24. Vezérlés és szabályozás

Írta: Herángli Gergő, Apró Csaba, Trádler Máté, Ujvári Barbara

Lektorálta: Szabó Ádám

BEVEZETÉS

Az irányítás, az irányítani akarás egy elemi emberi tulajdonság. A történelem folyamán számos esetben megnyilvánul ez az emberi jellemző. Ennek oka, hogy egy tárgyat vagy egy folyamatot akkor érzünk igazán saját tulajdonunknak, ha azt irányítani is tudjuk. Azt mondhatjuk tehát, hogy az irányításelmélet fejlődésének, sőt tágabb értelemben a tudomány fejlődésének egyik mozgatórugója a birtoklási vágy. Természetesen további számos aspektusa van ennek a fejlődésnek, azt sem tagadhatjuk, hogy a mérnöki fejlesztések egy jelentős részéért a kevésbé magasztos lustaság a felelős, vegyük például a távirányító esetét.

Egy irányítási kör megvalósítása alapvetően egy leírási, egy tervezési, végül egy implementációs feladatból áll.

Leírás alatt azt értjük, hogy képesek vagyunk egy olyan modellt létrehozni, amely kellő pontossággal visszaadja nekünk a valóságban lévő folyamat jellemzőit. Ennek végeredménye egy matematikai összefüggés, például az Ohm törvény. Folyamat alatt, mérhető fizikai mennyiséget értünk. Vegyünk például egy olyan motort, amely az előbbi tananyagrészekben már részletesen tárgyalásra került (egyenáramú motor). A motornál a folyamat lehet például a forgórész nyomatéka, szögsebessége, pozíciója. Ezek mindegyike mérhető vagy származtatható mennyiség. A nyomaték a motor áramából, a szögsebesség és a pozíció pedig valamilyen enkóder felhasználásával határozható meg a legkönnyebben. Azok a jelenségek amelyek nem mérhetők, nem tekinthetők folyamatnak.

Tervezés alatt azt értjük, hogy előállítunk egy olyan algoritmust vagy más néven programot, amely az általunk fontosnak tartott folyamatot az előírt értéken tartja. Ha szeretnénk egy fúrófej fordulatszámát 1000 RPM (Revolutions Per Minute) értéken tartani, akkor azt várjuk ettől az algoritmustól, hogy a fúró motorjának fordulatszáma pontosan 1000 RPM legyen (feltéve, hogy nincs áttétel). Ez az egyik legnehezebb feladat, amely során számos kompromisszumot kell kötnünk. Például ha a forgórész éppen nem 1000 RPM-el forog, hanem csak 100 RPM-el, vagy esetleg álló helyzetben van, akkor nem várhatjuk el hogy, azonnal 1000-es fordulattal forogjon. (Ehhez végtelen nagy energia kellene.)

Implementáció alatt pedig azt a műveletet értjük, amikor az algoritmust egy a mikroprocesszor számára végrehajtható programkód formájában állítjuk elő.

Előfordulhat, hogy az iménti mondatok ködösnek tűnnek az olvasó számára, ám tananyagunk következő bekezdéseiben példákon keresztül igyekszünk világossá tenni az előbbi fogalmakat. E tananyagrésznek nem célja, hogy átfogó tudást adjon, hanem egy egyszerű, reményeink szerint érthető, és használható tudást adjon az irányítástechnikáról.

A következő részekben szeretnék néhány fogalmat tisztázni, szeretném pontosabban kifejteni mit jelent a rendszerleírás, majd végezetül szeretnék egy egyszerű, gyakran alkalmazott irányítástechnikai módszert bemutatni.

IRÁNYÍTÁSI KÖRÖK

A bevezetőben felületesen átfutottunk a legfőbb fogalmakon, amelyeket feltétlenül be kellett vezetni. Most kicsit rendszerezettebb formában bemutatjuk az irányítási köröket. Elsőként tisztáznunk kell milyen elemek vesznek részt egy irányítási körben.

Van egy folyamat, amit irányítani szeretnénk. Emellett van egy szilárd elhatározásunk, hogy milyen értéken szeretnénk tartani a folyamatot, ez az alapjel. Az alapjelre érdemes úgy gondolnunk, hogy nagyjából bármilyen értéket felvehet, ezért kell egy alapjel formálási blokk, amely nem engedi meg, hogy tetszőlegesen nagy értéket kérjünk az irányítástól, illetve általában ez felelős azért is, hogy mi lehet az alapjel maximális változási sebessége.

Kell lennie valamilyen beavatkozó szervnek is a körben, amely abban segít, hogy a folyamatot befolyásolni tudjuk. Erre azért van szükség, mert az irányítási algoritmusok általában egy mikrokontrolleren futnak, és mint az előző tananyagrészekből is kiderült, a mikrokontrollerek 3,3V vagy 5V-os perifériákkal rendelkeznek. Nyilvánvaló, hogy egy ilyen kimenet nem képes meghajtani például egy villanymotort. A beavatkozó szerv feladata, hogy átkonvertálja a mikrokontroller “gyenge” jeleit nagyszintű “erős” jelekké. A gyakorlatban ez valamilyen elektronikus jelerősítő fokozat általában, de ha például egy robbanásveszélyes térben kell irányítanunk egy folyamatot, akkor lehet, hogy beavatkozó szervként pneumatikus aktuátorokat kell használnunk. Magától értetődő, hogy egy robbanásveszélyes környezetben egy egyszerű mezei relét nem használhatunk kapcsolóelemként, hisz a relé nyitásakor létrejövő szikra tüzet -az pedig robbanást- okozhat.

