

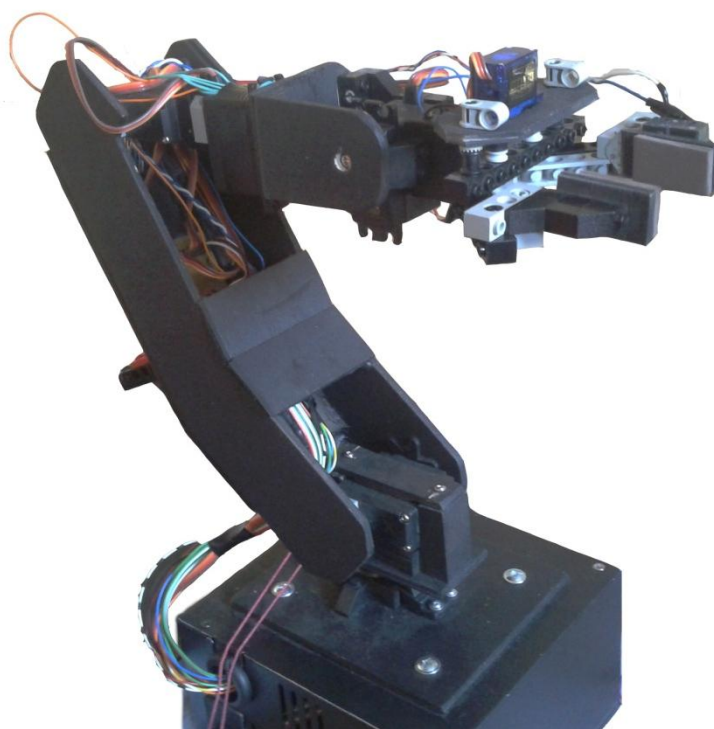
Óbudai Egyetem

Bánki Donát Gépész és Biztonságtechnikai Mérnöki Kar
Mechatronikai és Autótechnikai Intézet



TUDOMÁNYOS DIÁKKÖRI DOLGOZAT

HUMANOID ROBOTKAR ÉPÍTÉSE ÉS PROGRAMOZÁSA



Szerző:

Nagymáté Gergely

Mechatronikai mérnöki szak

Konzulens:

Dr. Bencsik Attila

Egyetemi docens

Budapest, 2012

Tartalom

Ábrajegyzék	3
1. Bevezető.....	4
2. Áttekintés	5
3. A robotkar felépítése	6
3.1 Mechanikai méretezés	6
3.2 Megfogó mechanika.....	7
3.3 A szervó motorok működése	7
3.4 A robot vezérlőpanelje	9
4. A vezérlő program működése	11
4.1 Vezérlőjelek előállítása mikrokontrollerrel	11
4.2 Vezérlőjel kivitelezése megszakításokkal több szervóra	12
4.3 Szervó pozíciók beolvasása	14
4.4 Kommunikálás LabVIEW-val	15
4.5 A megfogó vezérlése.....	16
4.6 Programozható Point To Point mozgások	17
5. A robot kinematikája	19
5.1 A direkt kinematikai feladat megoldása.....	19
5.2 A robot inverz kinematikája	21
6. A robot, mint egy automatizálási rendszer része	25
7. A robot vezérlése 3D kapacitív interfésszel.....	27
8. Összefoglalás és következtetések.....	29
Irodalomjegyzék.....	30
Mellékletek	31
I. Melléklet: A robot szervóinak választása és mechanikai ellenőrzésük	31
II. Melléklet: A vezérlőpanel és a kábeleinek kiosztása	32
III. Melléklet: A robot direkt kinematikája	33
IV. Melléklet: A szortírozó vezérlésének folyamatábrája	34

Ábrajegyzék

1. ábra: A robotrendszer áttekintése	5
2. ábra: A megfogó mechanizmus	7
3. ábra: Egy RC szervó felépítése	8
4. ábra: A szervó vezérlő jele	9
5. ábra: Vezérlőpanel beépítés előtt	10
6. ábra: Három szervó vezérlése megszakításokkal	12
7. ábra: A vezérlés blokkvázlata	18
9. ábra: Az inverz kinematika analitikus megoldása	22
10. ábra: A szortírozó áttekintése	26
11. ábra: 3D interfész kapcsolása	28

