Ipari mintavételes PID szabályozóstruktúra megvalósítása

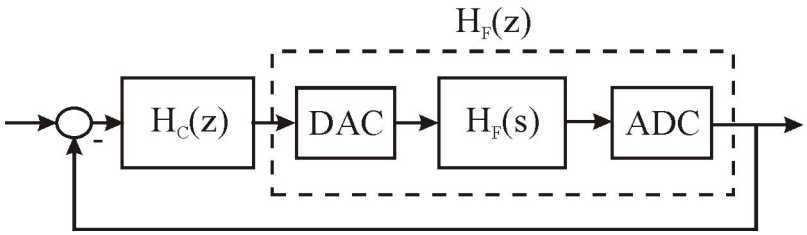
1. A gyakorlat célja

Készítsen diszkrét PID szabályozót megvalósító programot C++, obiektumorientált környezetben. Teszteléssel igazolja a szabályozó működését.

2. Elméleti bevezető

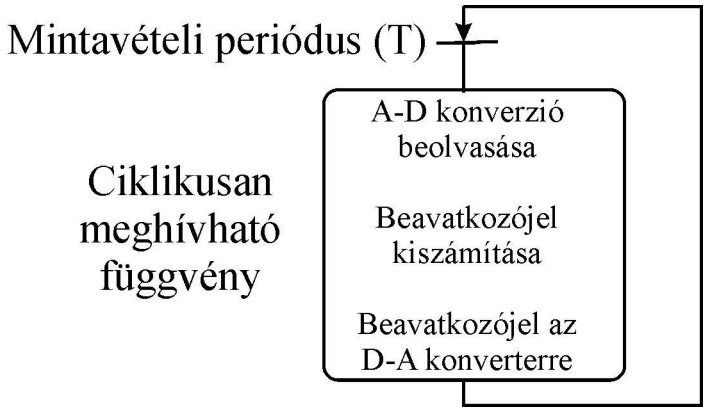
Mintavételes szabályozások megvalósítása

Az irányítási algoritmusok megvalósításának legelterjedtebb módja a szoftveres implementálás. Az irányítási algoritmust megvalósító függvények, eljárások általában egy nagyobb, ipari folyamatok felügyeletére, irányítására, monitorizálására kialakított szoftver része. Az irányítási algoritmus a folyamat mért kimenete és az előírt kimenet (alapjel) alapján számítja ki a beavatkozó jelet. A folyamat kimenetét analóg-digitális átalakítón keresztül olvassuk be. Az alapjelet ugyancsak analóg-digitális átalakítón keresztül olvashatjuk be vagy megadhatjuk számítógépes periférián keresztül is (például billentyűzet segítségével). A második esetben nincs szükség analog-digitális átalakításra az alapjel esetében. A kiszámolt beavatkozó jelet digitális-analóg átalakítón keresztül küldjük a beavatkozó bemenetére.



1 Ábra: Mintavételes szabályozási rendszer kialakítása

A mintavételes megvalósításnál szükségünk van egy ciklikusan (konstans periódusonként) meghívható függvényre. A konstans periódust *mintavételi periódus*nak nevezzük. Az ipari szoftver főszála ciklikusan megszakítódik és végrehajtódik az irányítási algoritmust megvalósító eljárás. Az eljárás főbb részei a következők:
- a folyamat bemenetének beolvasása analóg-digitális átalakítóról.
- a beavatkozó jel kiszámítása az alapjel és a folyamat bemenetének függvényében.
- a beavatkozó jel kiküldése a digitális-analóg átalakítón keresztül.

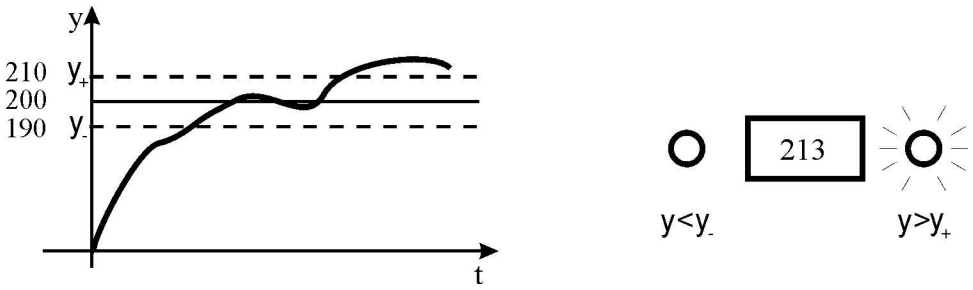


2 Ábra: Mintavételes szabályozás megvalósítása

A mintavételes kialakítás felvet néhány speciális problémát: az analóg-digitális, digitális-analóg átalakítás miatti kerekítési hibák, a mért jelek digitális szűrése, kalibrációja, Shannon tétel, a mintavételi periódus meghatározása.