Az esetek 99%-ában a rendszerben a vezérlést/szabályzást végző egység egy mikrokontroller. Ennek feladata, hogy az irányítási algoritmust futtassa, azaz eldöntse, hogy a bemenete alapján milyen beavatkozó jelet állítson elő a beavatkozó szervnek. Számos előnnyel jár a kontrollerek használata, hiszen könnyen újrakonfigurálhatók, akár futás közben is módosíthatunk minden paramétert. További előny, hogy számítási kapacitásuk növekedésével és a perifériáik számának és komplexitásának fejlődésével, csaknem minden algoritmus futtatható ezeken a rendszereken.

Felmerülhet a kérdés, hogy miért csak 99%. Léteznek analóg irányítási körök, amelyek irányító berendezése nem programozható, illetve a rendszerben előforduló jelek nincsenek digitalizálva. Ezek jelenléte főként speciális területekre korlátozódik. Ha szükség van a nagyon nagy gyorsaságra, akkor még mindig analóg szabályozókat használnak. Például a fázis zárt hurkok (PLL) esetén.

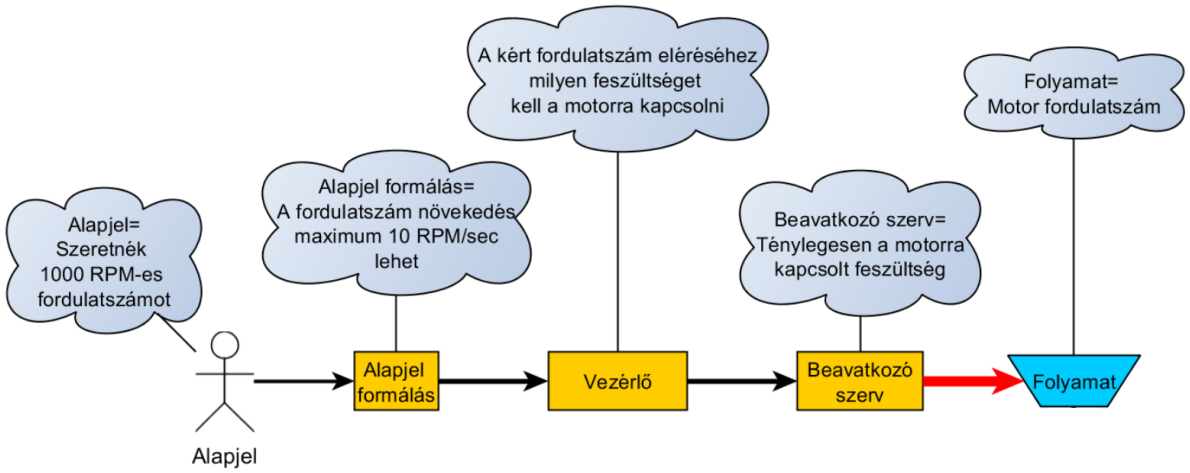
Mivel a mérnökök jó része vizuális típusú ember, így az irányítástechnika területén is számos rajzot, illusztrációt használnak az értelmezés segítésére. Az irányítástechnikai elemeket és azok logikai összekapcsolását tartalmazó ábrát hatásvázlatnak nevezzük, amely hozzájárul a működés és az egyes blokkok funkciójának megértésében.

Működési elv tekintetében a hagyományos irányítás fogalma alapvetően két fő részre bontható. Az egyik a vezérlés, a másik a szabályozás.

VEZÉRLÉS

Elsőként kezdjük az egyszerűbb vezérlés fogalmával. Egy irányítást akkor nevezünk **vezérlés**nek, ha a végrehajtás során a bemenő jel végigfut a hatásláncon, a vezérlés egyes szervein, majd a vezérlő kimenete működteti a beavatkozó szervet, amely befolyásolja a vezérelt berendezés folyamatát. Tehát a hatáslánc nyitott (angolul open-loop control-nak hívják), azaz a vezérlő nem ismeri a beavatkozás hatását, mivel nincs érzékelő a folyamat kimenetén (vagy ha van is, akkor sem veszi figyelembe az irányítás alatt). Azonkívül különböző forrásokból érkező zavaró jelek is befolyásolhatják a folyamatot, amik hatását előre kell ismernünk, ahhoz, hogy kiküszöbölhetőek legyenek.

A következő ábrán egy általános vezérlési hatásvázlatot láthatunk.



***1. ábra** - Vezérlést szemléltető példa*

Az ábrán feltüntettük sárga színnel azokat a blokkokat, amelyekről az előző részben szó volt. Felhőkben jelöltünk egy időpillanatot, amit a rendszer “tapasztal”.

Vegyük sorra a példában szereplő logikai blokkoknak megfelelő tényleges elemeket!

A hatásvázlat elején ott vagyunk Mi, a kezünkben egy változtatható fordulatszámú fúrógéppel és szeretnénk a motor fordulatszámát 1000 fordulat/perc értékűre beállítani. Valószínűleg a fúrógép indítógombján található tekerőt az 1000 RPM feliratú rovátkához fogjuk tekerni. Minden bizonnyal ez mögött egy potenciométer található, amely jelét a hatásvázlat további elemei fel fogják használni. Ezzel be is állítottuk az alapjelünket a rendszer számára.

