SZAKDOLGOZAT

Kovács György

Debrecen

2010

Debreceni Egyetem
Informatikai Kar

Mikrokontroller alapú léptetőmotor-vezérlő program fejlesztése

Témavezető:
Szabó Zsolt
mérnök tanár

Készítette:
Kovács György
mérnök informatikus

Debrecen
2010

Tartalomjegyzék

Ábrajegyzék

1. Bevezetés

1.1. Témaválasztás

Mérnök informatikus szakon a mérés és folyamatirányítás szakirányt választottam, mert számomra ez ötvözi az informatikát más, általam is érdekesnek tartott dolgokkal, gondolok itt többek közt a fizikára és az elektronikára. Sok érdekes tárgyat hallgathattam, de a mikrokontrollerek programozása különösen elnyerte a tetszésemet. Ezek a kis szilícium lapkák szinte képesek mindenre, mindenre amire képességeit jól kihasználva meg tudom „tanítani”.

1.2. A szakdolgozat célja

A dolgozat célja, mint a címe is mutatja egy léptetőmotor-vezérlő program fejlesztése. A munkát Kígyósi Tibor szaktársammal csoportmunkában vállaltuk. Mindkettőnket érdekel a mikrokontrollerek világa és azért választottunk ilyen szakdolgozati témát, hogy a programozásukban is mélyebb ismereteket szerezhessünk. Egymás között felosztva a feladatot ő vállalta a kommunikációs rész fejlesztését, én pedig a motorvezérlés fejlesztését.

Egy viszonylag nagy projektbe kapcsolódtunk be, melynek célja egy általánosan is használható (moduláris felépítése miatt, melyben a hardverspecifikus részek jól elkülönülnek) létetőmotor-vezérlő fejlesztése. Az általunk kifejlesztett kód a témavezetőnk által fejlesztett programban került felhasználásra.

Az alap program a Debreceni Egyetem Orvos- és Egészségügyi Centrum (Klinikák) területén található Pet-CT Diagnosztikai Kft-nek készült. Egy 60 centiméter átmérőjű körön elhelyezett 12 radioaktivitás detektor közé kell az úgynevezett ágyhoz rögzített és radioaktív izotóppal befecskendezett kísérleti patkányokat behelyezni és mozgatni, mint egy normál CT berendezésnél. A berendezést az ATOMKI-ban fejlesztették és miniPET-nek hívják. Az ágy mozgatását 2 léptetőmotor végzi horizontális és vertikális irányokban, csigatengelyek és fogaslécek segítségével.

Az elektronikai megvalósítással nem kellett foglalkoznunk, az már készre volt

szerelve. A feladatom az volt, hogy a motort közvetlenül meghajtó teljesítményvezérlő számára szükséges három jelet előállítsam, az engedélyező- és a forgási irányt meghatározó jeleket, és a léptető impulzusokat.

Dolgozatom négy nagy részre tagolható:

1. a mikrokontrollerek rövid általános ismertetése,
2. a léptetőmotorok rövid általános ismertetése és a vezérlési módok, valamint meghajtási technikák bemutatása,
3. a vezérlőprogram működési alapelvének bemutatása, a szükséges számítások ismertetése,
4. és az általam írt kódrészletek bemutatása.

2. A mikrokontrollerek

2.1. Általános bemutatás

Mikroszámítógépnek egy integrált áramkörökből álló programot végrehajtani tudó rendszert nevezünk. Típustól függetlenül minden mikroszámítógépnek rendelkeznie kell központi egységgel (mikroprocesszor, CPU), program- és adattárral, be- és kiviteli egységgel (I/O periféria) és az egységek közötti adatáramlást biztosító vonalakkal úgynevezett sínekkel vagy buszokkal. A mikroszámítógépek fejlődésében a méretek jelentős csökkenése a jellemző. Kezdetben minden funkcionális egység külön áramköri kártyákon volt megvalósítva. Ezután a technológia fejlődése lehetővé tette, hogy az összes egységet egy áramköri lapra integrálják. Ezek az egykártyás mikroszámítógépek vagy SBC-k (Single Board Computer). Például a számítógépek alaplapja is ilyen. Az integrálási technológia fejlődésének köszönhetően lehetővé vált, hogy minden részegységet egyetlen szilícium lapkára integráljanak. Ezek az egytokos mikroszámítógépek, másnéven mikrokontrollerek, magyarul mikrovezérlők.

Ezek általában vezérlési feladatokra kitalált és optimalizált kisméretű számítógépek. A személyi számítógépekben és egyéb nagy teljesítményű általános célokra alkalmazott mikroprocesszorokkal szemben itt az egyszerűség a fő szempont. Beágyazott rendszerek fő egységeikén képesek költséghatékonyan ellátni olyan feladatokat, melyek viszonylag egyszerűek, nem igényelnek nagy számítási teljesítményt. Egy egységbe foglalva található meg minden olyan fontos alkatrész, melyre a feladat megoldása során szükség lehet. Az egyes típusok órajele DC–100MHz között van. Egyes típusok órajele működés közben változtatható lehetővé téve a gazdaságosabb energiafelhasználást. Egy speciális üzemmódban, az úgynevezett „alvó” üzemmódban a CPU órajele és a legtöbb funkció ki van kapcsolva, ezért nanowatt körüli fogyasztást tudnak elérni, mellyel jelentősen megnövelhető az akkumulátor élettartama. Más típusaik viszont alkalmasak nagyobb számításigény kiszolgálására is, mint például DSP jelfeldolgozás (Digital Signal Processing = digitális jelfeldolgozás), magas órajellel és természetesen nagyobb teljesítményfelvétel mellett.

Főbb részei a következők:

- oszcillátor (lehet külső és/vagy belső), melyek a mikrokontroller processzora számára

szolgáltatja az órajelet,

- operatív tár a vezérlőprogram futtatására,
- EEPROM memória, mely kalibrációs adatok vagy egyéb paraméterek és adatok tárolására szolgál,
- számlálók az egyszerű impulzusszámlálási és időzítési feladatokra,
- analóg-digitális és digitális-analóg átalakítók, analóg komparátorok,
- PWM (Pulse-Widht Modulation = impulzus-szélesség moduláció) generátor, mely egy speciális négyszögjelet szolgátat a megadott kitöltési tényezővel,
- kommunikációs buszok, mint például SPI (Serial Peripheral Interface), I2C (Inter-Integrated Circuit) és USART (Universal Synchronous/Asynchronous Reciver/Transmitter = univerzális szinkron/aszinkron adóvevő), melyek különböző eszközök egymással való kommunikációját valósítják meg.

Legfontosabb különbség más processzoros rendszerekkel szemben, hogy a mikrokontrollerek beépítve tartalmazzák a fent említett perifériákat, melyek vezérlési folyamatnál szükségesek, és így nem kell külön perifériákra költeni. Továbbá különbség, hogy felépítésük nem a klasszikus Neumann-architektúrára épül, hanem a Harvard-architektúrára. Itt az adat- és utasításbuszok szétválnak, a program- és adatmemóriát elkülöníti, így lehetővé téve a gyorsabb műveletvégzést.

A mikrokontrollerek felhasználási területe szinte végtelen. Mai világunkban szinte mindenhol találkozhatunk velük. Ott vannak az autónk motor ellenőrző elektronikájában és üzemanyagszint-visszajelzőjében, az irodai gépekben, a háztartásban a modernebb televízióinkban és mosógépeinkben, távirányítókban, akkumulátor töltőkben, WC-k öblítőberendezéseiben, kézszárítókban, kéziszerszámokban, játékokban, biztonságtechnikai eszközökben, ipari szinten robotkarokban, gyártósorokban, mérőrendszerekben. Mivel jelentősen csökkentik a költségeket, gazdaságosan alkalmazhatók több eszköz és folyamat digitális szabályozására.

Jellemző input és output eszközeik közé tartoznak például gombok, relék, LED-ek (Light Emitting Diode = fénykibocsátó dióda), optokapuk, rádiófrekvenciás eszközök,

különböző szenzorok, mint például hőmérséklet-, páratartalom-, fényszenzorok, kisméretű LCD kijelzők. Beágyazott rendszereknél nincs személyi számítógépekre jellemző billentyűzet, lemezes egység, képernyő, de ezek nem is szükségesek, mert specifikus feladatuk elvégzéshez nem szükséges emberi beavatkozás, valamint nagy operációs rendszer felügyelete, csak kis bonyolultságú szoftverek működnek, melyek minimális memóriát és programhosszt igényelnek.

Ma már léteznek igen nagy teljesítményű mikrovezérlők is, melyekre már Java-ban is írhatunk kódot vagy akár Linux operációs rendszert is futtathatunk rajtuk. Ezek azonban még nem használatosak széles körben. Jellemzően assembly és C nyelveken folyik a fejlesztés. Egyes típusokra BASIC, illetve Pascal nyelven is lehet fejleszteni. Számos típushoz létezik ingyenes fordítóprogram, amely nagyban elősegíti az adott típus elterjedt használatát. Mai típusaik már flash memóriákat tartalmaznak, így akár több ezerszer is újraprogramozhatók.

A piacon rengeteg mikrovezérlő található, melyből a felhasználó a megoldandó feladathoz legalkalmasabbat tudja kiválasztani. Alacsony áruk, kis fogyasztásuk és rugalmas alkalmazhatóságuk nagyban segítette elterjedésüket. Legnagyobb gyártó cégek a Microchip (PIC), az Atmel, az Intel, a Silicon Laboratories, a Texas Instruments, a Toshiba, a Hitachi és még lehetne folytatni a felsorolást. Egyetemi kereteken belül PIC-ekkel és az Atmel cég ATMEGA128-as mikrovezérlőjével dolgoztunk már.

2.2. A készülékben alkalmazott mikrovezérlő

A motorvezérlő programját az ATOMKI-ban fejlesztették ki egy C8051F120 kódjelű mikrokontrollerre, melynek gyártója a Silicon Laboratories.

Ez a mikrovezérlő a 8-bites mikrovezérlők között a legnagyobbak közé sorolható. A mikrovezérlő jellemzői közül néhányat kiemelve:

- 100 lábú tokozás,
- 24,5 MHz-es belső oszcillátor, amely a belső PLL (Phase-Locked Loop = fáziszárt hurok) áramkörének (automatikus frekvenciaszabályzó áramkör) a segítségével akár 100MHz-es órajel előállítását is lehetővé tesz,
- 64 digitális I/O port,

- maximum 100 MIPS (Million Instructions Per Second = millió utasítás másodpercenként) számítási teljesítmény,
- 128 kB flash belső programozható rendszer-, és 8448 bájt adatmemória,
- 2 soros port,
- 5 darab 16 bites számláló/időzítő,
- 8 darab 12 bites és 8 darab 10 bites analóg-digitális átalakító bemenet,
- 2 darab 12 bites,
- és PCA (Programmable Counter Array = programozható számláló tömb), mely a programban igen fontos szerepet kap.

