru stanj za spoznavnostno kanonično obliko.

Imamo sistem podan v obliki

$$H(z) = \frac{Y(z)}{U(z)} = \frac{b_1 z^2 + b_2 z + b_3}{z^3 + a_1 z^2 + a_2 z + a_3}$$

prenosno funkcijo preuredimo v

$$Y(z^3 + a_1 z^2 + a_2 z + a_3) = U(b_1 z^2 + b_2 z + b_3) = 0$$

nato združimo člene z enako potenco

$$z^3 Y + z^2(a_1 Y - b_1 U) + z(a_2 Y - b_2 U) + (a_3 Y - b_3 U) = 0$$

preuredimo, da imamo na levi najvišjo predikcijo izhoda

$$z^3 Y = -z^2(a_1 Y - b_1 U) - z(a_2 Y - b_2 U) - (a_3 Y - b_3 U)$$

izraz delimo z najvišjo potenco z

$$Y = -\frac{1}{z}(a_1Y - b_1U) - \frac{1}{z^2}(a_2Y - b_2U) - \frac{1}{z^3}(a_3Y - b_3U)$$

ter zapišemo v vgnezdeno obliko

$$Y = \frac{1}{z}\left(b_1U - a_1Y + \frac{1}{z}\left(b_2U - a_2Y + \frac{1}{z}\left(b_3U - a_3Y\right)\right)\right)$$

vpeljemo stanja in določimo izhod sistema

$$\begin{array}{l} X_1 = \frac{1}{z}\left(b_3U - a_3Y\right) \\ X_2 = \frac{1}{z}\left(b_2U - a_2Y + \frac{1}{z}\left(b_3U - a_3Y\right)\right) = \frac{1}{z}\left(b_2U - a_2Y + X_1\right) \\ X_3 = \frac{1}{z}\left(b_1U - a_1Y + \frac{1}{z}\left(b_2U - a_2Y + \frac{1}{z}\left(b_3U - a_3Y\right)\right)\right) = \frac{1}{z}\left(b_1U - a_1Y + X_2\right) \\ Y = X_3 \end{array}$$

zapišemo v časovnem prostoru

$$\begin{array}{l} y(k) = x_3(k) \\ x_1(k+1) = b_3u(k) - a_3x_3(k) \\ x_2(k+1) = b_2u(k) - a_2x_3(k) + x_1(k) \\ x_3(k+1) = b_1u(k) - a_1x_3(k) + x_2(k) \end{array}$$

dobljeni končni izraz zapišemo v prostoru stanj, ki predstavlja končni zapis v spoznavni kanonični obliki

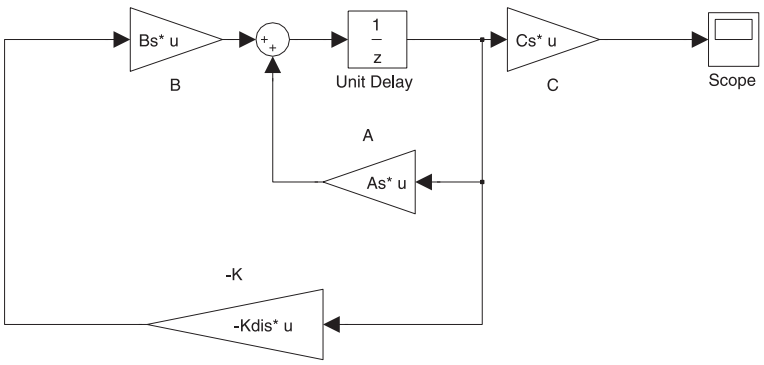
$$\begin{bmatrix} x_1(k+1) \\ x_2(k+1) \\ x_3(k+1) \end{bmatrix} = \begin{bmatrix} 0 & 0 & -a_3 \\ 1 & 0 & -a_2 \\ 0 & 1 & -a_1 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \\ x_3(k) \end{bmatrix} + \begin{bmatrix} b_3 \\ b_2 \\ b_1 \end{bmatrix} u(k)$$

$$y(k) = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \\ x_3(k) \end{bmatrix}$$

3.2 Regulator stanj

3.2.1 Regulator stanj za regulacijsko delovanje

Regulator stanj določi potreben vhod v sistem, tako da se vsa stanja sistema ustalijo na vrednosti 0 (delovna točka sistema). Regulator odpravlja motnje v sistemu. Možna izvedba regulatorja stanj v okolju Simulink je prikazana na sliki 3.1