Az alapjel formálás lehet például az, hogy a potenciométer maximális kitérései olyan értékeket reprezentálnak, amelyek nem engedik a rendszert a biztonságos tartományokon kívülre.

A vezérlő például egy mikrokontrolleren futó algoritmus, ami az alapjel formálás után érkező jelet felhasználja arra, hogy -a memóriában tárolt adatoknak megfelelően- olyan jelet állítson elő a kontroller kimenetén, amely megfelel a motor 1000-es fordulatszámának.

Míg a beavatkozó szerv egy olyan teljesítményelektronikai kapcsolás, mely képes meghajtani a motort.

Minden bizonnyal a fúrógép indítógombján beállítható érték a névleges terheléshez van méretezve, így meglehet, hogy üresjárásban (azaz terhelés nélkül) a tokmány fordulatszáma magasabb lesz a beállított értékhez képest.

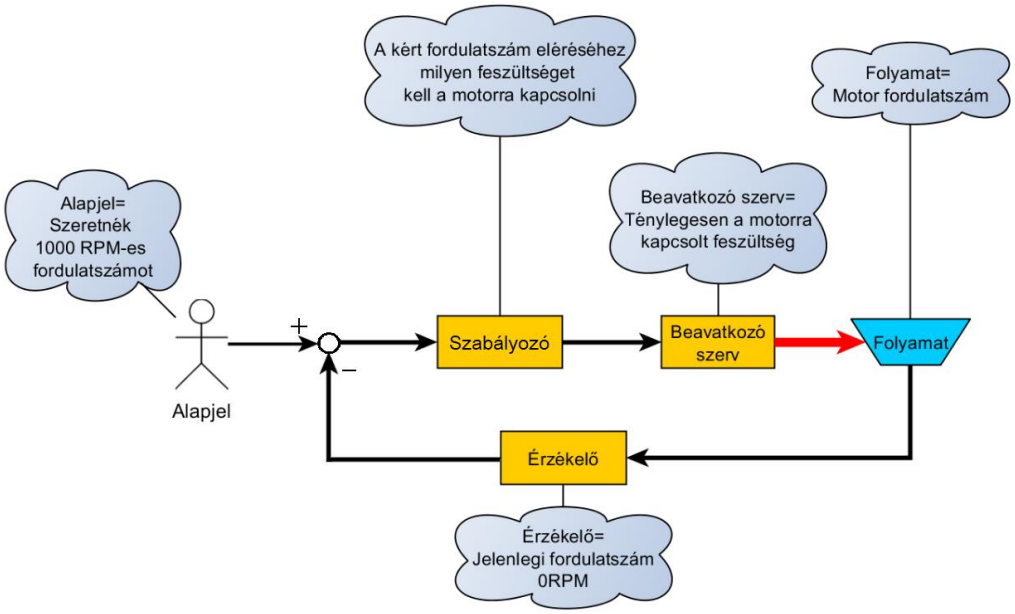
Abban az esetben ha nem rendeltetésnek megfelelően használjuk a fúrógépet, nevezetesen lefogjuk a tokmányát amennyire csak tudjuk, akkor például ez lehet egy olyan zavaró jel, aminek a hatását nem tudjuk kiküszöbölni, mivel nem ismertük előre. Ennél fogva az irányító egység, azaz a vezérlő logika nem fog nagyobb beavatkozó jelet kérni a beavatkozó szervtől, így motor fordulatszáma biztosan nem lesz 1000 RPM.

Általánosságban elmondható, hogy a vezérlést tartalmazó eszközök egyszerűbbek, ugyanis nem tartalmaznak érzékelőket, így azok megválasztása, beépítése a rendszerbe, illetve jeleik illesztése az irányító berendezéshez nem szükséges. Mindamellett egy vezérlő logika egyszerűbb lehet egy komplex, gyors szabályzást megvalósító logikához képest.

SZABÁLYOZÁS

A vezérléssel szemben a **szabályozás** során a folyamat kimenetét folyamatosan méri egy érzékelő szerv, aminek a jelét az alapjelből kivonva, előáll az a különbségi jel (a hibajel), ami a szabályzó bemenete lesz. Tehát a hatáslánc zárt (angolul closed-loop control-nak hívják), azaz a szabályzó ismeri a beavatkozás hatását. Ezenfelül olyan algoritmust kell futtatnia, hogy az alapjel vagy a környezetből érkező zavaró jelek változására, a beavatkozás csökkentse a hibajelet, hiszen a feladata, hogy a szabályozott jellemző az alapjel értékével megegyezzen.

A következő ábrán a motor fordulatszámos példánk szabályozási köre látható.



***2. ábra** - Szabályozást szemléltető példa*

A hatásvázlat kiegészíthető a különbségképző, jelerősítő blokkokkal és a környezetből érkező zavarójellel, de az egyszerűség és átláthatóság miatt ezeket nem, vagy csak egyszerű szimbólummal rajzoltam fel.

A szabályozástechnika az ipari irányítás talán legjelentősebb része, mivel a legtöbb ipari folyamatirányító rendszer szabályozást tartalmaz.