ábra 1: A C8051F120-as mikrovezérlő

Az 1. ábra forrása: http://media.digikey.com/photos/Silicon%20Labs%20Photos/C8051F120-GQ.JPG

3. A léptetőmotorokról

3.1. Általános bevezetés

A léptetőmotor egy kefe nélküli szinkron villanymotor, mely adott számú lépésből képes egy teljes fordulatot megtenni. Alapvetően különbözik a hagyományos egyenáramú motoroktól, ugyanis digitális jellel kell vezérelni. A jel hatására az adott pozícióba fordul, tehát olyan helyen érdemes alkalmazni, ahol digitális jellel dolgoznak és fontos a megfelelő pozíció és annak pontos ismerete. A leggyakrabban használt motorok esetében 1,8° vagy 0,9° szögelfordulás jellemző lépésenként. Megtalálhatók a számítógép egyes perifériáiban, mint például floppy-, CD- és DVD meghajtókban, merevlemezek író/olvasó fejét is ilyenek mozgatják, nyomtatókban, plotterekben, lapolvasókban, érdekességképpen a hűtőventilátorokat is így mozgatják (a vezérlés bele van integrálva), az iparban CNC gépekben.

Nagy előnyük, hogy az egyes lépések hibái nem adódnak össze. A pozícióba állás pontossága 3-5 %. Másik előnyük, hogy mivel nincs bennük szénkefe, ezért elég megbízhatóak és sokáig használhatóak. További előny még, hogy a nyomatékuk igen nagy és már az indulásnál képesek leadni, valamint a teljes sebességtartományban képesek tartani és a forgási sebesség független a tengely terhelésétől (ez csak részben igaz, mert egy bizonyos határ felett a motor nem képes követni a mágneses tér változását és így lépésvesztés következik be, melynek következménye a pozícióvesztés is, mivel nyílthurkú a vezérlés, azaz nincs visszajelzés a tengely helyzetéről). Hátrányaik közé sorolható a viszonylag kis fordulatszám, a nagyobb vibráció, mint más motoroknál (nagyobb frekvenciás vezérléssel csökkenthető, egyenletesebb lesz a forgás) és nem számítógépes irányítás esetén bonyolult vezérlés. Osztályozni lehet őket felépítésük, meghajtásuk módja és a vezérlési mód alapján.

3.2. Felépítésük

Három jellemző típust különböztethetünk meg a forgó mozgást végző léptetőmotorok között:

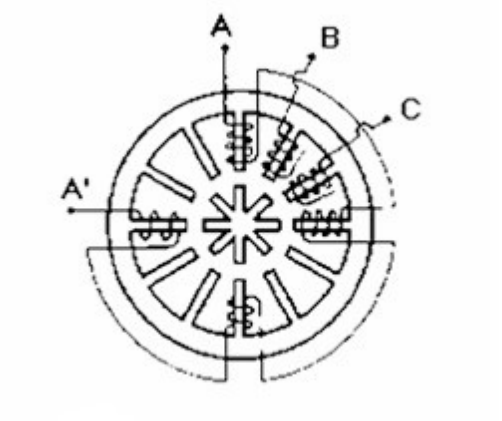
1. a változó reluktancia,

2. az állandó permanens mágneses,
3. és a hibrid szinkron motorokat.

Két részre tagolhatók: a belső forgó részre (rotor) és az ezt körülvevő tekercsekre. A rotor felületén nagyon sűrűn változik a mágneses északi és déli pólus. Ezek távolsága az egyik fő paraméter a lépésszög meghatározásában. A tekercseket gerjesztve változtatható a belső mágneses tér és a vonzás, illetve taszítás hatására fordul el a rotor. Léteznek lineáris léptetőmotorok is, melyeknél a mozgórész egy állandómágnesből és két elektromágnesből épül fel, az állórész pedig egy fogazott acélrúd, de ezek bővebb ismertetését nem kívánom dolgozatomban megtenni.

3.2.1. Változó reluktancia léptetőmotorok

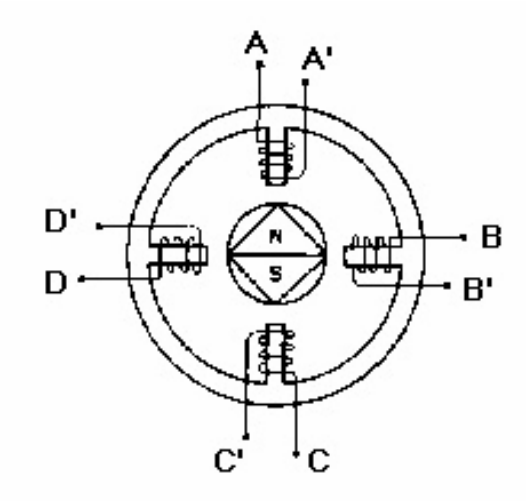
A forgó rész fogazott mágnesesen lágy anyag, mely a ferromágneses anyagok egyik csoportja és elektromágnesek és transzformátorok vasmagjaként szoktak alkalmazni. Az állórész gerjesztésekor a mágneses erővonalak az energiaminimumra törekszenek, amit a legközelebbi tekercs helyzeténél tudnak elérni, amikor a mágneses ellenállás a legkisebb. Az ekkor fellépő nyomatékot reluktancia-nyomatéknak nevezik (innen ered az elnevezés). Mivel a rotor nem állandó mágnes, ezért az ilyen típusú motoroknak nincs tartónyomatéka, amikor nem gerjesztjük a tekercseket. A fogak számát szokták növelni úgy, hogy több fogazott részből építik fel és egymáshoz képest még el is forgatják, ilyenkor a külső részben többfázisú a tekercselés.



ábra 2: Változó reluktancia
léptetőmotor elvi felépítése

3.2.2. Állandó mágneses léptetőmotorok

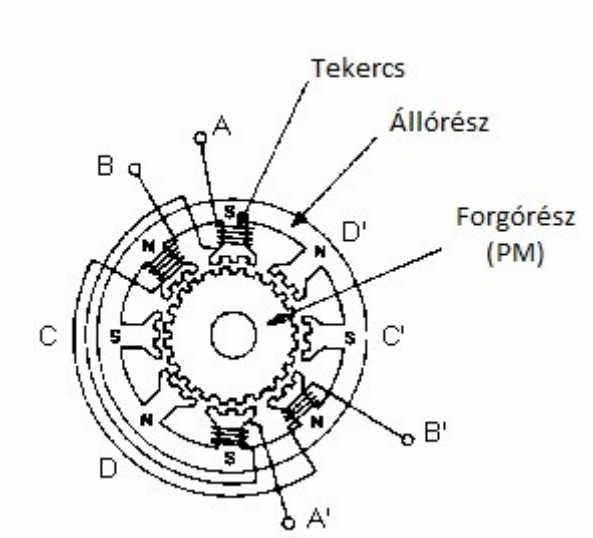
A rotor egy permanens mágnesből készült anyag, mely hosszirányban sűrűn mágnesezett. A tekercs gerjesztésekor a rotor a legközelebbi olyan helyre fordul, melynél az erővonalak a legrövidebbek. Nagy előnye az előző motorokhoz képest, hogy van tartónyomaték. Hátrányai között említhető, hogy a vezérlési határfrekvencia alacsony és forgás közben az állandómágnes átmágneseződhet és változhatnak az eredeti munkapontok.



ábra 3: Állandó mágneses léptetőmotor
elvi felépítése

3.2.3. Hibrid léptetőmotorok

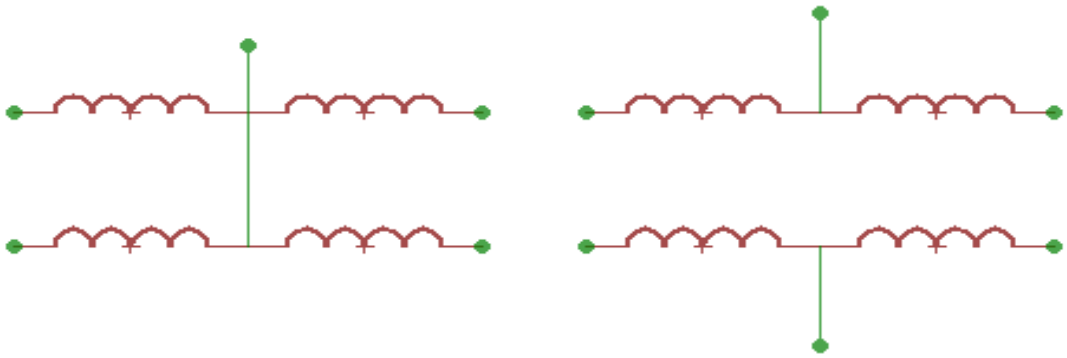
Az előző két típus előnyeit egyesítik és ennek köszönhetően a legelterjedtebbek. Az álló- és forgórész is fogazott (nem feltétlenül azonos számban), de a forgórész állandó mágnesből készül. Jellemző rájuk a reluktancia motorok viszonylag nagy sebessége és felbontása, valamint az állandómágneses motorok nagy nyomatéka.



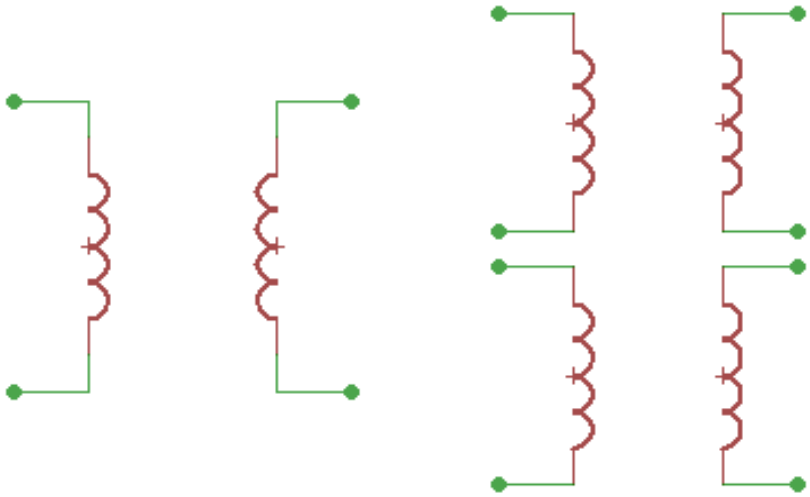
ábra 4: Hibrid léptetőmotor elvi felépítése

3.3. Vezérlési módok

Alapvetően két típus létezik: az unipoláris és a bipoláris. A különbség a kivezetések számában van, tehát igazából nem teljesen vezérlési mód típusok hanem különböző vezetékezési típusok. A gyártás során nincs is különbség, majd a felhasználó tud dönteni, hogy neki melyik vezérlési mód a megfelelő. A bipoláris motoroknak általában páros (4, 6, 8) míg az unipolárisaknak 5 vagy 6 kivezetésük van attól függően, hogy a tekercseket hogyan kötjük össze és vezetjük ki.