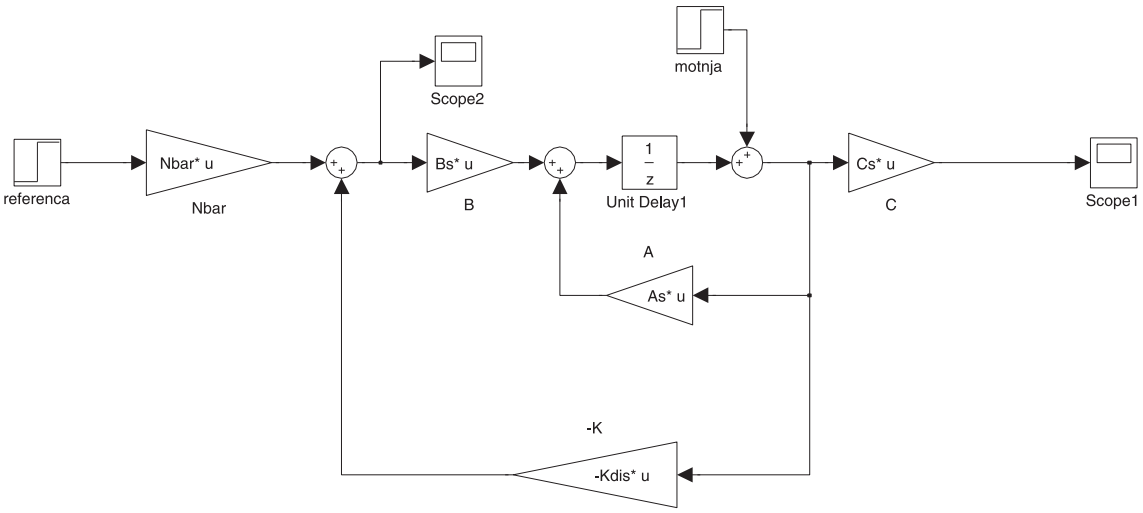


Slika 3.1: Regulator stanj za regulacijsko delovanje

3.2.2 Regulator stanj za sledilno delovanje

Regulacijska shema s predkrmilnim ojačenjem

Da doseženo sledilno delovanje (izhod procesa sledi referenci) je potrebno nadgraditi osnovno regulacijsko shemo procesa. Osnovo izvedbo prikazuje slika 3.2 Na vhod pripeljemo tudi referenco, ki je pomnožena z $\overline{N}$. V kolikor je $\overline{N} = 1$ ima



Slika 3.2: Regulator stanj s predkrmilnim ojačenjem

sistem velik pogrešek v ustaljenem stanju. V nadaljevanju je prikazan pristop za izračun $\overline{N}$.

Ustrezen vhod v proces pri konstantni referenci bi bil

$$u(k) = -K\left(x(k) - x_{ss}\right) + u_{ss}$$

kjer sta u_{ss} in x_{ss} ustaljene vrednosti vhoda in stanj pri referenci r. Določimo $x_{ss} = N_x r$ in $u_{ss} = N_u r$ in izpeljemo

$$\begin{aligned} u(k) &= -K\left(x(k) - N_x r\right) + N_u r \\ &= -Kx(k) + (KN_x + N_u)r(k) \\ &= -Kx(k) + \overline{N}r(k) \end{aligned}$$

Za ustaljeno vrednost velja

$$\begin{aligned} x_{ss} &= Ax_{ss} + Bu_{ss} \\ y_{ss} &= Cx_{ss} \\ r &= y_{ss} \end{aligned}$$

kar preuredimo v

$$\begin{aligned} N_x r &= AN_x r + BN_u r \\ r &= CN_x r \end{aligned}$$

sistem enačb lahko zapišemo v matrični obliki

$$\begin{bmatrix} (A-I) & B \\ C & 0 \end{bmatrix} \begin{bmatrix} N_x \\ N_u \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

in izračunamo

$$\begin{bmatrix} N_x \\ N_u \end{bmatrix} = \begin{bmatrix} (A-I) & B \\ C & 0 \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