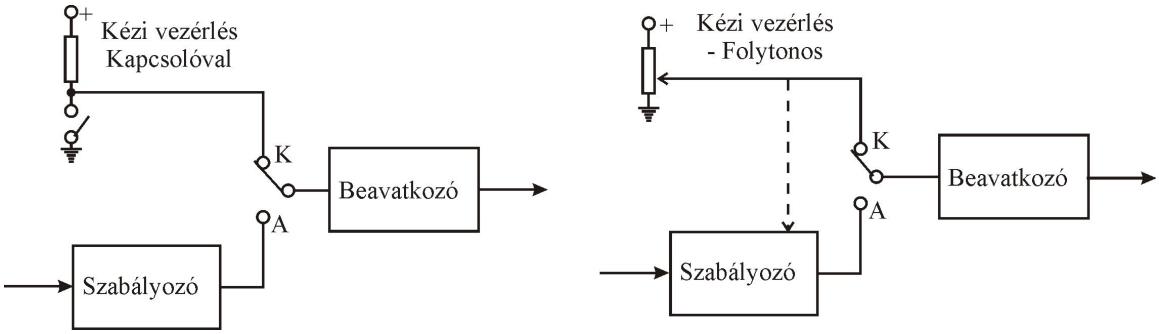
Kézi szabályozás

Kézi szabályozás esetében az ipari folyamatot kezelő, felügyelő személy végezi el az irányítási feladatot. Ebben az esetben a felügyelő személynek egyrészt információja kell, hogy legyen a folyamat kimenetéről, másrészt lehetősége kell legyen arra, hogy a beavatkozó bemenetét módosítsa. Ebben az esetben is megvalósul a visszacsatolás, csak a kezelőszemélyen keresztül.
Az ipari folyamatok esetén mindig meg kell hagyni a lehetőséget a kézi szabályozásra is, arra az esetre, hogy ha a folyamatot irányító számítógépben szoftver- vagy hardverhiba lép fel.
A folyamat kimenetét általában numerikusan jelezzük ki, a kimenet értékét SI mértékegységekben kifejezve. Lassú folyamatok esetén hasznos a grafikus kijelzés is (a folyamat kimenetének ábrázolása idő függvényében). Célszerű még figyelmeztető kijelzéseket is alkalmazni: például, ha a folyamat kimenete meghalad egy felső határt, vagy kilép egy előre meghatározott toleranciasávból, ami már veszélyes lehet a szabályozókör helyes működésére, akusztikus jelzést vagy fényjelzést ad az irányítást megvalósító eszköz (lásd 3 Ábra).



3 Ábra: A szabályozott jellemző (irányított folyamat kimenetének) kijelzése

A kezelőszemélyzet beavatkozása:
A folyamatot felügyelő személynek mindig lehetősége kell legyen arra, hogy vészhelyzet esetén leállítsa a szabályozást. Ebben az esetben az irányítási algoritmus inaktívvá válik, a beavatkozó bemenetére nullaértékű jel kerül.
Ha a kezelőszemélyzet irányítani és nem megállítani kívánja a folyamatot, lehetősége kell legyen a beavatkozó bemenetének folytonos módosítására. Ez történhet egy kis energiájú folytonos jellel. Másik lehetőség, hogy kapcsoló üzemmódú beavatkozást végzünk, ha a beavatkozó bemenetére lehet nemfolytonos jelet adni. (Például egy ellenállással fűtött kemencében az ellenállásra rákapcsoljuk a feszültséget, ha növelni szeretnénk a hőmérsékletet vagy lekapcsoljuk róla a feszültséget, ha azt szeretnénk, hogy csökkenjen a kemence hőmérséklete.) A két megvalósítás elvi rajza a 4 Ábrán látható.
A kezelő személy által végzett szabályozás (Kézi (*K*) szabályozás) és a szabályozó elem által végzett szabályozás (Automata (*A*) szabályozás) közötti átkapcsolást kétállású kapcsolóval oldhatjuk meg. Problémát jelenthet, hogy az átkapcsoláskor ugrások jelennek meg a vezérlőjelben. A kézi üzemmódban a kezelő személy a beavatkozó bemenetére adott feszültségszint és a szabályozó által meghatározott feszültségszint különbözhet. Ez az átkapcsolás pillanatában a beavatkozó bemenetén a jel ugrásszerű megváltozásában nyilvánul meg, ami akár a beavatkozó károsodásához is vezethet. Ennek elkerülésére a Kézi - Automata átkapcsoláskor azt a stratégiát alkalmazhatjuk, hogy a szabályozó visszaolvassa a kézi üzemmódban kiadott beavatkozó jelet és átkapcsoláskor a szabályozó ettől a kiinduló értéktől, módosítja tovább folyamatosan, az irányítási feladatnak megfelelően.