1. Bevezető

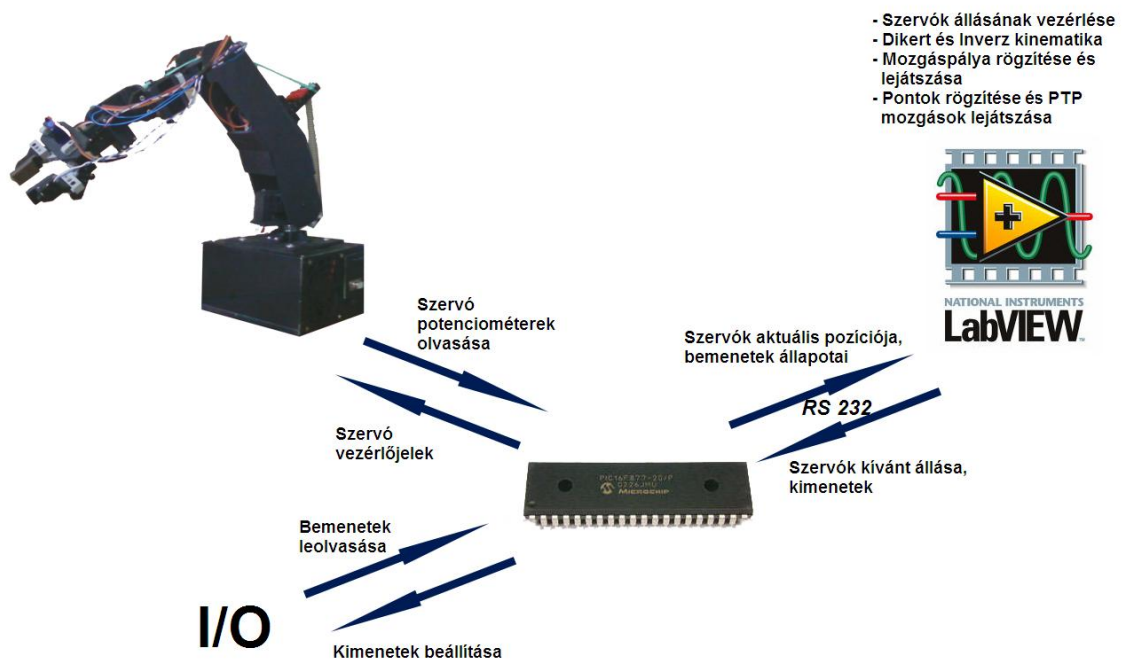
E dolgozat egy hat tengelyes robotkar építési tapasztalatait és működését ismerteti. Ez a kis robot akár meg is felelhet az ipari robot definíciójának, mely szerint „Az ipari robotok automatikus helyzetvezérelt, újraprogramozható, többcélú, több szabadságfokú eszközök. Alkalmasak különféle anyagok, szerszámok, alkatrészek kezelésére, különböző célfeladatok megvalósítására, továbbá különféle mozgásokra.” [1] A robot rendelkezik be- és kimenetekkel így beilleszthető egy irányítástechnikai rendszerbe is, és valóban képes kis tömegű tárgyak mozgatására. Ennek a robotnak fő funkciója azonban mégsem az, hogy ipari környezetben végezzen munkát, arra nem is lenne képes. Ez egy oktatási célt szolgáló robotkar, mely elsősorban az építője szakmai fejlődését segítette, de jó példával járhat elől más diákok számára is, akik tanulhatnak ebből a dolgozattól és mellékleteiből is. A mérnöki tanulmányok alatt hatalmas jelentősége van annak, amikor a hallgató önszántából, szabadidejében foglalkozik a szakmához kapcsolódóan valaminek a megépítésével, problémákat old meg és látja végül munkájának a gyümölcsét működni. Szeretném hát ezzel a dolgozattal hallgatótársaimat is ilyesfajta építésre ösztönözni, mely garantáltan hozzájárul szakmai fejlődésükhöz és mérnöki önértékelésük fejlődéséhez.

A dolgozat először áttekinti a robotot, mint rendszert, majd részletesebben ismerteti a fizikai felépítését és vezérlőprogramját. Sokat foglalkozik a PIC16F877-es mikrokontrollerrel és az abban futó programmal, ami lényeges része a robotnak. A működés jobb megértése érdekében a leírt dolgokat képek szemléltetik. Ezzel párhuzamosan foglalkozik majd a LabVIEW környezetben írt vezérlőprogrammal is, ami a valódi vezérlést valósítja meg. Ezt követően a robot kinematikájával is foglalkozik a dolgozat. A dolgozat végén egy működő automatizálási rendszer részeként tekintjük a robotot egy szortírozó folyamat elvégzésére programozott környezetben. Legvégül egy háromdimenziós érintésnélküli interfész kerül ismertetésre a robot irányításának egy lehetőségeként.

A dolgozat megírásával kapcsolatban tanácsokkal látott el Dr. Bencsik Attila. A táp és a soros kommunikáció kivitelezéséhez adtak ötletet Pál Márton és Csővári László mechatronikai mérnök hallgatótársaim. Ez úton szeretném mindnyájuknak megköszönni a segítséget.

2. Áttekintés

Mielőtt a részletekbe merülnénk, tekintsük át, hogy hogyan is működik ez a robotkar, melyek a főbb részei. Az 1. ábra szemlélteti vázlatosan a rendszer működését. Maga a robotkar egy falemezekből és LEGO elemekből épített szerkezet, melynek csuklóiban a távirányítható modell járművekben is megtalálható szervó motorok vannak. Ezek a motorok a robot vezérlőpaneljére vannak kötve, amelynek lényege egy PIC 16F877 mikrokontroller. Ez a mikrokontroller állítja elő az egyes szervók vezérlő jeleit, és olvassa be a szervók jelenlegi állását, valamint működteti a robot ki és bemeneteit, amelyek lehetővé teszik más rendszerekkel történő összekapcsolását. A PIC RS232 szabványt használó soros buszon kommunikál a számítógéppel, amelyen az igazi vezérlő program fut, ami meghatározza, hogy milyen állásban legyenek a szervók. Ez a program valósítja meg az ember-gép kommunikációt is, ugyanis egy virtuális vezérlőpanelt mutat a felhasználónak. A program a National Instruments cég által fejlesztett LabVIEW környezetben íródott, ami egyszerűen használható kommunikációs lehetőségeket kínál külső eszközökkel pl. soros porton keresztül. A LabVIEW program végzi továbbá a direkt és inverz kinematikai számításokat, mozgáspálya rögzítését és lejátszását, betanított pontok közti PTP (Point To Point) mozgások lejátszását a meghatározott idő alatt, valamint az e műveletek alapján meghatározott csuklókoordináták pillanatnyi értékét elküldi a PIC számára.



1. ábra: A robotrendszer áttekintése

3. A robotkar felépítése

A robot javarészt rétegelt falemezből készült, valamint az alkaron maguk a szervók alkotják a kar szerkezetét. A szervók a csuklókbán helyezkednek el és direkt hajtást valósítanak meg. A csuklókbán az egyik féltengely a szervó kimenő csapja, míg a másik oldalon LEGO elempár helyezkedik el siklócsapágyazást megvalósítva. A LEGO elempár itt könnyen bontható kötést is biztosít, ami megkönnyíti a szerelési-javítási munkát. A LEGO elemek a szervókhoz és fához valamint a falemezek egymáshoz és a szervókarokhoz cianoakrilát ragasztóval lettek rögzítve.