in iskan skalar

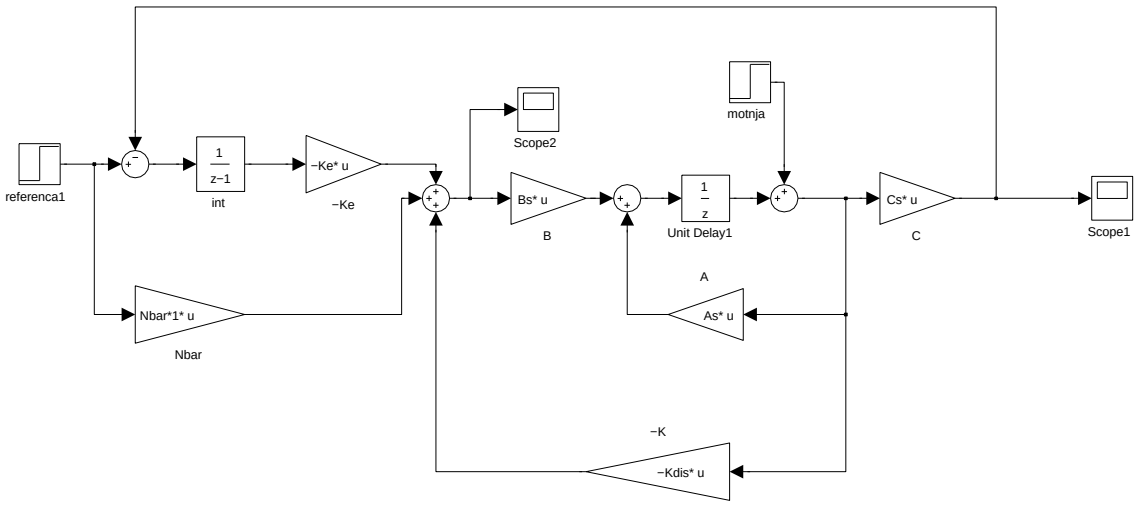
$$\overline{N} = KN_x + N_u$$

ki predstavlja potrebno ojačenje reference, da dosežemo ustaljeno stanje.

Regulacijska shema z uvedbo integrirnega značaja

Shema s predkrmilnim ojačenjem deluje dobro, v kolikor je model točen. Če model ne opisuje dobro procesa oziroma v primeru nastopa motenj v procesu bomo dobili pogrešek v ustaljenem stanju. Če shemo vodenja razširimo z dodatnim stanjem, ki predstavlja vsoto pogreška (trenutne in preteke vrednosti), potem to stanje vpelje integrirni značaj in odpravi vpliv netočnega modela oziroma motenj in omogoča sledenje brez pogreška v ustaljenem stanju. Razširjena shema je prikazana na sliki 3.3. Kot dodatni vhod v proces se pojavi še vsota pogreškov $\sum_k r(k) - y(k)$ pomnoženo z ojačenjem K_e. V nadaljevanju podamo izpeljavo za določitev regulatorja stanj. Integrirni značaj torej vpeljemo s stanjem, ki predstavlja rekurzivno vsoto pogreška

$$\begin{aligned} x_e(k+1) &= x_e(k) + (r(k) - y(k)) \\ &= x_e(k) + r(k) - Cx(k) \\ &= x_e(k) - Cx(k) + r(k) \end{aligned}$$



Slika 3.3: Regulator stanj z uvedbo integrirnega značaja

s tem dodatnim stanjem razširimo sistem in dobimo nov sistem

$$\begin{bmatrix} x(k+1) \\ x_e(k+1) \end{bmatrix} = \begin{bmatrix} A & \underline{0} \\ -C & 1 \end{bmatrix} \begin{bmatrix} x(k) \\ x_e(k) \end{bmatrix} + \begin{bmatrix} B \\ 0 \end{bmatrix} u(k) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} r(k)$$

$$y(k+1) = \begin{bmatrix} C & 0 \end{bmatrix} \begin{bmatrix} x(k) \\ x_e(k) \end{bmatrix}$$

kar zapišemo kompaktno kot

$$\begin{aligned} x_N(k+1) &= A_N x_N(k) + B_{Nu} u(k) + B_{Nr} r(k) \\ y(k+1) &= C_N x_N(k) \end{aligned}$$

in vhod v proces

$$u(k) = -K_N x_N(k) = -\begin{bmatrix} K & K_e \end{bmatrix} \begin{bmatrix} x(k) \\ x_e(k) \end{bmatrix}$$

zaprtozančni poli so določeni s karakterističnim polinomom zaprte zanke (kjer vzamemo $r = 0$)

$$\det\left(zI - (A_N - B_N K_N)\right) = 0$$

Regulator stanj za razširjen sistem torej lahko določimo z Ackermanovo metodo (ukaz *acker* ali *place*), kjer predpišemo želene pole zaprte zanke. Regulator lahko določimo tudi z LQR metodo (ukaz *lqr*). Za določitev regulatorja potrebujemo matriki A_N in B_{Nu} ter želene zaprtozančne pole.

