Féléves terv – Irányítástechnika tárgy

1.Feladat megfogalmazása:

Készítsen diszkrét PID szabályozót megvalósító programot. Teszteléssel igazolja a szabályozó működését és dokumentálja a programot. Bemutatandó a feladatot megoldó kód, a terv dokumentációja.
A feladatot mindenki egyénileg kell, hogy megoldja! A terv végső leadási határideje: a szemeszter utolsó Irányítástechnika laboratóriuma. Az Irányítástechnika vizsga feltétele a terv megoldása.

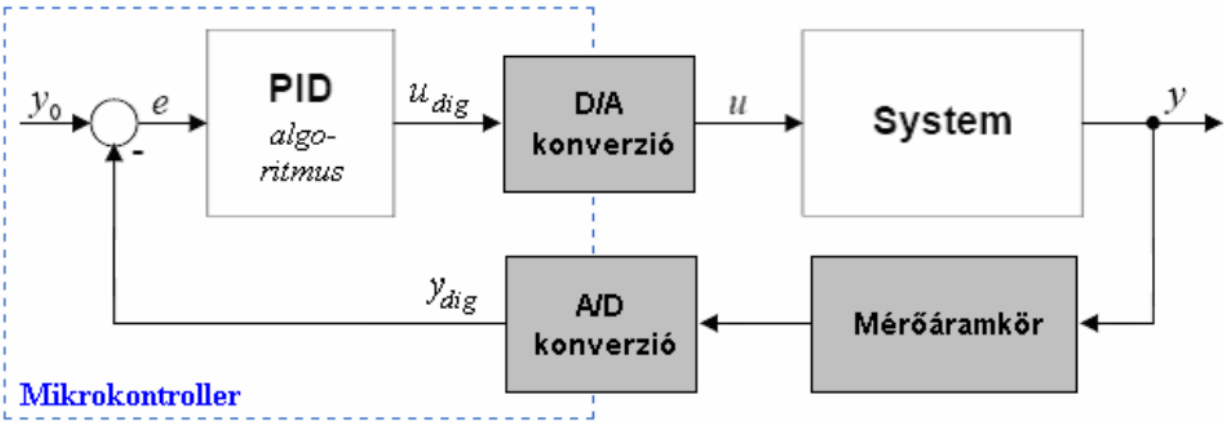
2.Mintavételes PID algoritmus:

A projekt keretén belül egy olyan diszkrét szabályozó tömböt valósítottam meg, amely választható működési funkcióval és több üzemmódban képes egy adott referencia jelnek megfelelően a kimenetet úgy szabályozni, hogy ezáltal elvárt eredményhez jussunk. Ennek alapját egy diszkrét szabályozó algoritmus képezi, amelyet a legismertebb szabályozó elemekkel kapcsoltam össze, így valósítottam meg egy diszkrét PID szabályozót és az ezt alkotó elemek kombinációjából összetett más szabályozókat. Fontos azt megjegyeznem, hogy a szabályozónk több üzemmódban is működhet, mint például automata üzemmód, manuális üzemmód, kaszkád szabályozó elvén alapuló üzemmód, amelyek közül jelenlegi feladatunkban egy PD és PID szabályozót használtunk fel automata üzemmódban.

3.A diszkrét PID szabályozó

Napjainkban az irányítási algoritmusok zömét digitális számítógépekkel valósítják meg. Az ilyen irányítási algoritmusok esetében a véges mintavételi periódus miatt a folyamat kimenetére csak fix mintavételekben van információnk. A másik fontos jellemzőjük, hogy két mintavétel között az irányítási algoritmus által kiszámított vezérlőjel nem változik. A diszkrét PID szabályozó alapelve a szabályozott jellemző mért y értékének az előző y alap jellel történő összehasonlításán alapszik. Az e két mennyiség különbségeként kép ezt hibajelből (e – error) ezután kiszámítható az új beavatkozó jel (u) értéke oly módon, hogy annak hatására a szabályozott mennyiség értéke közelebb kerüljön az alapjel értékéhez. Ezen folyamatnak a hossza a mintavételezési időtartam hosszától függ.
Abban az esetben, ha a valóságban szeretnénk felhasználni a leprogramozott PID szabályzónkat, akkor elkerülhetetlen gondolnunk a folytonos-diszkrét, diszkrét-folytonos átalakításokra, mivel a rendszerek többsége folytonos jelet fogadnak és ugyancsak folytonos jelet is szolgáltatnak.

A diszkrét PID algoritmusunkban a szabályozó minden tagjának megjelenik egy aktuális és egy előző értéke, amelyben az előző érték minden mintavételi periódus végén felveszi az addigi aktuális értéket. A PID szabályozó által szolgáltatott kimenetet érdemes lekorlátozni, mivel az I és a D tag adott esetekben túlcsordulást okozhat.