Például egy atomerőműben nem élhetünk a vezérlés egyszerűségével. Gondoljunk csak bele, ha nem győződne meg az irányító berendezés arról, hogy a reakciót lassító grafitrudak a megfelelő pozícióban vannak, akkor meglehet, hogy csak túl későn vennénk észre ezt a hibát. Tehát mindenképpen egy önműködő szabályozásra van szükség, amely a megadott alapjelnek megfelelően (például 10MW teljesítmény), a szabályozó -a lehető legnagyobb pontossággal- a folyamat kimenetét 10MW-ra szabályozza.

PID szabályozó

Szabályozási feladatokra napjainkban előszeretettel használják a PID (Proporcionális Integráló Deriváló) szabályzót. Egy szabályozási körben a szabályzó paramétereinek meghatározása, behangolása nem egyszerű feladat, de miután a nekünk megfelelő paramétereket megtaláltuk, már nagyon egyszerűen meg is tudjuk azt valósítani például egy mikrokontroller segítségével. A PID szabályzónkat a programunkban csupán egyszerű szorzások, összeadások és kivonások fogják alkotni. De mit is csinál a PID szabályzó? A szabályzó reagál az aktuálisan jelenlévő, a múltban történt, illetve a jövőben fellépő hibára is. Ezt a nevében szereplő három tag segítségével tudja elérni. Az első betű a P jelöli a proporcionális tagot, ami magyarul arányos tagnak neveznek. Ez fogja a mért hibának megfelelő beavatkozó jelet kiszámítani, ami a hibajel konstanssal való szorzását foglalja magában. Az I az integráló tagot jelenti, ezáltal veszi figyelembe a múltban fellépő hibákat a szabályzó. A hiba integrálódik, ez leegyszerűsítve azt jelenti, hogy az aktuális hibajel hozzáadódik az addig mért hibajelek összegéhez. Végül a D, a deriváló tag következik, amely a hibajel meredekségét, egyszerűbben a múltbeli és az aktuális hibajel különbségét, ezáltal a jövőbeli alakulását hivatott megadni. Ebből a három értékből együttesen fogja a szabályzó előállítani a beavatkozó jelet. A PID egy viszonylag egyszerű szabályzó típus, ennél bonyolultabbak szabályzókat is használnak a gyakorlatban.

SZABÁLYOZÁS ÉS VEZÉRLÉS ÖSSZEHASONLÍTÁSA

A vezérlés és szabályozás rövid ismertetése után felmerülhet bennünk a kérdés: A tananyag eddigi részeiben látott és megvalósított mikrovezérlős projektek melyik csoportba tartoznak? Irányítástechnikai szempontból nézve ezek egytől egyig vezérlési feladatok voltak. Hiszen az ismertetett szenzorok (pl.: LDR, NTC ellenállás) és a beavatkozó perifériák (pl.: piezo, ventilátor) külön-külön voltak használva, vagy együtt használatuk során sem volt zárt kör kialakítva.

Azonban, ha egy irányítási folyamat alapjelét egy szenzorról vesszük (pl.: analóg hőmérséklet szenzor), és ezt a jelet felhasználjuk az irányító eszközben arra, hogy egy perifériát működtessünk (pl.: ventilátor), akkor még nem biztos, hogy szabályozásról van szó.

Példának okáért, ha egy szobában egy irányító berendezés egy ventilátort képes bekapcsolni a konyha egy adott hőmérséklete fölött, akkor ez egy vezérlési feladat, abban az esetben ha a két légtér egymástól független, pontosabban nem képesek egymással hőt cserélni. Azonban, ha a ventilátor a másik helyiségből hidegebb levegőt képes áramoltatni, akkor szabályozásról van szó.

Amennyiben a ventilátort egy zárt szobában működteti a berendezés, akkor szintén szabályozásról beszélünk, viszont rendkívül rossz konstrukcióval. Ugyanis a ventilátor hatására (levegő áramoltatás) a bőrünk többet tud párologtatni, azaz hűvösebbnek érezzük a környezetet, viszont a hőmérséklete biztosan nem csökken (sőt inkább kis mértékben növekedhet, mert a tekercsek disszipált teljesítménye hő formájában távozik).

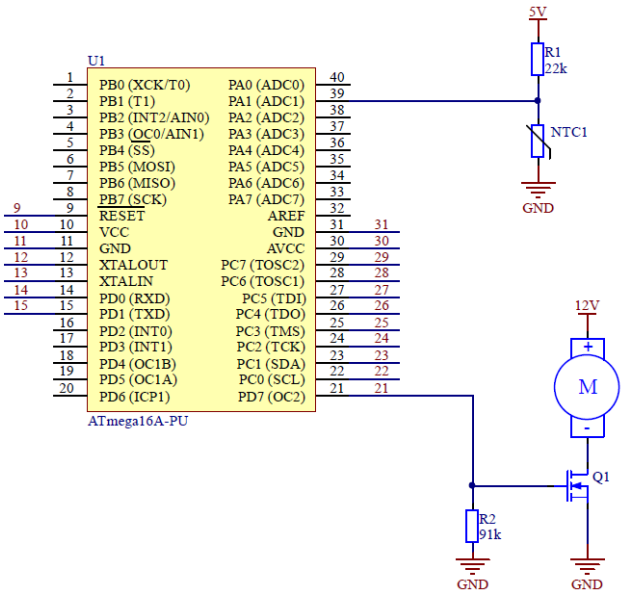
Egy másik eset például a tekerőgombos fényerőszabályzó esete, amely a köznyelvben helytelenül terjedt el. Hiszen tudjuk, hogy általában nincsen semmiféle fényérzékelő elhelyezve a falban vagy a lámpában, amivel visszamérhetné az eszköz a megvilágítás intenzitását, így valójában ez is egy vezérlési, nem pedig szabályozási folyamat. Itt a tekerő gomb alatt rejlő potenciométer egy teljesítményelektronikai eszköz alapjelét állítja, amellyel a lámpa átlag árama változtatható. Hogyha készítünk egy olyan eszközt, amely képes vezérelni -adott változtatható tartományban- a lámpát meghajtó teljesítményelektronikát, illetve rendelkezik valamilyen fényérzékelővel, amit a berendezés figyelembe vesz az algoritmus futása során, és egy előre megadható fényintenzitást képes előállítani, akkor szabályozásról beszélünk.