Nadalje regulator adaptiramo še s predkrmilnim ojačenjem,

$$u(k) = -K_N x_N(k) + \overline{N} r(k)$$

kot je prikazano na sliki.

3.3 Observator stanj

Observator stanj določimo tako, da bodo realni deli polov 2-3 krat hitrejši, kot so zaprtozančni poli sistema (z regulatorjem). Observator načrtamo le za osnovni proces, torej brez dodatnega stanja $x_e(k)$. Pri načrtovanju uporabimo funkcijo *place*, ki je v osnovi namenjena za določitev regulatorja stanj, zato je potrebna določena sprememba kot sledi.

Sistemska matrika z regulatorjem stanj je $A-BK$, sistemska matrika observatorja pa $A-LC$. Če transponiramo sistemsko matriko za regulator stanj dobimo $A^T-C^TL^T$, kar je podobno regulatorju stanj. Z ukazom *place* določimo matriko ojačenj L observatorja z ukazom

```
L=place(A',C',zeleni_poli)';}
```

observator lahko določite tudi z ukazom *lqe* ali *kalman*, kjer postopek načrtovanja zahteva določitev matrik uteži za določeno kriterijsko funkcijo (glej dokumentacijo omenjenih ukazov). Te matrike se lahko podajo kot kovariančne matrike šuma procesa (na stanjih) in šuma izhoda sistema (meritve).

3.4 Naloge

1. Za nek proces imamo ocenjen zvezni linearni model $G(s) = \frac{4}{(s+1)(s+2)(s+3)}$, ki je veljaven v neki delovni točki. Določite primeren čas vzorčenja procesa in določite diskretni model procesa (funkcija *c2d*).

2. Diskretni model procesa podajte v prostoru stanj v spoznavni kanonični obliki. Preverite vodljivost in spoznavnost sistema.

3. Določite regulator stanj za regulacijski problem (odpravljanje motenj) za zapis sistema v spoznavni kanonični obliki. Predpostavite, da imate vsa diskretna stanja merljiva. Regulator stanj lahko določite z ukazom *place* ali z ukazom *lqr*. Gre za dva različna pristopa načrtovanja.

   - **Z ukazom *place*:** Želena dinamika zaprtozančnega sistema je podana z dušenjem $\zeta = 0.8$ in lastno frekvenco $\omega_n = 4$ ter za realni zaprtozančno pol $s = -5$. Poli v prostoru s torej so

     $$s_{1,2} = -\zeta\omega_n \pm i\omega_n\sqrt{1-\zeta^2}$$
     $$s_3 = -5$$

     Pole v zveznem prostoru pretvorite v diskreten prostor z (uporabite relacijo $z = e^{sT}$). Regulator stanj načrtajte z ukazom *place*.

- **Z ukazom *lqr*:** Pri tem postopku moramo podati utežitveni matriki Q in R za kriterijsko funkcijo optimizacije $J = \sum \left(x^T Q x + u^T R u\right)$. Vrednosti matrik določimo glede na to, katero stanje oziroma vhod naj ima večji vpliv pri regulaciji. Primer možne izbire matrik je

  ```
  Q=[1 0 0;0 1 0; 0 0 1]*200; R=1;
  ```

Preverite delovanje regulatorja stanj na diskretnem modelu procesa. Regulator preizkusimo za začetne vrednosti stanj, ki so različna od želene `[0 0 0]` (torej smo izven želene delovne točke sistema).

4. Razširite dobljeno regulacijsko shemo iz točke 3) v regulator stanj za sledilno delovanje s prekrmilnim ojačenjem. Delovanje preverite na stopničasti referenci in ob nastopu motnje na stanju, ki predstavlja izhodni signal.