3. ábra – Digitális PID szabályozási séma

4.A Program leírása:

Ahogy már fentebb is említettem, a projektet C++ nyelvben valósítottam meg, mivel ez a nyelv könnyedén lehetővé teszi bárkinek, aki élete első szabályozó algoritmusát szeretné egy mikrokontrolleren kipróbálni, hogy egy Arduinóra feltöltve elvégezzen egy szabályozási feladatot egy iparilag is elismert rendszerrel.
Ahhoz, hogy a megirt kód könnyebben olvasható legyen, strukturális felépítés használtam. Így elsősorban létrehoztam három struktúrát, amelyek a bemeneti adatokat tartalmazzák.

A *ControllerType* struktúra tartalmazza azt a változót, amelyet a hozzá kapcsolt függvény segítségével változtatva a főprogramban alkalmazhatjuk azt a szabályozó típust, amelyet a feladat kér.

Az *OperatingMode* struktúrában találhatóak az üzemmódok, amelyekben alkalmazhatjuk a kiválasztott szabályozónkat.

A *Parameters* struktúra tartalmazza a bemeneti adatainkat.
Ezt követően megtalálható a *SelectOperatingMode* amely felelős az üzemmód kiválasztásáért.
Az ezt követő függvények segítségével a szabályozási feladatot részfeladatokra bontottam.

• A *calculateError* függvény az aktuális hiba kiszámításáért felel.
• A *calculateYP* függvény segítségével a P tagot számolom ki.
• A calcYIauto visszatéríti az I tag értékét automata üzemmódban.
• A calcYImanualhalt ugyancsak az I értékét adja meg manuális vagy halt üzemmódban.
• A *calcYDautocascade* függvényen belül van egy elágazás, amely a *d_on_pv* változó függvényében eldönthetjük, hogy a D tag értékét a folyamatértékek felhasználásával számítsuk ki, vagy pedig az aktuális és előző hiba különbségének felhasználásával kapjuk meg az eredményt kaszkád vagy automata üzemmódban.

A főp rogramban az első két sorban létrehozzuk az sp (*SetPoint* - referencia) és a pv(ProcessValue – folyamat érték) tömböt.
Ezután néhány változó példányosítása után az üzemmód kiválasztó funkcióval rendelkező függvény meghívása következik, amelyben paraméterként 1-es és 0-os segítségével a dokumentációban leírt szabály alapján kiválaszthatjuk azt az üzemmódot, amelyet szeretnénk. Mivel a feladatunkat két bemeneti adat állomány segítségével szeretnénk kipróbálni, azért létrehoztam egy változót, amely segítségével kiválaszthatjuk az aktuális feladat elvégzéséhez szükséges rendszerkonfigurációkat és bemeneti adatokat. Ez a *tesztNr* változó. A fent említett adatok beírása után, létrehozzuk a programban azokat a változókat, amelyek az előző értékeket fogják tárolni, mivel ez elengedhetetlen egy diszkrét szabályozó algoritmus megvalósítása esetében. Elérkeztünk ahhoz a részhez, ahol a megírt függvényeket és bemeneti adatokat felhasználva el tudjuk végezni a szabályzást. A ciklikus számítást egy *for* ciklussal valósítottam meg, amely annyi lépést végez el ahány mintavételünk van a referencia tömbben. Minden ciklus elején számolok egy hibát az aktuális referencia és a folyamat érték között. A szabályozó típus kiválasztását elvégző függvény értékének feldolgozását egy *switch-case* útválasztóval végzem el, amely egy 3 számjegyű számot kap, ahol a pozíció szerint a PID kifejezésre levetítve az 1-es érték jelöli azt, hogy a pozíciónak megfelelő szabályozó tag részét kép ezi-e a kiválasztott szabályozónak, vagy pedig 0 esetén nincs benne az adott tag. Így összesen négy féle szabályozó típust különítettem el, amelyek a következők: P (100), PI (110), PD (101), PID (111). A *case*-eken belül még megtalálhatóak az üzemmód kiválasztást végrehajtó *if*-ek. Mielőtt még az eredményt kiíratnám a kép ernyőre, elvégzem a régi értékek felülírását, azáltal, hogy az eddigi aktuális értékeket örökölik azok a változók, amelyek eddig az egy ciklussal ezelőtti értékeket tárolták. Ami még hátravan az Y eredmény kiszámolása, amely egyben a szabályozó kimenetét is jelenti. Ezt az értéket lekorlátozom, azért, hogy be tudjak állítani egy minimális és maximális értéket, a biztonságos vezérlés érdekében. Ha mindezzel megvagyok jelen esetben már csak a kiírtatás van hátra, amelyet követően a ciklust folytatva elvégzem az összes számítást.

Pál János-Attila

Automatizálás III B