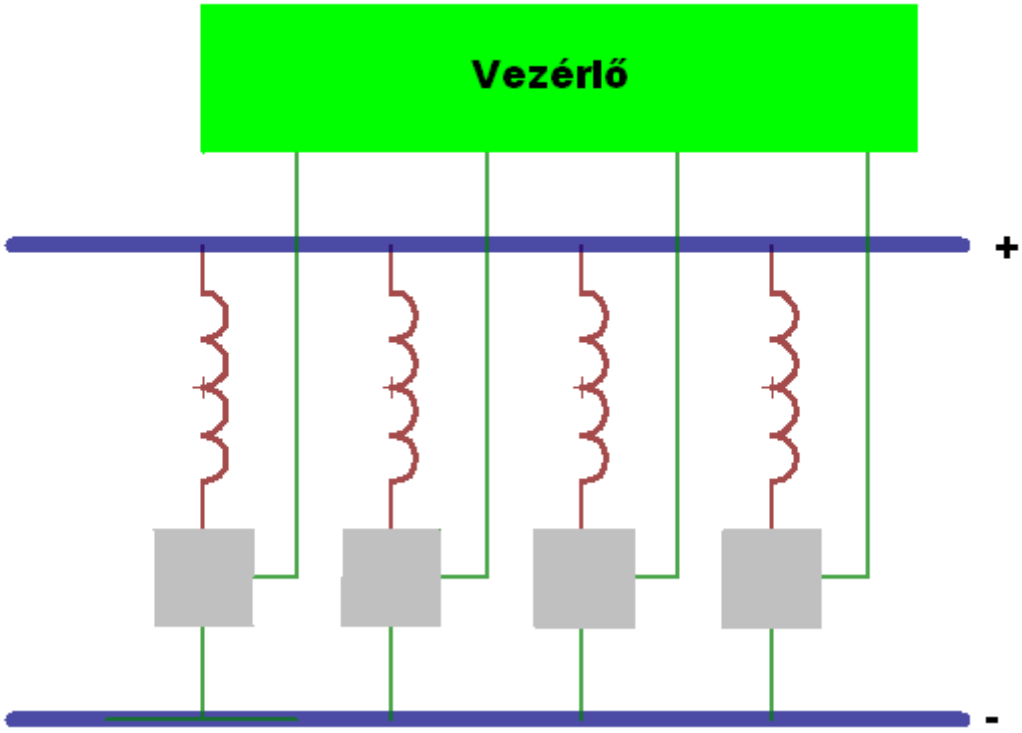
ábra 5: 5 és 6 kivezetéses vezetékezés elvi rajza



ábra 6: 4 és 8 kivezetéses vezetékezés elvi rajza

3.3.1. Unipoláris vezérlés

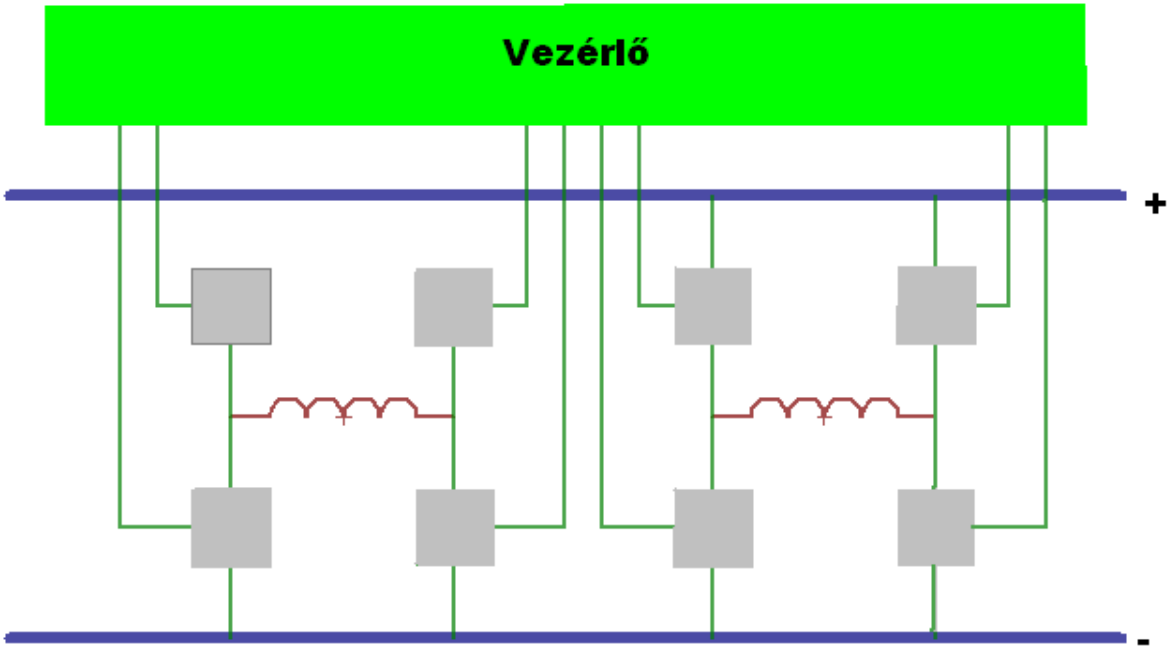
Ez a legegyszerűbb típus. Vezérlésük jóval egyszerűbb a bipolárisaknál, kevesebb alkatrészből is megoldhatók. Két egymástól független tekercset tartalmaznak, melyek rendelkezhetnek egy középvezetékkel, melyet akár egymással összekötve is kivezethetnek. Meghajtásuk igen egyszerű: a közös szálat vagy független középső szálakat a tápvonalra kötjük és a tekercseket valamilyen vezérlési mód szerint a földelésre kötjük. Ez a kapcsolás megvalósítható tranzisztorral vagy relével is (az ábrán a szürke négyzet jelöli a kapcsolóelemet). A tekercsek magjának fele vesz részt a mágneses tér létrehozásában, így teljesítményük kissé csökken. Ha 6 kivezetés található rajtuk vezérelhetőek bipoláris módon is.



ábra 7: Egyszerű unipoláris vezérlés elvi rajza

3.3.2. Bipoláris vezérlés

Tekercselésükben nincs középkivezetés, így a motor maga egyszerűbbé válik, de a meghajtó elektronika már bonyolultabb, mert itt a tekercsekben folyó áram irányát is változtatni kell. Ez két úgynevezett H-híd kapcsolással lehet megoldani. Ez kétszer annyi alkatrészt és kétszer annyi I/O portot igényel a mikrokontrollerünktől. A tekercselés teljes egészében részt vesz a mágneses tér kialakításában, ezért hatásfoka jobb az unipolárisnál. Ha 6 vagy 8 kivezetésük van, akkor unipolárisan is vezérelhetők.



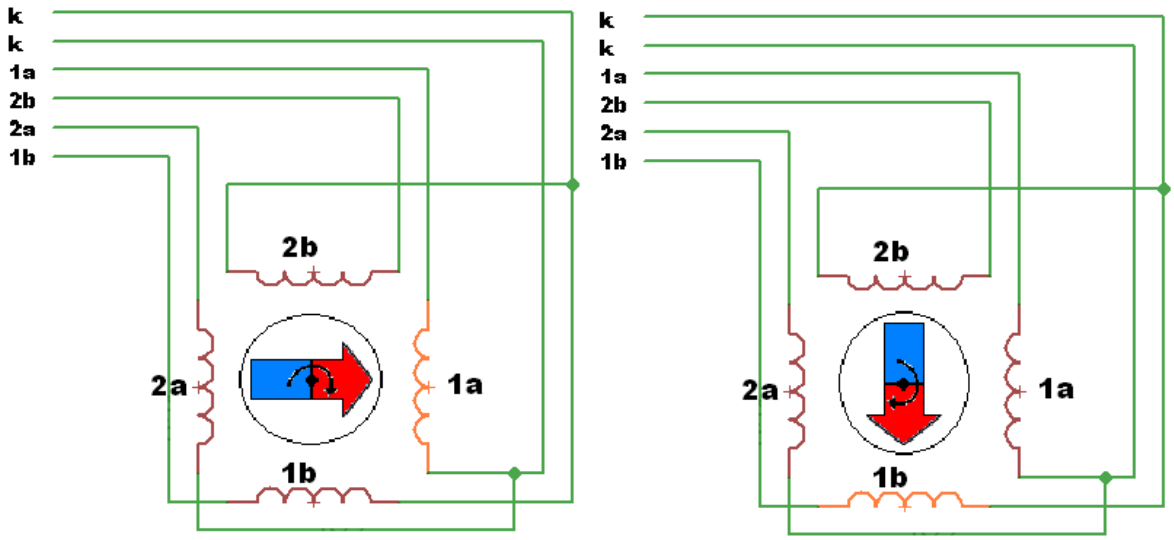
ábra 8: Elvi bipoláris vezérlés H-híd kapcsolással

3.4. Meghajtási üzemmódok

Főleg az unipoláris léptetőmotorokat lehet sokféle módon vezérelni.

3.4.1. Egyfázisú üzemmód

Egyszerre egy tekercs gerjesztett. Nevezik még hullámhajtásnak is. Ebben a módban a lépésszög megegyezik a fizikai lépésszöggel. Előnye, hogy kevés energiát használ, mivel egyszerre csak egy tekercs gerjed. Nagy hátránya az alacsony nyomaték és a kis felbontás. Csak speciális estekben alkalmazzák, például akkumulátoros működtetés esetén.



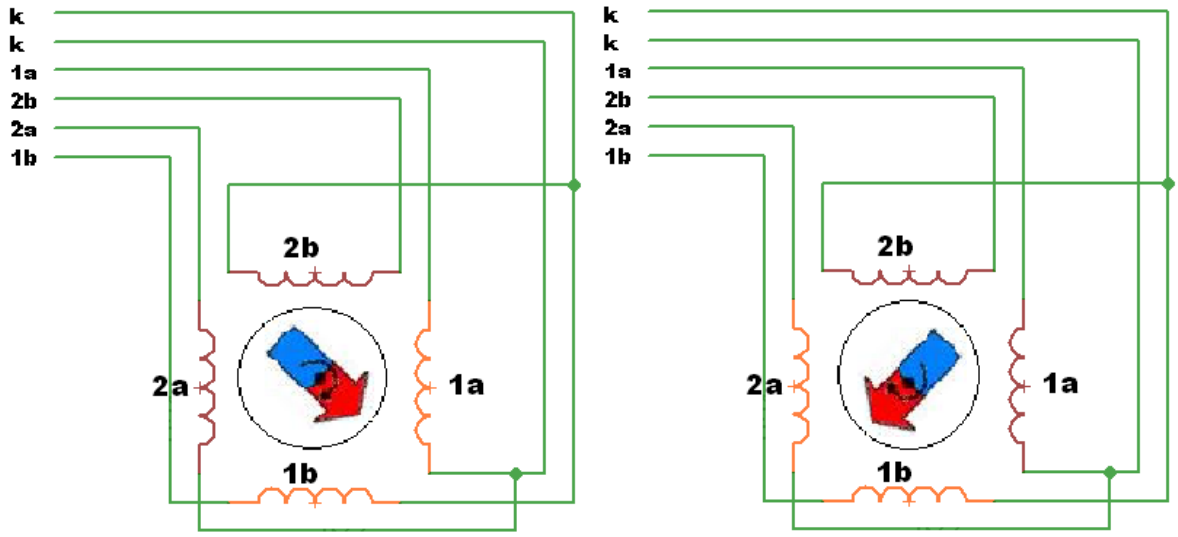
ábra 9: Egyfázisú hajtásmód első két lépése

Az alábbi táblázatban az a szekvencia látható, mely alapján az egyes tekercseket gerjesztjük. Látható, hogy a negyedik lépés után ismétlődik a szekvencia.

	tekercs			
lépésszám	1a	1b	2a	2b
1.	**1**	0	0	0
2.	0	**1**	0	0
3.	0	0	**1**	0
4.	0	0	0	**1**
5.	**1**	0	0	0
6.	0	**1**	0	0
7.	0	0	**1**	0
8.	0	0	0	**1**

3.4.2. Egészlépéses üzemmód (fullstepping)

Egyszerre két tekercset gerjesztjük. Ugyanazt a szögelfordulást produkálja, mint a hullámhajtás, de közel kétszer akkora nyomaték mellett, azonban az áramfelvétel megduplázódik.