3.1 Mechanikai méretezés

A szervók választásánál a legfontosabb szempont a nyomaték, amivel tartani képesek. Az RC modellezésben használt szervók nyomatékát kg*cm-ben szokták megadni. Becslés alapján az I./a mellékletben felsorolt szervók lettek a robothoz választva, melyek súly és nyomaték adatainak ismeretében ellenőrző számítást lehetett végezni a robotkarra, mint tartó szerkezetre (Melléklet I./b).

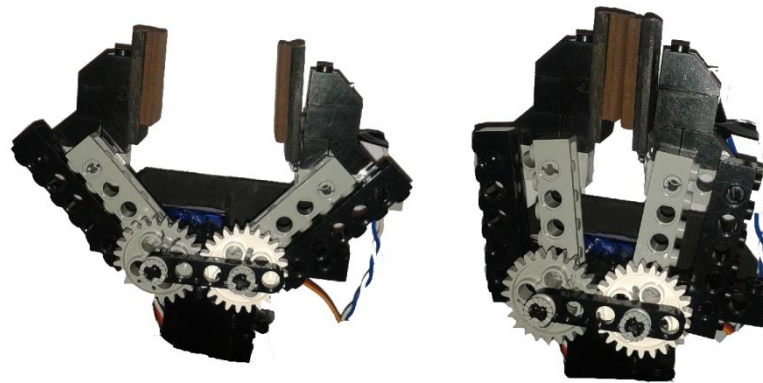
A számítás igazolta ezeket a szervókat, így képes a robot apróbb tárgyak felemelésére is, de saját súlyát mindenképp elbírja. Tekintve, hogy ez a robot és annak elkészítése oktatási célt szolgál, ez elegendő is. Így is megfigyelhető azonban a roboton egy erőteljes rángatózás, amikor relatíve nagy nyomaték ellen kell a szervóknak pozíciójukat tartani (pl. vízszintesen kinyújtott kar esetén). Ennek az az oka, hogy csak addig fejtenek ki nyomatékot, amíg vezérlőjelük logikai magas állapotban van (lásd 3.3). Így a vezérlőjel peridusának kitöltetlen idejében a kar zuhan, majd ismét helyére pozicionál a következő ciklusban. Ez nagyon gyorsan történik, és rázkódásban nyilvánul meg a roboton. Ezt ellensúlyozandó egy rugó is segíti a robotkart a pozíciótartásban, ami akkor feszül meg leginkább, amikor a kar előre fele dől, tehát legnagyobb az erőkar. Az alkart hasonló módon a felkarhoz rögzített hagyományos befőttes gumi könnyíti.

A robot alapját képező fémház tetején a robotkar egy axiális golyóscsapágyra fekszik fel, hiszen az első szervó csapja egy 5 mm átmérőjű finoman bordázott műanyag tengely, ami önmagában nem bírná megtartani a robotkart az eldőléstől. Hiába van a szervókar abba belecsavarozva, a műanyag roncsolódna és elengedne. A nagyobb átmérőjű axiális csapágy sokat segít ezen.

Szintén egy golyóscsapágy van a 4. szervó és az 5. szervót tartó villás kialakítású szerkezet között, ám itt annak célja a műanyag tengelyen fellépő nyíróerő csökkentése. Ez egy radiális csapágy. Hasonló körülmények a hatodik szervónál vannak, itt azonban csak a megfogó súlya nehezedik a csapra. Itt nem indokolt könnyítő csapágyazás

3.2 Megfogó mechanika

A megfogó LEGO elemekből épült és mechanizmusának lényege az, hogy a megfogó felületek mozgás alatt és végállásokban is végig párhuzamosak maradnak. A két karpár belső elemei a fogaskerekhez vannak ragasztva, ezért azzal együtt fordulnak. A szervó az egyik fogaskereket hajtja, míg a másik ezzel szimmetrikusan forog. A szervó túlerőltetésének elkerülése miatt egy kontakt-érzékelő jelzi, amikor a megfogó bezárult vagy megfogott valamit. Ilyenkor az összezáródás megáll. Ebből következik, hogy csak egy bizonyos erővel képes szorítani, aminél már benyomódik a kontaktérzékelő. A megfogó felületek egy szivacsos réteggel vannak ellátva, ezáltal elősegítve a jobb tapadást a felületükön. A megfogó mechanizmust a 2. ábra szemlélteti.



2. ábra: A megfogó mechanizmus

3.3 A szervó motorok működése

A szervó motorok könnyű használhatóságuk miatt igen elterjedtek a modellezők körében. Szinte minden RC autó, repülő vagy helikopter tartalmaz legalább egy szervó motort. A szervó lényege, hogy nem folytonos forgó mozgást produkál, hanem egy tartományon belüli pozicionálást tesz lehetővé. A robot szempontjából a szervók nagy előnye más motorokkal szemben (DC, léptető motor), hogy nincs szükség H-hídra és külső áramkörre, mert ezek a szervón belül találhatóak meg. Valamint az is előnyös, hogy a nagy nyomaték elérése már eleve biztosított a belső fogaskerékes áttételek miatt, így azt sem kell a motoron kívül létrehozni.

A legtöbb szervó felépítése gyártótól függetlenül megegyezik. Az 3. ábra metszetben szemlélteti egy RC szervó belső felépítését.