KONFIGURÁLHATÓ VEZÉRLÉS

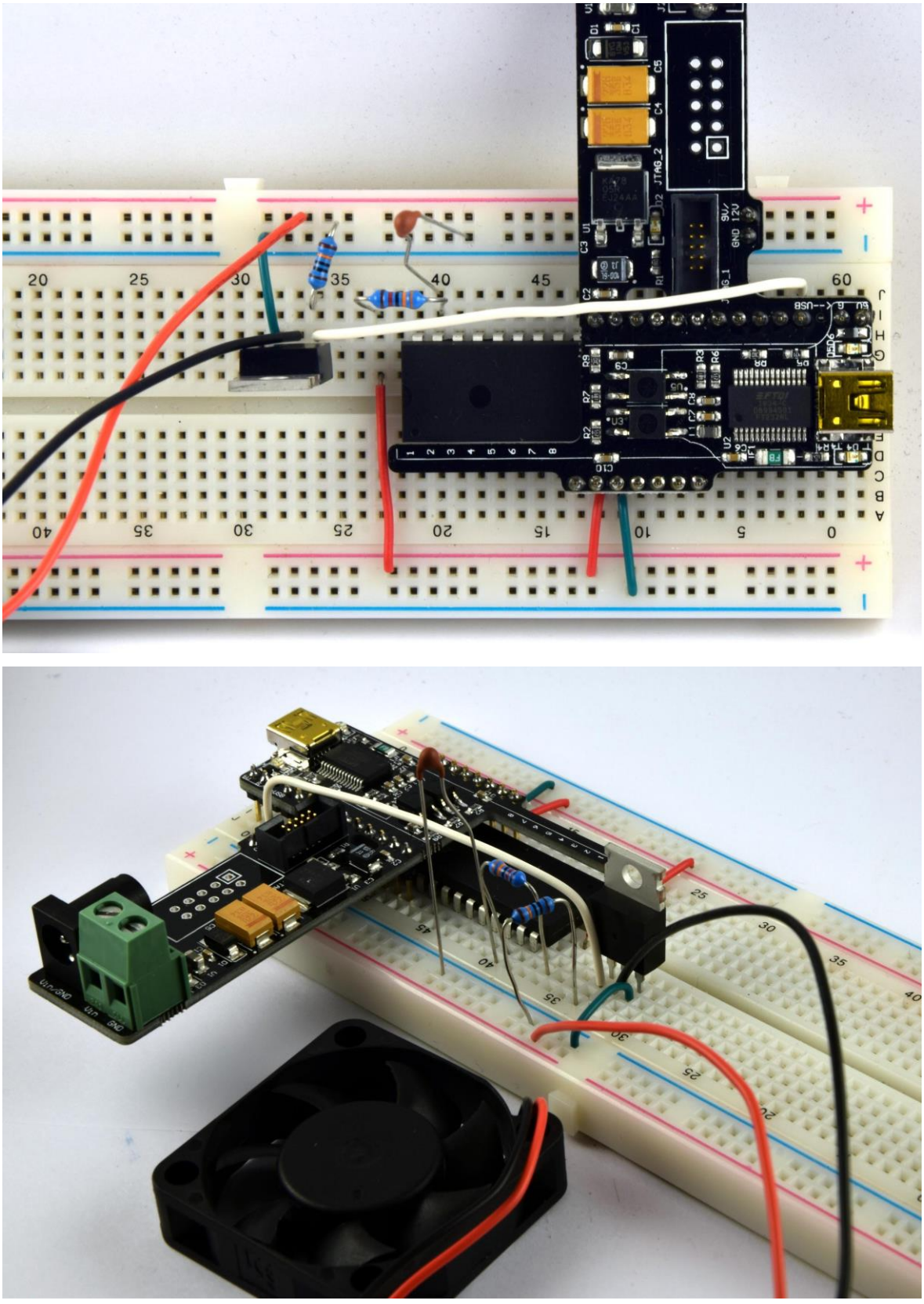
A mostani példa projekt - a korábbi feladatok nagy részéhez hasonlóan - egy vezérlős feladat lesz. Azonban most egy UART-on keresztül konfigurálható vezérlést fogunk megvalósítani.