ábra 10: Egészlépéses mód első két lépése

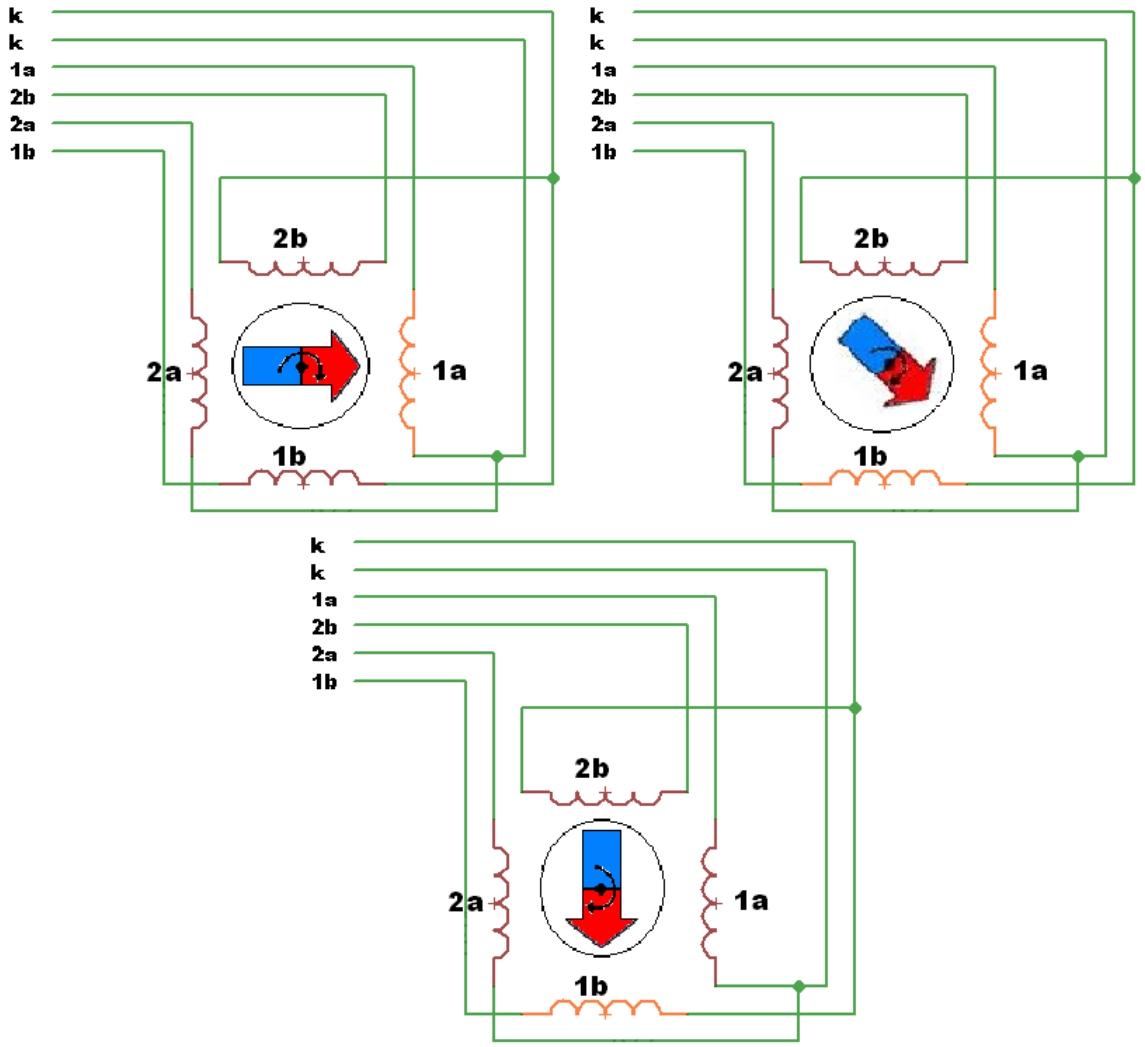
A gerjesztési szekvencia ebben az esetben is ismétlődik a negyedik lépés után.

	tekercs			
lépésszám	1a	1b	2a	2b
1.	**1**	**1**	0	0
2.	0	**1**	**1**	0
3.	0	0	**1**	**1**
4.	**1**	0	0	**1**
5.	**1**	**1**	0	0
6.	0	**1**	**1**	0
7.	0	0	**1**	**1**
8.	**1**	0	0	**1**

3.4.3. Féllépéses üzemmód (halfstepping)

Ebben az esetben először az egyik tekercs gerjed, majd a mellette lévő is és utána csak a második, azaz az előző két mód keveréke. A körülforduláshoz szükséges lépésszám megduplázódik, így pontosabban pozicionálhatunk ugyan azzal a motorral és a dinamikai

jellemzői is javulnak. Elvben a leadott nyomaték is megduplázódik (valójában az előző két mód nyomatéka közé esik), de mivel itt is két tekercset gerjesztünk az áramfelvétel itt is duplázódik.



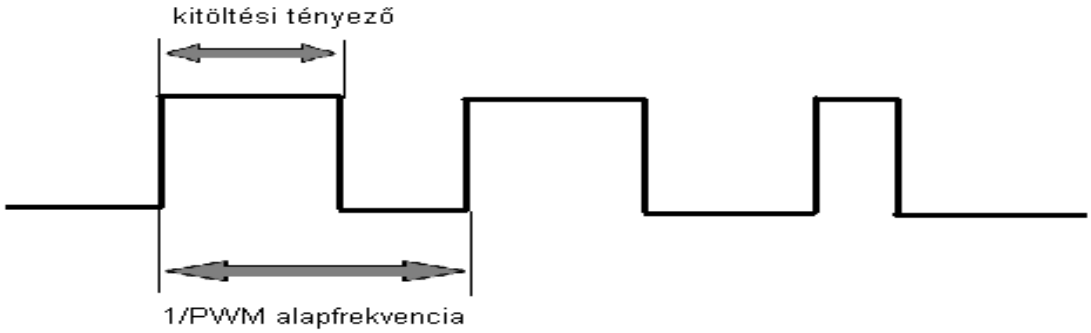
ábra 11: Féllépéses mód első három lépése

A szekvencia táblázatban nincs ismétlődés csak nyolc lépés után, tehát valóban megduplázódik a lépésszám az előző két meghajtási módhoz képest.

	tekercs			
lépésszám	1a	1b	2a	2b
1.	**1**	0	0	0
2.	**1**	**1**	0	0
3.	0	**1**	0	0
4.	0	**1**	**1**	0
5.	0	0	**1**	0
6.	0	0	**1**	**1**
7.	0	0	0	**1**
8.	**1**	0	0	**1**

3.4.4. Mikroléptetéses üzemmód (microstepping)

Ennél a vezérlési módnál a motor névleges feszültségét és/vagy áramát (teljesítményvezérlő elektronikától függ) szétosztjuk az egymás melletti tekercsek között. Minél több részre osztjuk, annál több lépésből tevődik össze egy egész lépéshez szükséges szögelfordulás. A tekercset PWM jellel gerjesztjük, melynek kitöltési tényezőjével tudjuk szabályozni a tekercs által létrehozott mágneses tér erősségét. Elvben nagyon kis lépésszög elérhető, de a valóságban a vezérlő alkatrészeinek késleltetése szab határt. A bevezető részben említett vibráció ilyen vezérléssel csökkenthető. Az egyenletes mozgatás fogaslécek és csigatengely alkalmazásával még inkább elérhető, tehát mechanikailag is javíthatjuk a rendszert.



ábra 12: Egy PWM ciklus

A fejezetben található 5., 6., 7., 8., 9.,10., 11. és 12. ábrák és a három léptetési táblázat az irodalomjegyzékben felsorolt magyar források közül a 6., 15., és 16. alapján készültek EAGLE Layout Editor 5.6.0 freeware verziójával, valamint a Microsoft Windows XP operációs rendszeren található Paint nevű rajzoló program felhasználásával. A 2., 3. és 4. ábra forrása sorrendben:

- http://www.anaheimautomation.com/images/old-site/stepmtr5.gif
- http://www.anaheimautomation.com/images/old-site/stepmtr6.gif
- http://www.steppermotors.us/images/stepmtr7.gif

4. A vezérlő program

Feladatom olyan számítások elvégzése volt, mely a felhasználó számára megkönnyíti a program paraméterezését, valamint a kapott képlet alapján gyökértékek generálása, tárolása egy táblázatban, és olyan kódrészlet megírása, mely a mikrovezérlőn fut és ezeket kezelni tudja.

Ebben a fejezetben bemutatom a léptetőmotor vezérléséért felelős kódrészletet és hogy miért és mit dolgoztam. A továbbiakban erről a kódrészletről úgy beszélek, mint egy önálló program. Ez az ATOMKI által fejlesztett miniPET nevű berendezésének az ágymozgatásért felelős, mikrokontrollerrel megvalósított motorvezérlőjének a programjába lett beintegrálva. Az általam készített kódrészletek nem közvetlenül lettek felhasználva, hanem kisebb módosításokkal és bővítésekkel lettek beépítve, ezért a mellékelt végleges kódoknak csak egyes részei az én munkám.

4.1. Célkitűzés

A feladat alapjában véve nem tűnik túlságosan bonyolultnak. Léptetőmotor segítségével kell mozgatni egy úgynevezett ágyat, amelyre fel van rögzítve a vizsgálandó test (egy kisállat vagy egy fantom test). A motor fogaskerekek és fogaslécek rendszerén keresztül kapcsolódik az ágyhoz. A berendezést (miniPET) irányító számítógép a mozgatáshoz szükséges parancsot valamilyen kommunikációs csatornán (soros port vagy Ethernet) elküldi a motorvezérlőnek. A motorvezérlőnek „csak” annyi dolga van, hogy a parancsban megadott lépéssel, a megadott irányba elforgassa a motort. Ehhez be kell állítania az *engedélyező*- és az *iránykiválasztó* kimeneteteket, majd ki kell adnia a *léptető* kimeneten a parancsban megadott számú impulzust.