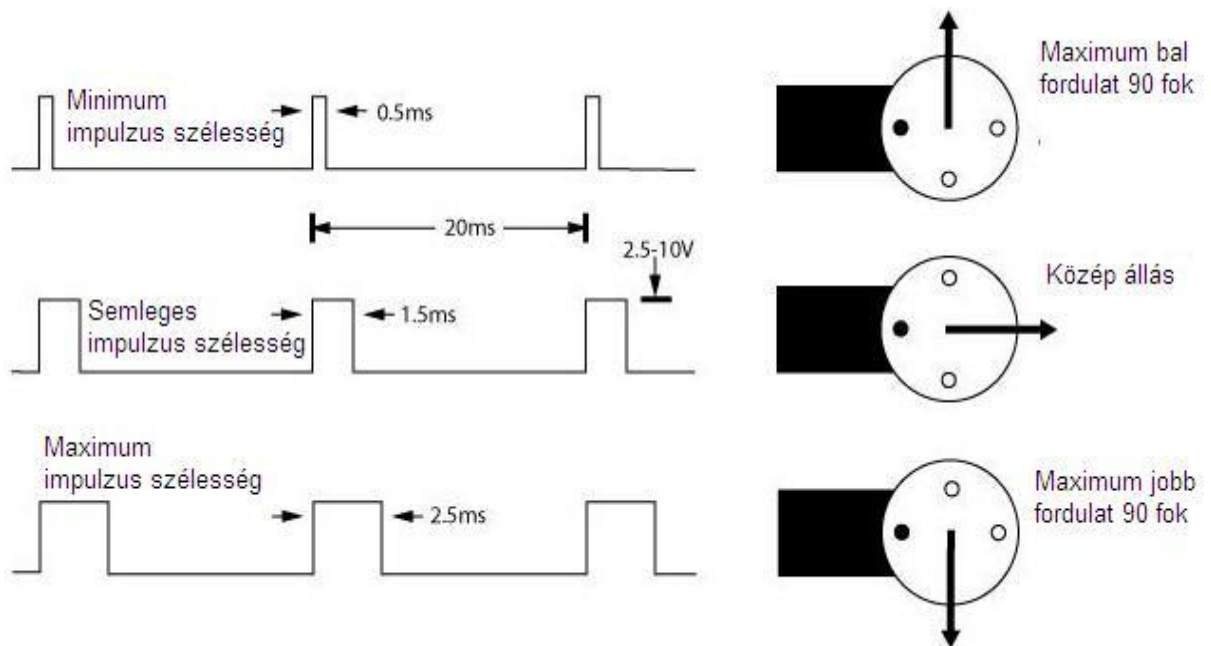
3. ábra: Egy RC szervó felépítése¹

Alul látható a vezérlő áramkör, amely értelmezi a vezérlőjelet és megfelelő irányú forgásra kapcsolja a motort, ha szükséges. Nem lehet egyértelműen látni, de van a szervóban még egy potenciométer is (más néven egy változtatható értékű ellenállás), amely a pozíció visszajelzésére szolgál a vezérlő elektronika számára.[2] A robotkarhoz használt szervók úgy lettek módosítva, hogy a potenciométer középső lábához egy vezeték lett forrasztva és kivezetve a szervóból a mikrokontroller analóg bemenetére. Ez által nyomon követhető a szervó aktuális pozíciója.

Ezen kívül eredetileg három kábel lép ki a szervóból: a tápkábel (piros), föld (fekete) és a vezérlőjel kábele (sárga vagy fehér). Ez a vezérlő jel kábel szintén a PIC egy lábára van kötve, míg a tápkábel egy 5,7 voltos tápra, a föld kábel pedig a közös földre.

A vezérlő jel egy PWM jel (pulse-width modulation signal), ami azt jelenti, hogy ez egy négyyszög jel, aminek kitöltési tényezője variálható, és a kitöltési idő hossza szállítja az információt. Az általunk vizsgált szervó esetében ez egy 50 Hz frekvenciájú ($T=20000\ \mu s$) négyyszögjel, aminek kitöltési ideje 500 és 2500 μs között változtatható (4. ábra).

¹ http://www.pyroelectro.com/tutorials/servo_motor/parts/hitec_servo_b.jpg

4. ábra: A szervó vezérlő jele²

3.4 A robot vezérlőpanelje

A vezérlő panel a robotkar alapját képező fém dobozban foglal helyet. Ez a doboz eredetileg egy számítógép tápegységének volt a burkolata. A panel lényege a PIC16F877 mikrokontroller. A PIC órajelét egy 20 MHz-es kristály oszcillátor biztosítja. A robotba kívülről 12 VDC-s vezeték csatlakoztatható. A 12 voltból 7805-ös feszültségszabályozó IC-k állítják elő a szükséges üzemi feszültségeket. Két ilyen IC van a panelon, mert a szervók és a PIC külön tápról működnek. Ennek két oka van. A szervók működéséből adódó áramingadozás negatív hatással lehet a PIC stabil működésére, valamint a szervók magasabb feszültség mellett nagyobb nyomaték leadására képesek. A robotban használt szervók 4,8-6 V üzemi feszültségen üzemeltethetők. Ha egy 7805-ös IC középső lábát egy diódán keresztül kapcsoljuk a földre, akkor a dióda nyitófeszültsége (0,7V) hozzáadódik az IC kimeneti 5 voltos feszültségéhez. Így a szervók számára egy 5,7 voltos tápot tud biztosítani a nagyobb nyomatékért. A PIC 5 voltos tápját egy hagyományos 7805-ös kapcsolás biztosítja.