A ventilátorunkat a hőmérséklettől függően szeretnénk ki/bekapcsolni. A mikrovezérlő ADC perifériájának a segítségével egy NTC ellenálláson eső feszültséget fogunk mérni. A feszültségosztó feszültségeiből az NTC ellenállását, majd abból a hőmérsékletet fogjuk kiszámolni. Soros porton fogadjuk majd, hogy milyen hőmérséklet érték felett kell bekapcsolni a ventilátort.

Építsük meg a következő kapcsolást és nyissuk meg a honlapon található *KE24_1_Konfiguralhato_vezerles* projektet. A korábbi tananyagrészekben már megismerkedtünk az összes ehhez szükséges perifériával, így ezek beállításának magyarázatát most kihagyjuk.



3a. ábra *- Példa áramkör kapcsolási rajza és összeállítása*



3b. ábra *- Példa áramkör kapcsolási rajza és összeállítása*

A fontos dolgok a `while(1)` végtelen ciklusban vannak. Az AD konverzió és az UART fogadás megvalósítása interruptokkal történik. Ezekben, csak az értékek mentése történik, hogy minél kevesebb időt töltsön a processzor a megszakítás végrehajtásával. A számítások és a kiértékelés a végtelen ciklusban történik egy meghatározott időzítéssel, jelen esetben másodpercenként.

```
while (1)
{
	// ADC érték konvertálása feszültséggé
	U_NTC = adc_value;
	U_NTC = U_NTC * ADC_CONST;

	// feszültség érték konvertálása hőmérsékletté
	// ezzel az egyenessel való közelítés nagyjából -10 és 40°C között
	// ad jó eredményt
	// a közelítő egyenest egyelőre hidd el nekünk
	temperature = (-172 * U_NTC + 602530)/10000;

	// Ha a mért hőmérséklet nagyobb mint az UART-on beállított érték
	if (temperature > limit)
	{
		// Ventilátor bekapocslása (kitöltési tényező 100%)
		OCR2 = 255;
	}
	else
	{
		// Ventilátor kikapocslása (kitöltési tényező 0%)
		OCR2 = 0;
	}
	// mért feszültség és hőmérséklet, és beállított limit küldése
	printf("mert: %ld mV	%ld C	limit: %d C\n", U_NTC, temperature,
limit);
	_delay_ms(1000);
}
```

Első lépésben az AD konverzió értékét alakítjuk át feszültséggé. Az `adc_value` érték 8 biten tárolja a 0-5V között mért feszültséget. Ez azt jelenti, hogy

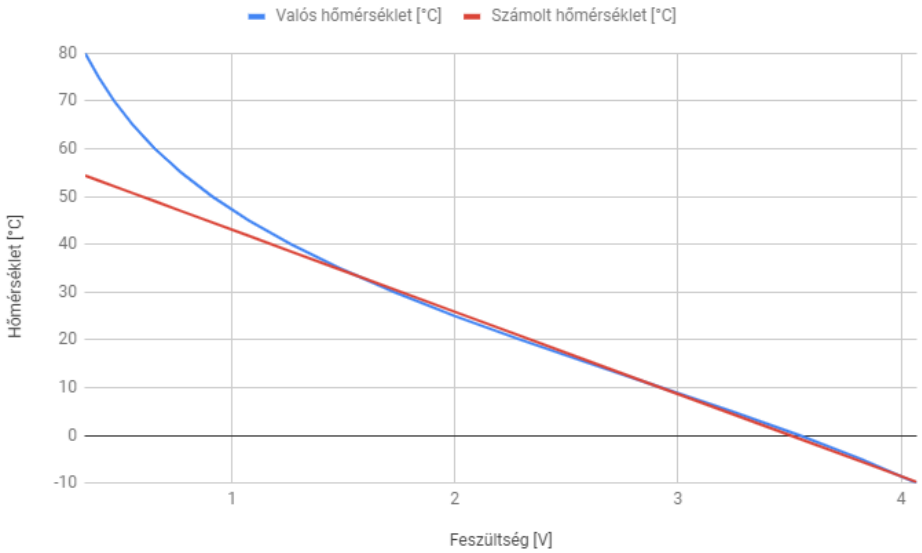
$$U_{NTC} = ADC_{value}/256 \cdot 5\ [V]$$

Képlet határozza meg a mért feszültséget. Ha ezt írjuk bele a programunkba, szomorúan fogjuk tapasztalni, hogy mindig 0V lesz az `U_NTC` értéke. Ennek oka, hogy az `adc_value` 0 és 255 közötti szám. Ezt elosztva 256-tal, a processzor 0-t fog végeredményül adni, mivel egész számok osztásánál mindig csak az egészrészt adja vissza a processzor. Továbbá ha voltban akarjuk kifejezni a mért feszültséget, akkor törtszámok használatára lenne szükség. Ezt az ATMega16A csak korlátozottan támogatja, emiatt érdemes mást kitalálnunk. Számoljuk ki a mért feszültséget mV-ban! A képlet végül így fog kinézni:

$$U_{NTC} = ADC_{value} \cdot 5000/256\ [mV]$$

Itt viszont arra is kell figyelnünk, hogy egy 0-255 közötti szám és 5000 szorzata 0-1.275.000 között van, amelyet 32 bites változóban tudunk csak tárolni, így a műveletet is 32 bites számokon kell elvégeznünk.