4.2. Programabble Counter Array

A 2.2-es bekezdésben már említettem, hogy a PCA igen fontos szerepet tölt be a program működésében. Az egész kód állapotgépesen van felépítve és a PCA ütemezi a kiszolgálást. Ugyanis a MAIN függvény csak az egyes modulok kiszolgálófüggvényeit hívja meg egy végtelen ciklusban. A program futása közben az egyes függvényhívások és egyéb

folyamatok az aktuális állapot szerint történnek, melyek futás közben változnak. A kiszolgálófüggvény meghívása utána modulban állapot és elvégzendő feladat ellenőrzés történik, mely eredményeképpen:

- ha van feladata az aktuális modulnak, akkor állapotváltozás történik és megkezdődik a feladat végrehajtása, majd visszatér a MAIN-be,
- ha nincs feladata, akkor állapotváltozás nélkül lép vissza a MAIN-be

és a következő kiszolgálófüggvény meghívása következik. Ezzel a technikával a szűkös erőforrások ellenére is lehet hatékonyan szekvenciális műveleteket végrehajtani.

Az alkalmazott mikrokontroller 16 bites számláló/időzítő egysége két 8 bites SFR-ből (Special Function Register = speciális funkciójú regiszter) épül fel.

4.3. Problémák és a program működésének alapelvei

A megvalósítás több problémát vet fel. Mennyi ideig tartson egy impulzus? Ha az impulzusok periódusideje nagy, azaz a léptetési frekvencia alacsony, akkor az ágy „lassan” mozog, ha pedig a léptetési frekvencia nagy, akkor az ágy „gyorsan” mozog. Mindkét sebességre szükség van, sőt nem csak kétféle sebességre van szükség, hanem a rendszer elektromos és mechanikai korlátai által meghatározott korlátokon belül tetszőleges sebességre szükség lehet.

A probléma abból adódik, hogy „gyorsan” is szeretnénk forgatni a motort. Viszont a motort álló helyzetből nem lehet tetszőlegesen nagy fordulatszámmal elindítani, mivel a motornak véges nagyságú a forgatónyomatéka, a mozgatott rendszernek pedig nullánál nagyobb a tömege. Ugyancsak nem lehet a már „gyorsan” mozgó rendszert hirtelen megállítani. Egy hirtelen megállás mechanikai meghibásodással járhat (pl. fogaskerekek fogai törhetnek le).

Ezeket mérlegelve nyilvánvaló, hogy az ágy „A” pontból „B” pontba haladása legalább három szakaszra bontható.

1. Gyorsítási szakasz, ahol az ágyat az „A” pontból (álló helyzetből) indítjuk és a megadott maximális sebességre gyorsítjuk.
2. Állandó sebességű szakasz ahol az ágyat a gyorsítási szakaszban elért sebességgel

mozgatjuk.

3. Lassítási szakasz, ahol az ágyat lelassítjuk és megállítjuk az előírt „B” pontban.

A gyorsítási és a lassítási szakaszban az ágy gyorsulása állandó és azonos nagyságú, csak természetesen ellentétes irányú.
A sebesség nagyságát időben ábrázolva általános esetben trapézt, speciálisabb esetben pedig egy egyenlő szárú háromszöget kapunk, ilyenkor kimarad a második szakasz. Ez a speciális eset olyankor áll elő, amikor az „A” és a „B” pont olyan közel van egymáshoz, hogy nem elég a távolság ahhoz, hogy a maximális sebességre fel lehessen gyorsítani az ágyat. A gyorsítási szakaszban a léptetőmotort egy ω_0 szögsebességgel indítva, a állandó szöggyorsulással, T ideig forgatjuk. A következő állandó sebességű szakaszban az elért szögsebességgel forgatjuk tovább a motort. Az utolsó szakaszban pedig az első szakaszban leírtakat hajtjuk végre időben visszafelé.

A motorvezérlőnek mindenképp biztosítania kell azt, hogy a felhasználó menet közben bármikor megadhassa a mozgatáshoz szükséges paramétereket, ráadásul a felhasználó számára praktikusabb módon az f_{min} , f_{max} és T adatokkal, amelyek jelentése a következő:

- f_{min} : A léptetőmotor álló helyzetből történő indulásakor alkalmazott léptetési frekvencia.
- f_{max} : A maximális léptetési frekvencia. A gyorsítási szakasz végére (amennyiben az „A” és a „B” pont távolsága megengedi) ezzel a frekvenciával kell léptetni a motort.
- T : A gyorsítási szakasz ideje, azaz ennyi idő alatt kell a kezdő léptetési frekvenciáról a maximális léptetési frekvenciára felgyorsítani a motort.

A megadott f_{min} , f_{max} és a T adatokból kiszámoljuk, hogy hány darab lépés kell a gyorsuló szakaszra. Két ilyen szakasz van, az 1-es gyorsítási-, és a 3-as lassítási szakasz. Mivel a 3-as szakasz az 1-es szakasz időben ellentétes irányú megismétlése, így mindkét szakasz azonos léptetéssel valósul meg. Amennyiben az így meghatározott érték kétszeresénél nagyobb a parancsban megadott lépésszám, akkor mindhárom szakasz megvalósítható, ellenkező esetben a parancsban megadott lépésszám felényi lépésben

gyorsítjuk a motort, majd ugyan annyi lépésben lassítjuk. Amennyiben a parancsban megadott lépésszám páratlan, akkor a 3-as szakasz után még egy léptető impulzust kiadunk az előző periódusidőnek megfelelően.

4.3.1. A gyorsuló szakaszok

Az első szakaszban az indulási léptetési frekvencia periódus idejét kiszámoljuk és felprogramozzuk a mikrokontrollerben található egyik PCA-t úgy, hogy az megszakítást generáljon ezen idő elteltével, és közben kiadunk egy impulzust a *léptető* kimeneten. Az egy darab impulzust úgy állítjuk elő a *léptető* kimeneten, hogy a PCA felprogramozása előtt „1”-be állítjuk a kimenetet, majd a fenti idő felével programozzuk fel a PCA-t. Az idő leteltével a PCA által generált megszakításkor pedig „0”-ba állítjuk a kimenetet és ismét felprogramozzuk az idő felével a PCA-t. A következő megszakítás pedig már a ténylegesen várt megszakítás lesz. Az idő leteltével keletkezett megszakítást kiszolgáló rutinban kiszámoljuk azt a következő periódusidőt, ami ahhoz kell, hogy teljesüljön a gyorsítási szakaszra előírt feltétel. Az így meghatározott periódusidő alatt ismét kiadunk egy impulzust és ezzel az idővel újra felprogramozzuk a PCA-t. Ezt mindaddig ismételjük, ameddig el nem érjük a maximális léptetési frekvenciát.

A harmadik szakaszban a fent leírtakat hajtjuk végre visszafelé.

4.3.2. Az állandó sebességű szakasz

Ebben a szakaszban az 1-es szakasz végén alkalmazott periódusidővel és az előző szakaszban ismertetett módon használva a PCA-t továbbléptetjük a motort annyival, amennyivel több a gyorsuló szakaszhoz szükséges lépésszámok kétszeresénél a parancsban megadott lépésszám.

4.3.3. A program működési alapelvéhez szükséges számítások

A fenti vezérlést a motorvezérlőben alkalmazott mikrokontroller gyártója által megvalósított motorvezérlő program a következő alapelv szerint valósítja meg, mely az úgynevezett lineáris sebesség profil.

Az (1.) egyenlet azt írja le, hogy mennyi idő teljen el két egymást követő lépés között, melyet a mikrokontroller időzítőjével ellenőrzünk.

(1.) $T_n = T_{n+1} - T_n$

A szöggyorsulásra az

(2.) $\omega = \alpha * t_n$

képlet érvényes. A lépés pozíciója az

(3.) $n = \frac{1}{1} * \alpha * t_n^2$

képlettel írható le. Ennek az egyenletnek az időre vonatkozó megoldása

(4.) $t_n = \sqrt{\frac{2*n}{\alpha}}$

Ez az abszolút idő szükséges a lineáris gyorsítási profil biztosításához. Az első egyenletből szárazó lépési periódusokat a (4.) egyenletben felhasználva kapjuk a

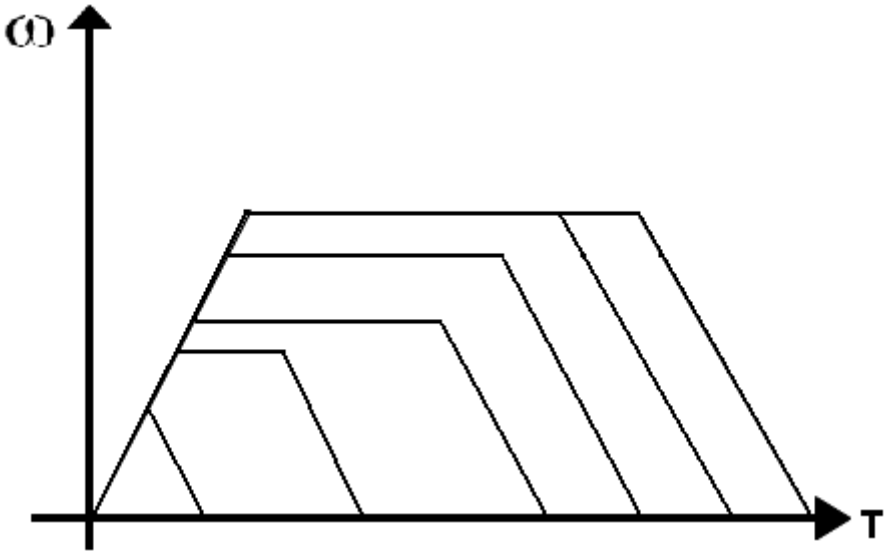
(5.) $T_n = t_{n+1} + t_n = T_0 * (\sqrt{n-1} - \sqrt{n})$

által leírt lépési periódusokat. Az állandó gyorsításban használt T_0 -t a következőképpen írhatjuk le:

(6.) $T_0 = \sqrt{\frac{2}{\alpha}}$

A T_0 értéke határozza meg a kezdeti lépési periódust $n = 0$ értéknél.

A lineáris sebesség profil megvalósítása nem olyan egyszerű, mint amennyire tűnik. Az értékeket egy lineáris sebesség táblázatban kell tárolni, melyeknek követniük kell az (5.) egyenletben leírt nemlineáris egyenletet. A nem megfelelő kapcsolat alkalmazása a léptetőmotor táblázatban gyakran hibás működést eredményezhet. Néhány különböző profilt szemléltet a 13. ábra.