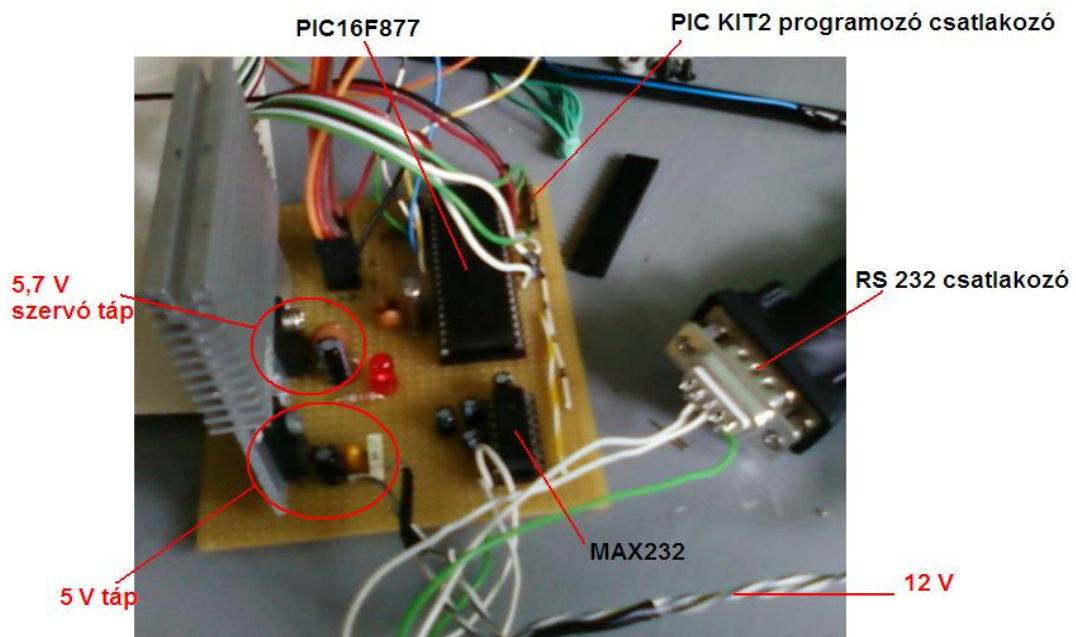
Az RS232-es kommunikációhoz a PIC RX és TX lábai vannak használva, azonban a PIC 0-5 voltos logikai szinteket használ. A 0 V feszültség a számítógép számára nem igazán értelmezhető, mert -12 és 12 voltos logikai szinteket használ a kommunikálásra. Ezt az eltérést hidalja át a MAX232-es IC, ami négy elektrolit kondenzátor segítségével végzi el a szintillesztést.

² <http://www.mitchr.me/SS/batteriesRequired/RCcontrolTheory/pics/RCsigPic.jpg>

A robot felkarjában helyet foglal egy vezeték elosztó panel is. Így elég csak egy táp- és földkábel elvezetni odáig valamint a vezérlő kábeleket és az érzékelő kábelét. Így megspórolunk tíz vezetéket. Ez az elosztó panel azt is nélkülözi, hogy a robot alkarjában lévő nem elég hosszú szervó vezérlőkábeleket meg kelljen hosszabbítani. A vezérlőpanelen a vezetékek közvetlenül a PIC lábai mellé vannak forrasztva, míg az elosztópanelra közös csatlakozóval vannak bekötve, így könnyen leválaszthatóak. A kábelek kiosztását a II/a. melléklet ismerteti.

Az első két szervó vezetékei még közvetlenül a vezérlőpanelra vannak kötve, mert vezetékeik elég hosszúak és elérnek a panelig, sőt, az első szervónak legnagyobb része magában a házban található.

A panel kapcsolási rajzát, mely Eagle nyáktervező programban készült a II/b melléklet szemlélteti. A 5. ábra pedig az elkészült vezérlőpanelt mutatja, ahol látható a feszültségszabályozókra szerelt hűtőborda. Amikor a szervók erőlködnek a pozíció megtartásán nagyobb áram folyik és ezek az integrált áramkörök nagyon felmelegednek. A dobozban benne maradt a számítógép tápegység eredeti hűtő ventilátora is, mely a meleg levegőt vezeti ki a dobozból.



5. ábra: Vezérlőpanel beépítés előtt

4. A vezérlő program működése

A vezérlést két program együttműködése valósítja meg: a mikrokontrollerben futó program és a számítógépen LabVIEW környezetben futó program. Utóbbi felelős a számítások elvégzéséért és az ember-gép kapcsolatért, míg a PIC legfontosabb feladatai a szervók vezérlőjelének előállítása, a szervók belső potenciométer állásainak lemérése és kommunikálás a számítógéppel. Ezek közül egyes funkciók nem választhatóak el teljes mértékben a számítógépen futó programtól, ezért azokat a részprogramokat (subVI-okat) is tárgyaljuk majd párhuzamosan, hogy jobban megértsük az összetettebb működését a rendszernek. A PIC programja C nyelven íródott. Nézzük meg először a legfontosabb funkciót: a vezérlőjelek előállítását.

4.1 Vezérlőjelek előállítása mikrokontrollerrel

A négyyszögjel előállítása úgy lehetséges, ha a PIC egy kimenetét ismétlődően magas majd alacsony szintre kapcsoljuk szabályos várakozással ezek között. Ha például középállásban szeretnénk vezérelni a szervót, akkor egy 20 ms periódusidejű és 1,5 ms kitöltési idejű négyyszögjelre lesz szükségünk. Ez legegyszerűbben a `delay_us()` funkció hívásával érhető el, amely a zárójelben megadott értékű μ s várakozást végez.