5. Razširite dobljeno regulacijsko shemo iz točke 4) v regulator stanj za sledilno delovanje z integrirnim značajem. Želeni poli zaprte zanke so lahko enaki tistim iz 3. naloge, katerim dodate še pol pri $z = 0,8$ za dodatno stanje (integral pogreška). Delovanje preverite na stopničasti referenci in ob nastopu motnje na stanju, ki predstavlja izhodni signal.

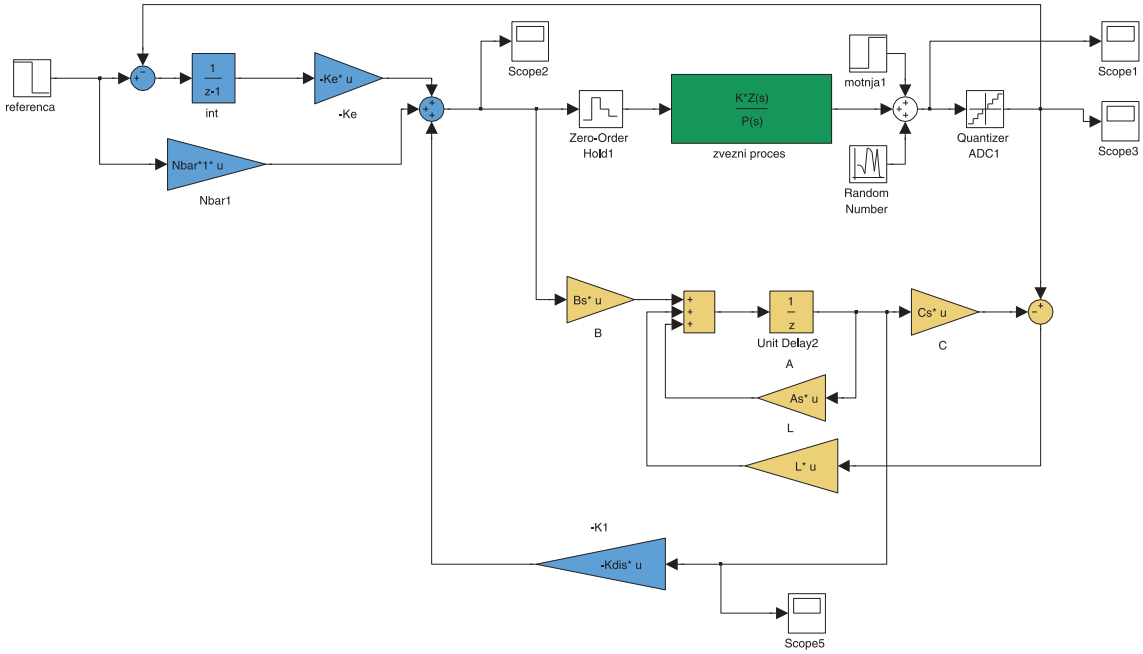
6. Določite observator stanj, ki bo ocenil diskretna stanja sistema (brez dodatnega stanja, ki predstavlja integral pogreška) in naredite regulacijsko shemo ob realnih predpostavkah:

   - simulirani proces je zvezen in ima dostopen le vhod $u(k)$ in izhod y(k),
   - vodenje procesa izvedemo z diskretnim regulatorjem stanj,
   - stanja procesa niso merljiva, zato jih ocenimo s pomočjo observatorja stanj.

   Želena dinamika observatorja je podana s poli v zveznem prostoru s kodo

   ```
   ceta=.95; wn=8;
   poli_obs=[-ceta*wn-i*wn*sqrt(1-ceta^2);
             -ceta*wn+i*wn*sqrt(1-ceta^2);
             -10]
   ```

   ki jih morate pretvoriti v diskretni prostor z. Celotno shemo vodenja prikazuje slika 3.4



Slika 3.4: Celotna shema vodenja

4. Načrtovanje regulatorja stanj in observatorja stanj v mikrokrmilniškem okolju

Vodenje izbranega procesa, ki ste ga spoznali že v prejšnjih vajah (1. in 2. vaja) bomo izvedli s pomočjo diskretnega regulatorja stanj z observatorjem v prostoru stanj. Postopek načrtovanja je podoben, kot v 3. vaji, le da imate ocenjene prenosne funkcije prvega reda oziroma nekatere skupine drugega reda. Observator načrtamo tudi v primeru, da imamo le eno stanje, ki je merljivo. Observator nam v tem primeru služi kot filter (na osnovi modela) in nam zmanjša šum meritve.