4 Ábra: Lehetőségek kézi szabályozás megvalósítására

3. A *mérés menete*

Adott a Quantum PLC PID blokkját leíró dokumentáció (*Quantum_PID_Appendix_L7.pdf*). Az elkészített program az abban leírt algoritmust kell, hogy megvalósítsa C++ nyelven.

A feladat részletezése:
- Építsük fel a dokumentációban szereplő struktúrákat (MODE_PID, PARA_PID, STAT_MAXMIN) *struct* változókat, vagy külön osztályokat alkalmazva.

- Implementáljuk a proporcionális, deriváló, integráló komponenseket a mellékletben szereplő képletek szerint.
- Implementáljuk a PID szabályozó kapcsolási logikáját (*manual mode, halt mode, automatic mode*) valamint a szabályozóstruktúra típusának kiválasztását (P, PI, I, PD, PID) a mellékletben szereplő blokkrajz és leírás alapján.
- Implementáljuk az integráló anti-windup kiegészítését.

Tesztelés: Adott bemeneti adatokra kell a vezérlőjelet előállítani. Mintavételek szekvenciáját *'for'* ciklussal valósítsuk meg. Lehetséges tesztesetek:

I teszteset:
Bementi adatok:

```
Mode_PID.man = false
Mode_PID.halt = false
Mode_PID.en_p = true
Mode_PID.en_i = false
Mode_PID.en_d = true
Mode_PID.d_on_pv = false

Para_PID.gain = 1
Para_PID.ti = 1
Para_PID.td = 1
Para_PID.td_lag = 1
Para_PID.ymax = 100
Para_PID.ymin = -100

FEED_FWD = 0
YMAN = 3

SP_vect = {1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1}
PV_vect = {0, 0.1, 0.2, 0.3, 0.4, 0.3, 0.5, 0.6, 0.9, 1, 1, 1, 1, 1, 1}
```

Kimeneti adatok: az Y-t tartalmazó vektor (`Y_vect`)

II teszteset:
Bementi adatok:

```
Mode_PID.man = false
Mode_PID.halt = false
Mode_PID.en_p = true
Mode_PID.en_i = true
Mode_PID.en_d = true
Mode_PID.en_d_on_pv = true

Para_PID.gain = 10
Para_PID.ti = 3
Para_PID.td = 0.1
Para_PID.td_lag = 0.01
Para_PID.ymax = 15
Para_PID.ymin = -15

FEED_FWD = 0.01
YMAN = 3

SP_vect = {1, 1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 1.7, 1.8, 1.9, 2, 2.1, 2.2,
2.3, 2.4};
```

```
PV_vect = {0, 0.5, 0.9, 0.9, 0.9, 1.1, 1.2, 1.5, 1.75, 1.9, 2.05, 2.2,
2.2, 2.3, 2.4};
```

4. Kérdések és feladatok

1. Keressük meg a Quantum PID dokumentációban, hogyan van megoldva, hogy az a helytelen integrálási idő (zéró érték) megválasztása ne vezessen számítási hibához.
2. Miért szükséges az integráló tag anti-windup kiegészítése?
3. A programban az integráló tag numerikus megvalósításánál trapéz módszert alkalmaztunk. Módosítsuk úgy a programot, hogy az integrálás téglalap módszerrel legyen megvalósítva.