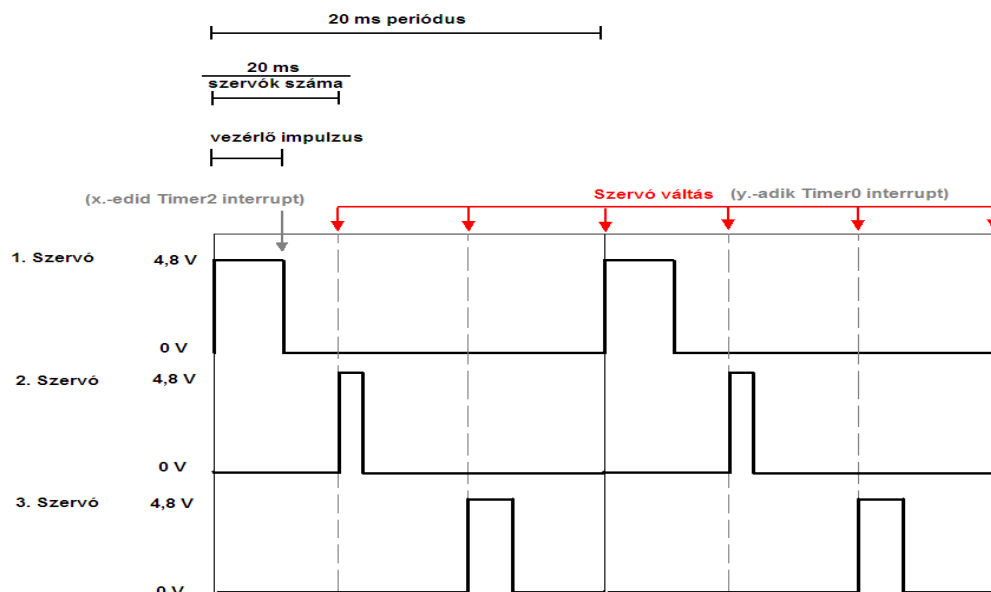
```
while(1)                                //örökké ismétlődő ciklus
{
    output_high(pin_b1);                  //magas jelszintre kapcsolja a PIC B portjának első
    lábát
    delay_us(1500);                       // vár 1,5 ms-ot
    output_low(pin_b1);                   //alacsony szintre állítja a kimenetet
    delay_us(18500);                      //vár, hogy befejeződjön a 20 ms-os periódus
}
```

Ha nem csak egy konstans vezérlőjelet akarunk előállítani, hanem mondjuk a szervó állását egy külső potenciométerrel szeretnénk vezérelni, akkor a while ciklus elejére egy AD konverziót beszúrhatunk, és a várakozások értékét pedig attól tesszük függővé. A potenciométer két szélső lábán lévő feszültségeket referencia feszültségnek használva egy 8 bites AD konverzió eredménye 0-255 közé eső szám lesz a potenciométer teljes tartományára értelmezve. A beállítható tartomány 500 μ s és 2500 μ s között van, tehát a nagysága 2000 μ s. $2000/255 = 7,8$. A 0-255 tartományt 8-al felszorozva 0-2040-et kapunk, ami 40 μ s extrával még belefér a pontatlan szervó vezérlőjelének tűrésébe. Ennyit várunk tehát az első delay függvényben, amíg magas jel van a kimeneten. Kikapcsolása után a második várakozáshoz 20000 μ s-ból kivonjuk a kiszámolt értéket, hogy teljes legyen a PWM periódus.

Mi a probléma ezzel a megoldással? A delay függvények teljesen lefoglalják a processzort és a mikrokontroller semmilyen más feladatot nem képes ez alatt ellátni. A PIC16F877 rendelkezik két PWM modullal. Ebben a modulban beállítható a jel frekvenciája és a kitöltés ideje is. A modul bekapcsolása után a processzor képes más feladatok ellátására, azonban ez a megoldás sem felel meg még a kívánalmaknak. Mivel a két PWM modul közös időzítőt használ (Timer2), különféle jeleket nem tudnak előállítani egy időben.

4.2 Vezérlőjel kivitelezése megszakításokkal több szervóra

A PIC 16F877 három időzítővel rendelkezik. Ezek közül a Timer0 és a Timer2 használatával megvalósítható a kívánt vezérlés, amely képes egyidejűleg nyolc szervó vezérlésére is. A Timer 2 időzítő itt is a vezérlő impulzus hosszáért felel, akárcsak a PWM modul működése során, azonban egy szervó vezérlőjének alacsony szinten töltött idejében a Timer2 a többi szervót vezérli. Fel kell osztani tehát a 20 ms-os vezérlőjelet annyi részre, ahány szervót vezérlünk. Arra azonban figyelni kell, hogy az egyes részek ne legyenek rövidebbek a leghosszabb kitöltési időnél. Esetünkben egy ilyen résznek legalább 2500 μ s hosszúnak kell lennie, ami a szervó jobb végállásba vezérléséhez szükséges. Ez azt jelenti, hogy pontosan nyolc szervó vezérlése válik így lehetővé egyidejűleg. A másik időzítő (Timer0) megszakításai váltanak egyik szervóról a másikra. Fontos megjegyezni, hogy egy globális változó tárolja, hogy melyik szervót vezéreljük éppen. Az 6. ábra három szervó vezérlését mutatja be ezzel a módszerrel. Ebben az esetben három részre osztjuk a vezérlő jel periódusidejét.



6. ábra: Három szervó vezérlése megszakításokkal

Ha átadódik a vezérlés egy új szervóra, a PIC megfelelő lába azonnal magas szintre kapcsol és úgy is marad, amíg a megfelelő számú Timer2 megszakítás be nem következik. Ez után, ha a Timer0 is elég megszakítást végzett, a vezérlő jel a következő szervón fog bekapcsolni. A Timer2 ismétléseinek a száma pedig attól függ, hogy melyik szervót vezéreljük éppen, és hogy azt milyen állásba szeretnénk állítani.

Konkrét esetünkben mi hét szervót vezérelünk. Egyet-egyét a robot csuklóiban, valamint a megfogót is egy szervó működteti. Ehhez 2857 μ s-onként kell szervót váltani.

A Timer0 egy 8 bites számláló, ami időbeli felbontásától függően növeli egyesével az értékét. Amikor túlcsordul, akkor okoz megszakítást.[3] A mi esetünkben a felbontás 0,2 μ s, a PIC számára pedig egy 20 MHz-es kristály biztosítja az órajelet, így a Timer0 beállításait alapul véve 51,2 μ s-onként fog megszakítást okozni. Ez azonban sokkal kevesebb, mint a kívánt 2857 μ s. Minden megszakításban ezért növeljük egy változó értékét eggyel, amíg 55 nem lesz. Ekkor értünk el ugyanis egy megközelítő értéket (2816) amivel még változatlanul működik a szervó. Ha ez bekövetkezett, törlődik a számláló értéke valamint növeli eggyel a „melyik_servo” változó értékét. Ha már a hetedik szervót vezéreltük, akkor nyolc helyett egyre változik az értéke. Amelyik szervót elkezdi vezérelni a rendszer, az ahhoz tartozó kimenetet magas jelszintre kapcsolja, valamint engedélyezi a Timer2 megszakítást.