4.1 Naloge

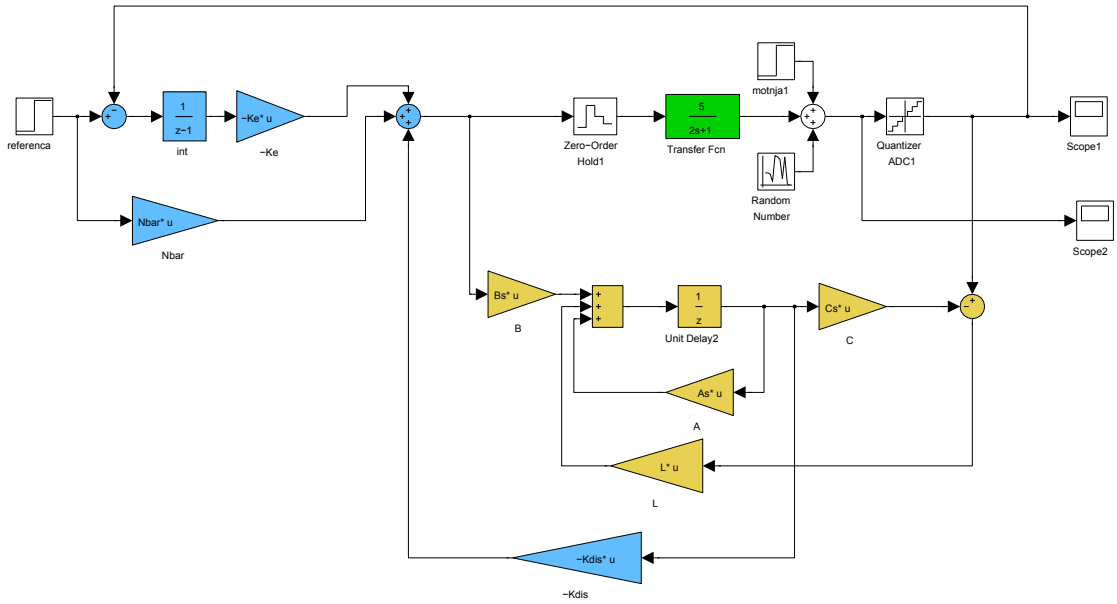
1. Za napravo v izbrani delovni točki določite diskreten model v prostoru stanj, tako da bo eno od stanj tudi izhod procesa.

2. Določite regulator stanj za regulacijski problem (odpravljanje motenj).

   Želene pole zaprte zanke v Laplaceovem prostoru določite tako, da bo zaprtozančni sistem deloval dvakrat hitreje (ali več), kot odprta zanka. Realni deli zaprtozančnih polov naj bodo torej dvakrat večji. V primeru, da imate sistem drugega reda pa lahko imajo poli tudi nekaj imaginarne komponente, tako da bo dušenje $\zeta = 0,9$. Dobljene želene pole pretvorite v diskreten prostor in z ukazom *place* določite regulator stanj.

3. Izračunajte še observator stanj, katerega poli (realni deli polov v prostoru s) naj bodo 2 krat hitrejši kot poli zaprte zanke z regulatorjem (želeni poli). Observator določite z ukazom *place*.

4. Dopolnite regulator stanj iz prejšnje točke tako, da bo primeren za sledilni način delovanja. Dopolnitev izvedite tako, da bo regulator imel integrirni značaj (uvedba dodatnega stanja, glej poglavje 3). Želeni poli naj bodo enaki kot v nalogi 2 za dodatno stanje (integral pogreška) pa dodajte še želeni pol $z = 0,8$. V pomoč vam je lahko sledeča koda