Korábbi tananyagban már megvizsgáltuk, hogyan néz ki az adott kapcsolásban az NTC-n mért feszültség a hőmérséklet függvényében.



***4. ábra** - Mért feszültség és a hőmérséklet kapcsolata*

Láthatjuk, hogy a kék görbe -10 és 40°C között egész egyenesnek mondható. Erre célszerű egy egyenest illesztenünk, amely segítségével a mV-ban mért feszültséget át tudjuk számolni °C-ban mért hőmérsékletté. A legoptimálisabb egyenes illesztése bonyolult matematikai folyamat, így higgyétek el nekünk, hogy az alábbi egyenletet kapjuk:

$$T = -0{,}0172 \cdot U_{NTC} + 60{,}253\ [°C]$$

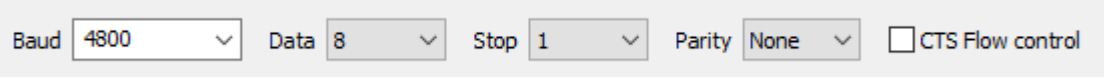
A korábbi megfontolások miatt a megvalósított egyenlet a következő:

$$T = (-172 \cdot U_{NTC} + 602530)/10000\ [°C]$$

Természetesen a mélyebb matematikai ismeret nélkül is, akár “kézzel” illeszthetünk egy egyenest az ábrázolt görbére, majd annak egyenletét leolvashatjuk a grafikonról.

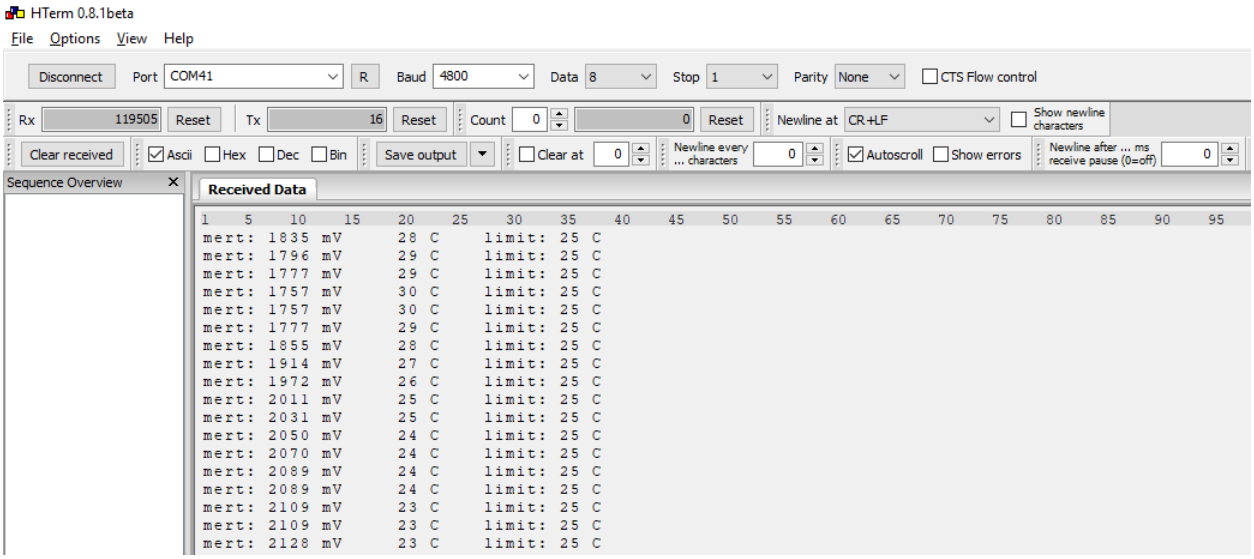
Az UART-on beolvasott határértéket összehasonlítjuk a fentebb kiszámolt hőmérséklettel, és ez alapján állítjuk a ventilátort vezérlő PWM kitöltési tényezőt. Alapesetben csak 0%-ot vagy 100%-ot állítunk be.

A program végén kiírjuk a mért/számolt értékeket a határértékkel együtt, így könnyebb ellenőrizni a program helyes működését. Programozzátok fel a mikrovezérlőt, csatlakoztassátok USB-n keresztül a számítógépetekhez, és indítsátok el az UART tananyagrészben használt HTerm programot. Használjatok 4800-as baudrate-t, 8 bites adathosszt, 1 stop bittel, paritás nélkül.



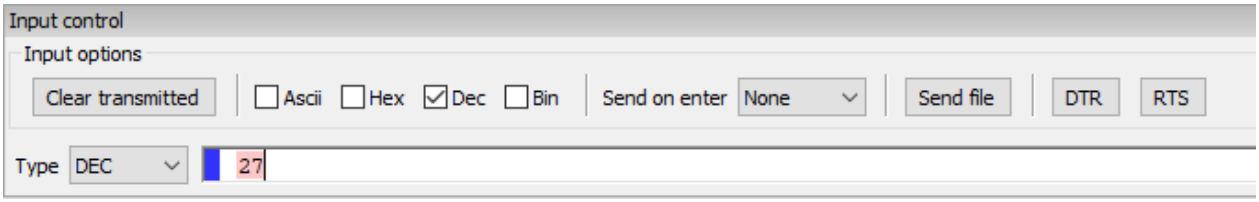
***5. ábra** - UART beállításai a HTerm-ben*

Csatlakozás (Connect) után a következőt fogjátok látni:



***6. ábra** - Mért feszültség és a hőmérséklet fogadása soros porton*

Ahhoz, hogy a határértéket (limit) változtatni tudjátok, állítsátok át az “Input control” típusát “ASC”-ről “DEC”-re. Írjatok be egy tetszőleges számot 0 és 255 között (példában 27), majd Enter-t ütve elküldhetitek a mikrovezérlőnek. Ez a szám jelenti a °C-ban megadott határértéket.