ábra 13: Különböző profilok

Egyik leggyakoribb hiba, hogy a lépési periódus lineárisan csökken a lépések számával. Ha például a kezdeti lépési periódus 256 * „timer tick” („timer-tick” jelöli azt az időt, amire a mikrokontroller egyik időzítőjét felprogramozzuk, ami a felprogramozott időnek megfelelően megszakításokat generál az időzítő rutin számára) és ezt csökkentik minden lépésnél. Ennek eredménye nemlineáris sebesség, mely hiperbolikusan nő, ahogy lépésenként közelít a nullát. Az ilyen profil alig mozog az elején, majd a végén a sebesség túl gyorsan nő.

A másik leggyakoribb hiba, hogy a lépési időszakot a lépésszám inverzével csökkentik. Ez olyan sebességgel végződik, mely lineárisan függ a lépések számától, de nem veszi figyelembe, hogy a lépési periódus folyamatosan változik. A sebesség az eltelt idő és nem a lépésszám függvénye. Ha a sebességet az eltelt idő függvényében ábrázoljuk, akkor a kapott görbe másodfokú, a sebesség az idő négyzetével nő. Ez a profil is túl lassan indul és a végén a gyorsulása túl nagy.

4.4. Az általam végzett számítások

Az előző részben említett mintaprogram az (1.)-(6.) egyenleteknek megfelelően úgy működik, hogy mindig ugyanannyi (256) lépés alatt gyorsítja fel a motort, és csak egy paramétert lehet megadni, a T_0 -at. Viszont mi azt szeretnénk, ha a felhasználó megadhatná

az induló-, és a maximális léptetési frekvenciát, valamint a gyorsítási szakasz idejét. Ezért részletesen újraszámoltam a gyorsítási szakaszra vonatkozó összefüggéseket, annak érdekében, hogy megkapjam azokat a képleteket, amivel a felhasználó által megadott adatokból kiszámolhatom a vezérléshez szükséges értéke(ke)t. A végcél a PCA beállításához szükséges érték meghatározása.

Természetesen a mintapéldához hasonlóan táblázatot fogok használni a négyzetgyökös kifejezés meghatározásához, ami ugyancsak 256 elemű lesz, de kétbájtos adatokat fog tartalmazni. Továbbá megengedem, hogy a gyorsítási szakasz lépcsős legyen, azaz két egymást követő idő azonos lehessen.

A gyorsítási szakaszra az

(7.) $\omega(t)=a*t+\omega_0$

képlet vonatkozik, ahol $t\in[0,T]$. Mivel azonban a vezérlőnk csak diszkrét időpillanatokban képes beavatkozni, ezért a

(8.) $t_N-t_0=T$

képletet használható számunkra, ahol t_n a beavatkozás időpontjait jelenti és $n=0,1,\ldots,N$. Ebből következik, hogy.

(9.) $\omega(T_n)=a*T_n+\omega_0$

A megtett elfordulás a következőképpen írható le:

(10.) $\theta(t_n)=\int_{t_0}^{t_n}\omega*dt=\frac{1}{2}*a*(t_n^2-t_0^2)+\omega_0*(t_n-t_0)$

Legyenek t_n -ek olyanok, hogy minden egymást követő időpillanat között ($T_n=t_{n+1}-t_n$) ugyanannyit forduljon a motor, azaz

(11.) $\theta(t_{n+1})-\theta(t_n)=K*\varphi_0$,

ahol $K>0$ egész és $\varphi_0=1{,}8°$ vagy $0{,}9°$, azaz a motor lépésenkénti szögelfordulása. Természetesen a t_n -ek közötti időkülönbségek nem lesznek azonosak.

(12.) $$K*\varphi_0=\theta(t_{n+1})-\theta(t_n)=\frac{1}{2}*a*(t_{n+1}^2-t_0^2-t_n^2+t_0^2)+\omega_0(t_{n+1}-t_0-t_n+t_0)=$$
$$=\frac{1}{2}*a*(t_{n+1}+t_n)*(t_{n+1}-t_n)+\omega_0*(t_{n+1}-t_n)=$$
$$=\frac{1}{2}*a*(2*t_n+T_n)*T_n+\omega_0*T_n=$$
$$=\frac{1}{2}*a*T_n^2+(\frac{1}{2}*a*2*t_n+\omega_0)*T_n$$

(13.) $$\frac{a}{2}*T_n^2+(a*t_n+\omega_0)*T_n-K*\varphi_0=0$$

A fenti egyenlet gyökei a másodfokú egyenlet megoldóképlete alapján:

(14.) $$T_{n\,1,2}=\frac{-(a*t_n+\omega_0)\pm\sqrt{(a*t_n+\omega_0)^2-4*\frac{a}{2}*(-K*\varphi_0)}}{2*\frac{a}{2}}\quad .$$

$$a>0$$
$$t_n>0$$
$$\omega_0>0$$
$$\varphi_0>0$$

Mivel csak pozitív megoldások jöhetnek szóba, ezért

(15.) $$T_n=\frac{-a*t_n-\omega_0+\sqrt{(a*t_n+\omega_0)^2+2*a*K*\varphi_0}}{a}$$

és mivel a pozitív, ezért elvégezhető a következő átalakítás:

$$T_n=\frac{-a*t_n}{a}-\frac{\omega_0}{a}+\frac{\sqrt{(a*t_n+\omega_0)^2+2*a*K*\varphi_0}}{a}=$$
$$=-t_n-\frac{\omega_0}{a}+\frac{\sqrt{(a*t_n+\omega_0)^2+2*a*K*\varphi_0}}{\sqrt{a^2}}=-t_n-\frac{\omega_0}{a}+\sqrt{\frac{(a*t_n+\omega_0)^2}{a^2}+\frac{2*a*K*\varphi_0}{a^2}}=$$
(16.)
$$=-t_n-\frac{\omega_0}{a}+\sqrt{(\frac{a*t_n+\omega_0}{a})^2+\frac{2*a*K*\varphi_0}{a^2}}=$$
$$=-t_n-\frac{\omega_0}{a}+\sqrt{(t_n+\frac{\omega_0}{a})^2+\frac{2*K*\varphi_0}{a}}=\sqrt{(t_n+\frac{\omega_0}{a})^2+\frac{2*K*\varphi_0}{a}}-(t_n+\frac{\omega_0}{a})=$$
$$=\sqrt{\frac{2*K*\varphi_0}{a}}+[\sqrt{[\sqrt{\frac{a}{2*K*\varphi_0}}*(t_n*\frac{\omega_0}{a})]^2+1}-\sqrt{[\sqrt{\frac{a}{2*K*\varphi_0}}*(t_n+\frac{\omega}{a})]^2}]$$

Vezessük be a következő jelölést, mint független változót:

(17.) $X_N=[\sqrt{\frac{a}{2*K*\varphi_0}}*(t_n+\frac{\omega}{a})]^2$,

és követeljük meg, hogy $X_N \le 255$ teljesüljön. Ekkor a következő kifejezést kapjuk:

(18.) $T_n=\sqrt{\frac{2*K*\varphi_0}{a}}*[\sqrt{X_N+1}-\sqrt{X_N}]$

(19.) $t_n=\sqrt{\frac{2*K*\varphi_0}{a}}*\sqrt{X_N}-\frac{\omega_0}{a}$

Beírva ezt a (8.) egyenletbe, amelyből az X_N és K lehetséges értékeit határozhatjuk meg, azt kapjuk, hogy:

(20.) $$T=t_N-t_0=\sqrt{\frac{2*K*\varphi_0}{a}}*\sqrt{X_N}-\frac{\omega_0}{a}-\sqrt{\frac{2*K*\varphi_0}{a}}*\sqrt{X_0}-\frac{\omega_0}{a}=$$
$$=\sqrt{\frac{2*K*\varphi_0}{a}}*(\sqrt{X_N}-\sqrt{X_0})$$

(21.) $\sqrt{\frac{2*K*\varphi_0}{a}}=\frac{T}{\sqrt{X_N}-\sqrt{X_0}}$

A (17.) egyenletből $n=0$ esetén azt kapjuk, hogy

(22.) $\sqrt{X_0}=\sqrt{X_N}*\frac{\omega_0}{\omega_0+a*T}$.

Ennek felhasználásával:

(23.) $\sqrt{\frac{2*K*\varphi_0}{a}}=\frac{T}{\sqrt{X_N}-\sqrt{X_0}}=\frac{T}{\sqrt{X_N}*(1-\frac{\omega_0}{\omega_0+a*T})}=\frac{T}{\sqrt{X_N}*\frac{a*T}{\omega_0+a*T}}=\frac{\frac{\omega_0}{a}+T}{\sqrt{X_N}}$

Felhasználói szempontból viszont kényelmesebb az ω_0 szögsebesség és az a szöggyorsulás megadása helyett a kezdő és a maximális léptetési frekvenciát (f_{min} , f_{max}) megadni. Ezek segítségével az ω_0 és az a a következőképpen írható fel:

(24.) $a=\frac{(f_{max}-f_{min})*\varphi_0}{T}$

(25.) $\omega_0 = f_{min} * \varphi_0$

A (23.) egyenletből a következő összefüggéseket kapjuk:

(26.) $$\frac{\omega_0}{a} + T = T * \left(\frac{f_{min} * \varphi_0}{(f_{max} - f_{min}) * \varphi_0} + 1 \right) = \frac{T * f_{max}}{f_{max} - f_{min}}$$

(27.) $$\frac{2 * K * \varphi_0}{a} = \frac{2 * K * \varphi_0 * T}{(f_{max} - f_{min}) * \varphi_0} = \frac{2 * K * T}{f_{max} - f_{min}}$$

A (23.) egyenlet felhasználásával az alábbi összefüggéseket kapjuk az X_N és K lehetséges értékeire:

(28.) $$\frac{\left(\frac{\omega_0}{a} + T \right)^2}{X_N} = \frac{2 * K * \varphi_0}{a}$$

(29.) $$\frac{T^2 * f_{max}^2}{(f_{max} - f_{min})^2 * X_N} = \frac{2 * K * T}{f_{max} - f_{min}}$$

(30.) $$\frac{T * f_{max}^2}{(f_{max} - f_{min}) * X_N} = 2 * K$$

(31.) $K * X_N = \frac{T * f_{max}^2}{2 * (f_{max} - f_{min})}$ és $K \geq 1$ egész, valamint $X_N \leq 255$

Írjuk fel az X_n és T_n meghatározására kapott egyenleteket a (23.), (24.) és (25.) összefüggések felhasználásával!