Slika 4.1: *Izgled simulacijske sheme z zveznim modelom procesa in diskretnim regulatorjem z observatorjem.*

```
% uvedba dodatnega stanja za integrirni značaj (glej poglavje 3)
% sistem razširimo z dodatnim stanjem
% xn(k+1)=An*xn+Bnu*u(k)+Bnr*r(k)
% yn(k+1)=Cn*xn(k)
n=1;  % število stanj procesa
An=[As zeros(n,1);
    -Cs   1];
Bnu=[Bs;0];
Bnr=[zeros(n,1); 1];
Cn=[Cs 0];

% želeni poli zaprte zanke so poli od prej ter še za integrator (z=1),
% ki ga prestavimo na z=0.8
poli_dis2=[poli_dis, 0.8];
Kdis2=place(An,Bnu,poli_dis2);
Kdis=Kdis2(1);
Ke=Kdis2(2);

% uvedba predkrmilnega ojačenja na referenci
 [NN]=[(As-eye(n)) Bs ; Cs     0]^-1*[zeros(n,1);1];
 Nx=NN(1:n);
 Nu=NN(n+1);
 Nbar=Kdis*Nx+Nu;
```

Delovanje regulatorja stanj z observatorjem preverite na simulaciji v okolju Simulink, kot prikazuje slika 4.1.

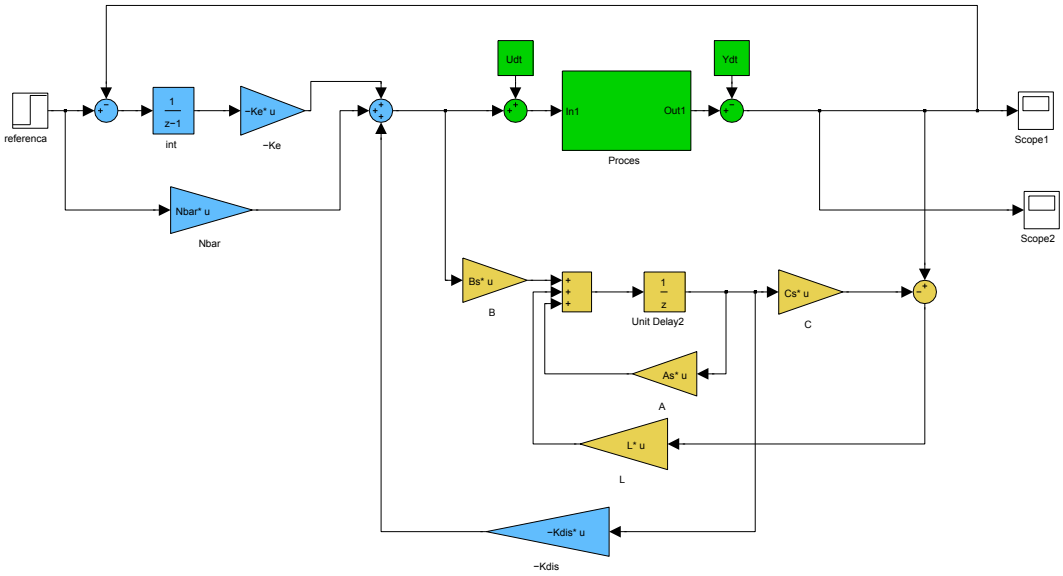
5. Regulator iz naloge 4 implementirajte na Arduino Uno in preverite njegovo delovanje. Bodite pozorni na ustrezno kompenzacijo delovne točke (glej sliko 4.2). Referenco (odstopanje od delovne točke izhoda) lahko po želji definirate kot konstantno vrednost ali pa jo posredujete preko serijske povezave ali preko dodatnega analognega vhoda (drugi A/D kanal).

   Enačbe za izračun vhoda v proces in izračun ocenjenih stanj observatorja lahko glede na sliko 4.2 zapišemo kot

$$u(k) = -Kx(k) - K_e x_e(k) + \overline{N}r(k)$$

$$x_e(k+1) = x_e(k) + (r(k) - y(k))$$
$$x(k+1) = (A - LC)x(k) + Bu(k) + Ly(k)$$

   kjer so $x(k)$ ocenjena stanja diskretnega modela procesa, $x_e(k)$ dodatno stanje, ki predstavlja rekurzivno vsoto pogreškov (integrirni značaj), $y(k)$ deviacija izhoda procesa ($y(k) = y_{proces}(k) - Y_{DT}$) in $u(k)$ izračunan deviacijski vhod v proces ($u_{proces}(k) = u(k) + U_{DT}$).



Slika 4.2: *Princip kompenzacije delovne točke procesa. Regulator in observator sta načrtana le za linearni model (deviacijski model), ki opisuje le odstopanja od izbrane delovne točke. Na sliki referenca predstavlja odstopanje od delovne točke izhoda.*