***7. ábra** - Hőmérsékleti határérték megváltoztatása*

Ekkor láthatjátok, hogy a másodpercenként küldött üzenetben megvátozott a limit értéke az általatok beírt számra.

Ha a kezetek melegével felmelegítitek az NTC ellenállást, akkor láthatjátok, hogy csökken a rajta eső feszültség, illetve ezzel egyidejűleg nő a mért hőmérséklet. Amint ez átlépi a határértéket, bekapcsol a ventilátor, illetve kikapcsol, amikor elengeditek a hőmérőt és az elkezd visszahűlni szobahőmérsékletre.

Ezzel megvalósítottuk a ventilátor hőmérsékletfüggő és konfigurálható vezérlését.

Házi feladat

Ha van kedvetek, próbáljátok megírni az alábbi kiegészítéseket:

A ventilátor kitöltési tényezője a beállított határérték és az annál 10°C-kal nagyobb érték között fokozatosan változzon.

UART-on két értéket fogadjon a mikrovezérlő, és ezek között fokozatosan állítsa a kitöltési tényezőt.

Kitekintés

A legkülönfélébb helyeken találkozhatunk mikrovezérlős szabályzásokkal: A polimeráz láncreakcióról

Írta: Ujvári Barbara

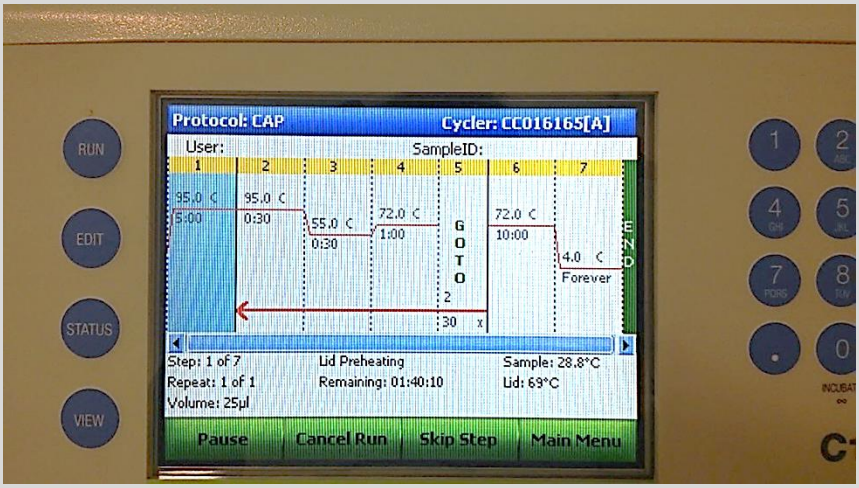
A polimeráz láncreakció (PCR: Polymerase Chain Reaction) története 1985-ben kezdődött, amikor Kary Banks Mullis publikálta a módszer elvét a Science magazinban. A molekuláris biológia eszköztára ezzel egy egyszerű, mégis zseniális eljárással bővült. A módszer egy ciklikus DNS-szintetizáló technika, mely lehetővé teszi a vizsgálni kívánt DNS-szakasz másolását az élő szervezeten kívül. A reakciót többször megismételve az eredeti DNS szekvencia kimutatható mennyiségűre sokszorozható. Az újítás jelentőségét bizonyítja, hogy kidolgozásáért Mullist 1993-ban kémiai Nobel Díjjal jutalmazták. Napjainkra a PCR készülék a molekuláris biológiai laboratóriumok alapfelszerelésének tekinthető.

A PCR reakció lépései egy 50-95°C-os hőmérsékleti tartományban, ciklikusan ismétlődve zajlanak le. Az ehhez használt PCR gép egy mikrovezérlővel szabályzott, automatizált fűtőblokk, melyben a hűtést és a fűtést úgynevezett Peltier-elem nevű alkatrész biztosítja. A készülék képes az eltérő hőmérsékletű szakaszok közti váltást rendkívül gyorsan és pontosan kivitelezni. A programozható hőmérsékleti tartomány általában +4 és +99°C közé esik, a fűtési-hűtési sebesség pedig 2-3°C/másodperc. A gép az egyes lépésekhez szükséges egyenletes hőmérsékletű környezetet tized Celsius fok pontossággal képes biztosítani. A DNS-szintézist az erre a célra szolgáló PCR-csövekben végezzük. A csövek elhelyezésére szolgáló fűtőblokkok jó hővezető-képességű fémből, általában alumíniumból készülnek. A fémblokkok fűthető tetővel fedhetők le, mely megakadályozza a reakcióelegy elpárolgását.

8a. ábra *- egy PCR készülék fotója*

***8b. ábra** - egy PCR készülék fotója*

***9. ábra** - a PCR csövek térfogata 200 µl*



***10. ábra** - egy átlagos PCR protokoll diagramja*