A Timer2 számított és a gyakorlatban valóban működő beállításai tisztázatlan okokból nem egyeznek meg. Először vizsgáljuk meg a számolt beállítás logikáját. 4 μ s-os megszakítási periódust érünk el úgy, hogy a 20 MHz-es órajel mellett négyszeres előosztást alkalmazva 0,8 μ s-onként nő a Timer2 nyolcbites regiszterének az értéke. Négy túlcsordulási periódus után szeretnénk megszakítást, valamint egyszeres utóosztást alkalmazva sikerül is a megszakítást minden 4 μ s-ra időzíteni.

A Timer2 megszakításban szintén egy számláló értékét növeljük és hasonlítjuk össze azzal az értékkel, ami az aktuálisan vezérelt szervó vezérlőjének előállításához szükséges. Ez mint tudjuk időben legalább 500 μ s. Mivel 4 μ s-onként történik megszakítás, az említett számláló értékét 125-ig kell növelni ahhoz, hogy elérjük az 500 μ s-ot.

A teljes vezérelt tartomány 500 – 2500 μ s, így a változtatni kívánt rész 0-2000. A LabVIEW program a -90°-tól 90°-ig terjedő szervó állást egy 0-250-ig terjedő skálára arányosítja át és ezt az értéket küldi a PIC számára. $2000/250=8$. Ez azt jelenti, hogy a minimum 125-höz hozzá kell még adni a LabVIEW-ből kapott érték nyolcszorosát és ezt összehasonlítani a Timer2 megszakításokat számláló változóval. Ha a számláló eléri a kívánt értéket, akkor a vezérlőjel és a Timer2 interrupt engedélyezése kikapcsolt, a számláló lenullázódik, és a vezérlés átugrik a következő szervóra. Ez a programban az

első szervóra az alábbi módon nézne ki, mely után a case structure folytatódik a többi szervóval:

```
#int_TIMER2                //Timer2 megszakítás
void TIMER2_isr(void)
{
    count_t2++;
    switch(melyik_servo)
    {

        case 1
            if (count_t2>=125+(servo1*8))
                {output_low(servo1_pin);          //vezérlőjel alacsony szintre billentése
                 count_t2=0;                        //számláló nullázása a következő szervó számára
                 disable_interrupts(INT_TIMER2); //kikapcsolja a Timer2 megszakítást
                 melyik_servo++;                    //következő szervó vezérlésére vált
                break;
            ...
    }
```

A valóság az, hogy a helyes működést sohasem sikerült elérni ilyenfajta számolással, hanem próbálkozások eredményeként az alábbi formula hozott megfelelő eredményt a feltétel vizsgálatánál (if utáni rész):

$$\text{count_t2} \geq 20 + (\text{servo1}/2) \quad (1)$$

Ehhez azonban a Timer2 beállításai is mások, mert itt nem használ előosztást és rögtön az első túlsordulásnál már megszakítást okoz, tehát elvileg 0,2 μ s-onként történne interrupt. Így az előbbi logikát követve nem 500 μ s-ot érne el a minimum érték, hanem csupán négyet. Ennek ellenére így működik helyesen a vezérlés, tehát feltételezhetően az előbb ismertetett logika hibás valahol, vagy a beállításokkal van gond.

4.3 Szervó pozíciók beolvasása

Ha a megszakítások letiltásával a szervó vezérlőjelét kikapcsoljuk, a motorok elvesztik a nyomatékukat és kézzel mozgathatóvá válik a kar, betaníthatunk mozgáspályákat a robotnak. A szervók állásának ismerete lehetővé teszi mozgások rögzítését és lejátszását. Ehhez a szervó saját szabályzására használt belső potenciométerének állását lehet beolvasni. A PIC16F877 nyolc analóg bemenettel rendelkezik. Ezek az A/D átalakítók a bemenő analóg jelet egy 10 bites (vagy opcionálisan 8 bites) számmal reprezentálják. A nyolc analóg bemenetként használható láb közül kettő referenciafeszültség bemenetnek is beállítható a szoftverben. Beállítható referenciának azonban a VDD és VSS feszültségek is, ilyenkor felhasználható az előbb említett két láb másra is.

Első kíséretek alatt a szervó potenciométerének két szélső lába szolgáltatta a referenciafeszültséget és az A/D átalakítás csak 8 bites volt. Az egyik szélső lábon 0,48V, míg a másikon 2,5V volt mérhető multiméterrel. Ezek megegyeznek a -90° és 90° -os állásban mért értékkel a középső lábon. A középső lábon mért feszültségek az A/D átalakítás után 0-255 számnak feleltek meg. A LabVIEW-ban arányosítással ebből ismét -90° és 90° lett.

A referenciafeszültséget csak egy szervóról lehetett levenni, azonban a többi szervó értékei nem teljesen egyeztek meg ennek az első szervónak a feszültségeivel. Olyan is előfordult, hogy a másik szervón mért feszültség nagyobb volt, mint az elsőről levett referenciafeszültség.