(32.) $$T_n = \sqrt{\frac{2 * K * \varphi_0}{a}} * [\sqrt{X_{n+1}} - \sqrt{X_n}] = \frac{\frac{\omega_0}{a} + T}{\sqrt{X_N}} * [\sqrt{X_{n+1}} - \sqrt{X_n}] = \frac{\frac{T * f_{max}}{f_{max} - f_{min}}}{\sqrt{X_n}} * [\sqrt{X_{n+1}} - \sqrt{X_n}]$$

(33.) $T_n = T_c * [\sqrt{X_{n+1}} - X_n]$, ahol $T_c = \frac{T * f_{max}}{\sqrt{X_n} * (f_{max} - f_{min})}$

(34.)
$$X_n=[\sqrt{\frac{a}{2*K*\varphi_0}}*(t_n+\frac{\omega_0}{a})]^2=[\frac{\sqrt{X_N}}{\frac{\omega_0}{a}+T}]^2=$$
$$=X_N*[\frac{f_{max}-f_{min}}{T*f_{max}}*(t_n+\frac{T*f_{min}*\varphi_0}{(f_{max}-f_{min})*\varphi_0})]^2=X_N*[\frac{t_n*(f_{max}-f_{min})+T*f_{min}}{T*f_{max}}]^2$$

Tehát az X_n értékét az X_N leosztásával kapjuk a következőképpen:

(35.) $X_n=\frac{X_N}{(\frac{T*f_{max}}{t_n*(f_{max}-f_{min})+T*f_{min}})^2}$, ahol $t_0=0$ és $t_n=T_0+T_1+\ldots+T_{n-1}$

A $[\sqrt{X_N+1}-\sqrt{X_N}]$ kifejezés értékeit táblázatban tároljuk $X_n=0,1,2,\ldots,255$ értékeknél. A táblázat elemei kétbájtos egészek, ezért az

(36.) $M*[\sqrt{X_N+1}-\sqrt{X_N}]$

értéket tároljuk le, és az M értékét olyan nagyra választjuk, hogy ne legyen két egyforma érték, mivel 256 érték tárolása esetén nem célszerű redundáns adatokat tárolni a mikrovezérlő amúgy is szűkös memóriájában. Jelöljük a táblázat elemeit A_n-el . Ezen táblázat segítségével a négyzetgyökös kifejezéseket az alábbiak szerint határozhatjuk meg:

(37.) $[\sqrt{X_N+1}-\sqrt{X_N}]=\frac{1}{M}*A_n$

(38.) $\sqrt{X}=\frac{1}{M}*\sum_{i=0}^{X-1}A_i$, ahol $0\le X\le 255$ egész érték.

Ennek a táblázatnak az alkalmazása megkíméli a mikrokontroller processzorát a gyökvonás műveletének elvégzésétől, mely igen sok órajelet igényelne, továbbá mindenképp el akarjuk kerülni a lebegőpontos aritmetikai műveleteket, mivel ehhez terjedelmes könyvtári függvények kellenek és a mikrokontrollernek meglehetősen szűkös a programmemóriája.

A képleteket a dolgozat megírásához használt OpenOffice.org 3.2 szövegszerkesztőjében (Writer) található képletszerkesztővel készítettem.

4.5. Az általam készített kódok

Az M értékének meghatározására és a táblázatba kerülő gyökértékek legenerálására

a 7.1-es függelékben közölt *SqrtGen.java* programot írtam meg, mely a (36.) képlet alapján dolgozik és az M értéket úgy határozza meg, hogy ne legyen a táblázatban két azonos érték. A program kimenete a 7.2-es függelékben látható. A legenerált értékek a *motor.h* fájlban találhatóak.

```
#define MOTOR_SQRTARRAY_MULTIPLIER              ((u16_t)(14656))
const u16_t code m_SquareRootArray[256] =
{
0x3940, 0x17b7, 0x1232, 0x0f57, 0x0d84, 0x0c38, 0x0b3c, 0x0a75,
…
}
```

További feladatom annak a függvénynek a megírása volt, amelyik kiszámolja a vezérléshez szükséges paramétereket. A függvény paraméterként kapja meg a felhasználó által megadott az induló-, és a maximális léptetési frekvenciát, valamint a gyorsítási szakasz idejét. A függvény visszatérési értéke vagy nulla, vagy egy hibakód.

```
u8_t motor_calculate_params(u16_t wFrqMin, u16_t wFrqMax, u16_t wT)
{
…
        return(0);
}
```

Az első lépés a paraméterek ellenőrzése. A határértékek és a hibakódok értékei a *motor.h* fájlban találhatóak.

```
…
        if((wFrqMin == 0) || (wFrqMin > MOTOR_PARAMS_MAX_FRQ))
                return(MOTOR_PARAMS_ERRCODE_FRQMIN);
        if((wFrqMax < wFrqMin) || (wFrqMax > MOTOR_PARAMS_MAX_FRQ))
                return(MOTOR_PARAMS_ERRCODE_FRQMAX);
        if((wT == 0) && (wFrqMax != wFrqMin))
                return(MOTOR_PARAMS_ERRCODE_T);
…
```

A következő részben azt az egyszerű esetet kezelem le, amikor a motort állandó frekvenciával kell vezérelni. Ilyenkor az induló-, és a maximális léptetési frekvencia megegyezik.

```
…
        if(wFrqMax == wFrqMin) {
                m_wParam_FrqMin = wFrqMin;
                m_wParam_FrqMax = wFrqMax;
                m_wParam_T = 0;
                m_bParam_N = 0;
                m_bParam_xN = 0;
                m_bParam_K = 1;
                m_dwParam_Tc = 1000000;             // 1 000 000 us  == 1 sec
                m_dwParam_Tc /= wFrqMin;

                calculate_pca_cycle(0);         // dwPcaCycle=...

                m_PcaHighArray_xn[0] = dwPcaCycle >> 16;
                wTemp = dwPcaCycle;
                if(wTemp < MOTOR_PARAMS_MIN_PCA)
                        wTemp = MOTOR_PARAMS_MIN_PCA;
                m_PcaLowArray_xn[0] = wTemp;

                i = 1;
                do {
                        m_PcaHighArray_xn[i] = m_PcaHighArray_xn[0];
                        m_PcaLowArray_xn[i] = m_PcaLowArray_xn[0];
                        i++;
                } while(i != 0);
                return(0);
        }
…
```

A következő lépés a (31.) egyenlet kiszámolása. Meg kell keresni a $K \geq 1$ egész, valamint $X_N \leq 255$ legnagyobb megfelelő értékét.

```
…
//              T * fmax * fmax
//       xN = ----------------------
//               K * 2 * ( fmax - fmin )

        dwTemp1 = wFrqMax;
        dwTemp1 *= wFrqMax;
        wTemp = wFrqMax;
        wTemp -= wFrqMin;
        dwTemp1 /= wTemp;
        dwTemp1 *= wT;
        dwTemp1 /= 1000;              // mivel a T 'ms' -ban van
        dwTemp1 /= 2;

        dwTemp2 = dwTemp1;                  // ez a K*xN ertek
        bK = 1;
        while(dwTemp1 > 255) {
                if(bK == 255)
                        return(MOTOR_PARAMS_ERRCODE_K);
                bK++;
                dwTemp1 = dwTemp2 / bK;
        }
        bxN = dwTemp1;
…
```

Ha sikerült meghatározni, akkor a bK és a bxN lokális változókban vannak az új értékek. A globális változókba csak akkor írom át, ha a többi paramétert is meg lehet határozni. A következő lépés a (33.) egyenletben szereplő T_c meghatározása.

```
…
//                      T * fmax
//        Tc = --------------------------
//                sqrt(xN) * ( fmax - fmin )
//
//        A Tc erteket us-os egysegben tarolom,azaz m_dwParam_Tc = Tc * 10e6

        wTemp = wFrqMax;
        wTemp -= wFrqMin;
        dwTc = wT;
        dwTc *= wFrqMax;
        dwTc *= 1000;                           // T  'ms'-ban van !!
        dwTc /= motor_calculate_sqrt(bxN);
        dwTc /= wTemp;

        if(dwTc > MOTOR_PARAMS_MAX_TC)  {
                return(MOTOR_PARAMS_ERRCODE_MAXTc);
        }
…
```

Ezek után a (35.) egyenletnek megfelelően sorban kiszámolom az egyes T_n értékeit addig, amíg az összegük el nem éri a felhasználó által kívánt gyorsítási időt. A program gyorsítása érdekében a PCA beállításához szükséges értékeket két bájtos tömbben tárolom, így majd a megszakítási rutinban gyorsan elő lehet venni a következő értéket.

A fenti számításhoz szükség van a (37.) egyenletben meghatározott értékre, amit egy külön függvényben valósítottam meg.

```
u16_t motor_calculate_sqrt(u8_t n)
{
        u32_t sum;
        u8_t i;
        if(n == 0)
```

```
        return(0);
    if(n == 1)
        return(1);
    sum = MOTOR_SQRTARRAY_MULTIPLIER;
    for(i=1; i<=(n-1); i++) {
        sum += m_SquareRootArray[i];
    }
    sum /= MOTOR_SQRTARRAY_MULTIPLIER;
    return((u16_t)sum);
}
```

A *motor.h* és *motor.c* fájlokat hosszuk miatt nem közöltem a függelékben, de a szakdolgozathoz tartozó CD mellékletben megtalálhatóak.

5. Összefoglalás

A bevető részben leirtam, hogy egy nagyobb projektbe kapcsolódtunk be. A szakdolgozat alapjául szolgáló mikrokontroller alapú léptetőmotor vezérlő program a Debreceni Egyetem Orvos- és Egészségügyi Centrum (Klinikák) területén található Pet-CT Diagnosztikai Kft-nek készült. A program egy CT berendezés ágy részét mozgatja, melyre a radioaktív izotóppal befecskendezett kísérleti állatok kerülnek. A szakdolgozat megírásának fő célja számomra az volt, hogy megismerkedhessem a mikrokontrollerek programozásával és megérrezhessem milyen egy csoportos munkában való részvétel: milyen kötöttségekkel, milyen nehézségekkel és milyen előnyökkel jár.

Dolgozatom következő része a mikrokontrollerek bemutatásáról szólt. Szó volt a felépítésükről, előnyeikről és korlátaikról, és hogy miben különböznek más, általános célú mikroprocesszoroktól. Csak a legfontosabb dolgokat igyekeztem kiemelni, nem törekedtem a részletekbe menő ismertetésre.

Következő rész a léptetőmotorokról szól. A léptetőmotorok kefe nélküli szinkron villanymotorok, melyeket digitális jellel kell vezérelni, és a jel hatására az adott pozícióba fordul. Olyan környezetben használatosak, ahol fontos a pontos pozícionálás és az adott pozíció ismerete. Ismertettem a három fő felépítési típusukat (változó reluktancia, állandó mágneses és hibrid léptetőmotorok), a két nagy vezérlési elvet (unipoláris és bipoláris vezérlés) és 4 meghajtási módot (egyfázisú, egészlépéses, féllépéses, mikroléptetéses mód). Részletesebb ismertetésre törekedtem, hogy aki a dolgozatomnak csak ezen részét kívánja elolvasni, az is viszonylag teljes képet kaphasson a léptetőmotorok világáról, ezért minden alfejezetben ismertettem az előnyöket és hátrányokat is.

Dolgozatom fő fejezete a negyedik. Itt fejtettem ki a szakdolgozat alapjául szolgáló program továbbfejlesztésének részleteit. A számítások, melyeket levezettem, nem igényeltek bonyolult fizikai és matematikai ismereteket, mégis az egész-aritmetika, ami a mikrovezérlő szerény matematikai képessége miatt volt szükséges, sok problémát okozott. Ebben a részben ismertettem néhány kódrészletemet, melyek a vezérlő prograba lettek beintegrálva. Egy olyan függvény megírása volt a feladatom, mely a vezérléshez szükséges paramétereket számolja ki. A függvény paraméterként kapja meg a felhasználó által megadott induló-, és maximális

léptetési frekvenciát, valamint a gyorsítási szakasz idejét. Ehhez még egy függvény megírására volt szükség, mely a (37.) egyenletben meghatározott értéket számítja ki. Az *SqrtGen.java* fájlról is esett szó, melyet az M szorzótényező és a négyzetgyök értékek meghatározására készítettem.

Ez a szakdolgozat nagyon értékes dolgokkal szolgált számomra. Megismerhettem a mikrokontrollerek programozásának fogásait, melyekkel ezeket az igen szerénynek mondható erőforrásokkal rendelkező eszközöket igen komoly feladatok elvégzésére lehet felprogramozni. Továbbá egy szoftverfejlesztés menetébe is betekintést nyerhettem, mely igen hasznos lehet majd későbbi munkám során. Végül, de nem utolsó sorban az elmúlt szűk egy évben még jobban megszerettem a mikrokontrollerekkel való munkát és remélem, hogy a jövőben adódik lehetőségem mikrovezérlőkkel történő fejlesztésekre.

6. Irodalomjegyzék

Szakirodalom:

1. Kónya László-Kopják József: *PIC mikrovezérlők alkalmazástechnikája, PIC programozás C nyelven*, Harmadik, bővített, átdolgozott kiadás, ChipCAD Elektronikai Disztribúció Kft., Budapest,2009

Felhasznált adatlapok és alkalmazás leírások:

1. Silicon Labs AN155 Stepper Motor Reference Design:

   http://www.silabs.com/Support%20Documents/TechnicalDocs/an155.pdf

2. C8051F12x devices Data Sheet:

   http://www.kemt.fei.tuke.sk/predmety/KEMT411_ESM/_materialy/Prednasky/51_kompat/c8051f12x.pdf

3. C8051F120 Data Sheet (short):

   http://www.silabs.com/Support%20Documents/TechnicalDocs/C8051F120_Short.pdf

Internetes források:

Magyar:

1. http://www.mogi.bme.hu/letoltes/ERZEKELOK%20ES%20MUKODTETOK/M%C5%B1k%C3%B6dtet%C5%91k.pdf
2. http://www.sze.hu/~csizm/Gepipari%20mernokasszisztens_Anyagismeret/Masodik.ppt
3. http://www.hobbielektronika.hu/cikkek/leptetomotorvezerles_elmeletben_oldal1.html
4. http://www.hobbielektronika.hu/cikkek/leptetomotorvezerles_elmeletben_oldal2.html
5. http://www.hobbielektronika.hu/cikkek/leptetomotorvezerles_elmeletben_oldal3.html
6. http://www.hobbielektronika.hu/cikkek/leptetomotorvezerles_elmeletben_oldal4.html
7. http://www.hobbielektronika.hu/cikkek/leptetomotorvezerles_elmeletben_oldal5.html
8. http://www.hobbielektronika.hu/cikkek/leptetomotorvezerles_elmeletben_oldal6.html

9. http://www.hobbielektronika.hu/cikkek/leptetomotorvezerles_elmeletben_oldal7.html
10. http://www.hobbielektronika.hu/cikkek/leptetomotorvezerles_elmeletben_oldal8.html
11. http://www.hobbielektronika.hu/cikkek/leptetomotorvezerles_elmeletben_oldal9.html
12. http://uc.hobbielektronika.hu/kapcsolasok_unipolaris_leptetomotor_vezerlese_gyakorlatban_avr_segitsegevel_oldal1.html
13. http://uc.hobbielektronika.hu/kapcsolasok_unipolaris_leptetomotor_vezerlese_gyakorlatban_avr_segitsegevel_oldal2.html
14. http://uc.hobbielektronika.hu/kapcsolasok_unipolaris_leptetomotor_vezerlese_gyakorlatban_avr_segitsegevel_oldal3.html
15. http://qtp.hu/elektro/leptetomotor_mukodese.php
16. http://www.hun.cncdrive.com/tudasbazis/stp.htm
17. http://hu.wikipedia.org/wiki/Mikrokontroller
18. http://itl7.elte.hu/hlabdb/pic/pic.html
19. http://eki.sze.hu/ejegyzet/ejegyzet/gyimesi/10.htm
20. http://wiki.ham.hu/index.php/PLL_szint%C3%A9zer

Angol:

1. http://en.wikipedia.org/wiki/Stepper_motor
2. http://www.silabs.com/products/mcu/Pages/MotorControlApplications.aspx
3. http://electronics.howstuffworks.com/microcontroller.htm
4. http://electronics.howstuffworks.com/microcontroller1.htm
5. http://www.solarbotics.net/library/pdflib/pdf/motorbas.pdf
6. http://en.wikipedia.org/wiki/Phase-locked_loop

7. Függelék

7.1. SqrtGen.java

/*A számításaim alapján az M szorzófaktort és a négyzetgyök értékeket meghatározó program*/

/*NetBeans IDE 6.8*/

```
public class SqrtGen {

    private static int Calculate_Sqrt(int n, int M) {
        double sqrt;
        sqrt = Math.sqrt(n + 1) - Math.sqrt(n);
        sqrt *= M;
        return (int) Math.round(sqrt);
    }

    private static int Calculate_M() {
        int M = 1;
        int sqrt1 = 0, sqrt2 = 0;
        int i = 0;
        while (i <= 255) {
            if (sqrt1 == sqrt2) {
                M++;
                i = 0;
                sqrt1 = Calculate_Sqrt(i, M);
            }
            sqrt2 = sqrt1;
            i++;
            sqrt1 = Calculate_Sqrt(i, M);
        }
        return M;
```

```
    }

    public static void main(String[] args) {
        int sqrt_M;
        sqrt_M = Calculate_M();
        System.out.println("M=" + sqrt_M);
        System.out.println("A táblázat értékei: " + "\n--------------------");
        for (int i = 0, j = 0; i <= 255; i++) {
            System.out.printf("0x%04x, ", Calculate_Sqrt(i, sqrt_M));
            if (++j == 8) {
                j = 0;
                System.out.println();
            }
        }
    }
}
```

7.2. Az SqrtGen.java kimenete

M=14656

A táblázat értékei:

0x3940, 0x17b7, 0x1232, 0x0f57, 0x0d84, 0x0c38, 0x0b3c, 0x0a75,
0x09d3, 0x094a, 0x08d6, 0x0871, 0x0819, 0x07cb, 0x0785, 0x0746,
0x070c, 0x06d8, 0x06a8, 0x067c, 0x0653, 0x062d, 0x0609, 0x05e8,
0x05c9, 0x05ab, 0x0590, 0x0575, 0x055d, 0x0545, 0x052f, 0x051a,
0x0505, 0x04f2, 0x04e0, 0x04ce, 0x04bd, 0x04ad, 0x049d, 0x048e,
0x0480, 0x0472, 0x0464, 0x0457, 0x044b, 0x043e, 0x0433, 0x0427,
0x041c, 0x0412, 0x0407, 0x03fd, 0x03f3, 0x03ea, 0x03e1, 0x03d8,
0x03cf, 0x03c6, 0x03be, 0x03b6, 0x03ae, 0x03a6, 0x039f, 0x0398,
0x0390, 0x0389, 0x0383, 0x037c, 0x0375, 0x036f, 0x0369, 0x0363,
0x035d, 0x0357, 0x0351, 0x034b, 0x0346, 0x0340, 0x033b, 0x0336,
0x0331, 0x032c, 0x0327, 0x0322, 0x031d, 0x0319, 0x0314, 0x030f,
0x030b, 0x0307, 0x0302, 0x02fe, 0x02fa, 0x02f6, 0x02f2, 0x02ee,
0x02ea, 0x02e6, 0x02e2, 0x02df, 0x02db, 0x02d7, 0x02d4, 0x02d0,
0x02cd, 0x02c9, 0x02c6, 0x02c3, 0x02c0, 0x02bc, 0x02b9, 0x02b6,
0x02b3, 0x02b0, 0x02ad, 0x02aa, 0x02a7, 0x02a4, 0x02a1, 0x029e,
0x029c, 0x0299, 0x0296, 0x0293, 0x0291, 0x028e, 0x028c, 0x0289,
0x0286, 0x0284, 0x0281, 0x027f, 0x027d, 0x027a, 0x0278, 0x0276,
0x0273, 0x0271, 0x026f, 0x026c, 0x026a, 0x0268, 0x0266, 0x0264,
0x0262, 0x0260, 0x025d, 0x025b, 0x0259, 0x0257, 0x0255, 0x0253,
0x0251, 0x024f, 0x024e, 0x024c, 0x024a, 0x0248, 0x0246, 0x0244,
0x0242, 0x0241, 0x023f, 0x023d, 0x023b, 0x023a, 0x0238, 0x0236,
0x0235, 0x0233, 0x0231, 0x0230, 0x022e, 0x022c, 0x022b, 0x0229,
0x0228, 0x0226, 0x0224, 0x0223, 0x0221, 0x0220, 0x021e, 0x021d,
0x021b, 0x021a, 0x0219, 0x0217, 0x0216, 0x0214, 0x0213, 0x0212,
0x0210, 0x020f, 0x020d, 0x020c, 0x020b, 0x0209, 0x0208, 0x0207,
0x0206, 0x0204, 0x0203, 0x0202, 0x0200, 0x01ff, 0x01fe, 0x01fd,
0x01fb, 0x01fa, 0x01f9, 0x01f8, 0x01f7, 0x01f6, 0x01f4, 0x01f3,

0x01f2, 0x01f1, 0x01f0, 0x01ef, 0x01ed, 0x01ec, 0x01eb, 0x01ea,
0x01e9, 0x01e8, 0x01e7, 0x01e6, 0x01e5, 0x01e4, 0x01e3, 0x01e2,
0x01e1, 0x01e0, 0x01df, 0x01de, 0x01dd, 0x01dc, 0x01db, 0x01da,
0x01d9, 0x01d8, 0x01d7, 0x01d6, 0x01d5, 0x01d4, 0x01d3, 0x01d2,
0x01d1, 0x01d0, 0x01cf, 0x01ce, 0x01cd, 0x01cc, 0x01cb, 0x01ca,

8. Köszönetnyílvánítás

Elsősorban szeretném megköszönni témavezetőnknek, Szabó Zsoltnak, hogy mindig készséggel állt rendelkezésünkre, nagy türelemmel és maximális szakmai hozzáértéssel magyarázta el a bennünk felmerülő kérdésekre a választ, valamint szabadidejéből sokat áldozva nézte át dolgozatunkat és javaslatokat tett az esetleges módosításokra.

Másodsorban, de mégis kiemelt köszönet illeti Kígyósi Tibort, akivel sokat konzultálva alakítottuk szakdolgozatunkat és mindig segítőkészen állt rendelkezésemre.