A probléma megoldásaként referenciának a VDD és VSS feszültségek lettek beállítva, melyek 0V és 5V. Így ezek a lábak is használhatóak A/D bemenetnek, mert ennél a beállításnál nem kell őket külön a VDD és VSS-re kötni. Hogy ne veszítsünk a pontosságból így viszont 10 bites A/D átalakítást kell végezni, ami a referenciafeszültségek közötti tartományt 0-1023 számmal jelenti meg. A gyakorlatban az első szervó állása így 99-512 számtartományban változik és a középpálláshoz tartozó érték 316. Az arányosítás ebből szervónként úgy történik, hogy először a középpálláshoz viszonyítva eldöntjük, hogy pozitív vagy negatív szögben áll, majd egyenes arányosságot alkalmazunk. A szélső és közép állások értékeit tehát előzőleg le kellett mérni minden szervóra és ezeket konstansként használni a LabVIEW-ban.

4.4 Kommunikálás LabVIEW-val

A mikrokontroller ismétlődő programjának (megszakításokat kivéve) legfontosabb feladata kommunikálás a vezérlő számítógép programjával soros porton keresztül. Az RS232-es kommunikációs csatorna 9600-as baudrátát (átvitt jelek/másodperc) használ paritásbit nélkül és egy 8 bit hosszúságú adatot visz át. Az átvitt tartalom string, vagyis karakterek tömbje (c program szerint). A számítógépet és a mikrokontrollert időzítőkkal nem lehetne jól szinkronba hozni és könnyen problémát is okozhatna, ha mégis megpróbálnánk, ezért a kommunikációt a lassabb PIC irányítja.

Az PIC program inicializálások utáni ismétlődő részében történnek az A/D konverziók, melyeknek az eredményét rögtön el is küldi a program a számítógépnek. Ez azonban csak akkor történik, amikor a motorok ki vannak kapcsolva, különben a szervó értékei azok, amiket a vezérlőprogram határoz meg. A küldött string első négy karaktere meghatározza, hogy milyen információt küldünk (pl.: pot1 = az első potenciométer). Az ötödik karaktertől kezdve a beolvasott értéket küldi szintén decimális string formájában, itt azonban az információt a küldött ötödik karakter ASCII kódtábla szerinti sorszáma jelenti, jobban mondva a char típusú változó értéke.

A LabVIEW program alaptól vár az adat fogadására, majd akármilyen üzenetet fogad annak első négy karakteréből határozza meg a következő lépést: fogadja a további adatot, mint a potenciométerek állásánál, vagy választ kell küldenie, mint az általa meghatározott szervó állás értékét, amit a PIC a Timer2-es megszakításában használ.

Amikor a PIC a szervók kívánt állását kérdezi le a számítógépről, összesen négy karaktert küld (pl.: `srv1` = első szervó), majd várakozik egy karakter fogadására. Ilyenkor a LabVIEW válaszul a kért szervó állását küldi egyetlen karakter formájában. Előzőleg a -90° - 90° közötti érték át van skálázva 0 és 250 közé, majd az ennek a számnak megfelelő ASCII karaktert küldi a PIC-nek, amit az egy 1 bájt hosszúságú char-ként értelmez.

Hasonló kérdés-válasz módszerrel történik a motorok be- és kikapcsolását, a megfogó zárását és nyitását vezérlő és a robot be és kimeneteinek állapotát közlő kommunikáció is. Mindig a PIC mondja meg a LabVIEW számára, hogy mit küld most, vagy minek az értékét kéri.

A motorok úgy kapcsolhatóak ki, hogy egyszerűen megszüntetjük a vezérlőjelet azáltal, hogy letiltjuk az azt vezérlő megszakításokat. Ilyenkor a motor tápfeszültsége nem szűnik meg, így a feszültségek a potenciométeren megmaradnak és lehetővé válik a kézzel irányított mozgás rögzítése. Ilyenkor pályarögzítésnél egy két dimenziós tömbben tárolódnak a csuklókoordináták minden ötödik milliszekundumban. Lejátszásnál ugyanezzel a gyakorisággal ebből a tömbből olvassa ki a program a küldendő szervóállásokat. A gyakoriság változtatásával természetesen a rögzített mozgás lejátszása lassítható vagy gyorsítható is.

4.5 A megfogó vezérlése

A megfogó szervója nem éri el a végállásait a megfogó nyitott vagy zárt állásánál. A mechanikai szerkezetnek nincs szüksége és képtelen is akkora szögelfordulásra. A mikrokontroller minden ciklusban lekérdezi a vezérlő programtól, hogy a megfogó nyitva, vagy zárva legyen-e. Ha nyitva, akkor azonnal a nyitott állásnak megfelelő értékre pozícionál. Ha zárni kell a megfogót, akkor fokozatosan zárul be, amíg el nem éri a csukott álláshoz tartozó végértéket, vagy meg nem nyomódik a megfogást visszaigazoló kontaktérzékelő a megfogó egyik úján. Ha csak simán zárt végállapotba vezérelnének a megfogó szervóját, akkor a megfogott tárgytól nem tudna bezáródni és a szervó erőlködne, hogy oda pozícionáljon. Ilyenkor ugyan nagyobb erővel is tudná tartani a megfogott tárgyat, azonban a folyamatos használatból nagyon felmelegedne, illetve az emiatti áramingadozás remegést vált ki a többi szervóból. A megfogó szorító erejét a kontaktérzékelő küszöbje határozza meg.

4.6 Programozható Point To Point mozgások

A PTP mozgások során a TCP nem követ egy pályát és a mozgás ideje alatt nem őrzi az orientációját se. A lényege csak az, hogy a mozgás befejeztével egy adott pozícióban és orientációban legyen a szerszám. A program a PTP mozgásokat nem orientációra és pozícióra alapulva végzi, mert még nem használ inverz kinematikát, hanem a csuklókat egyenként mozgatva irányítja a felhasználó a robotot egy adott pozícióba és a programnak egy pontként a szervók pozícióját tanítja meg. Az ilyen módon gombnyomásra megtanított pontok egy kétdimenziós tömbben tárolódnak, ahol a sorok az egyes pontok, míg az oszlopok a csuklók koordinátái.

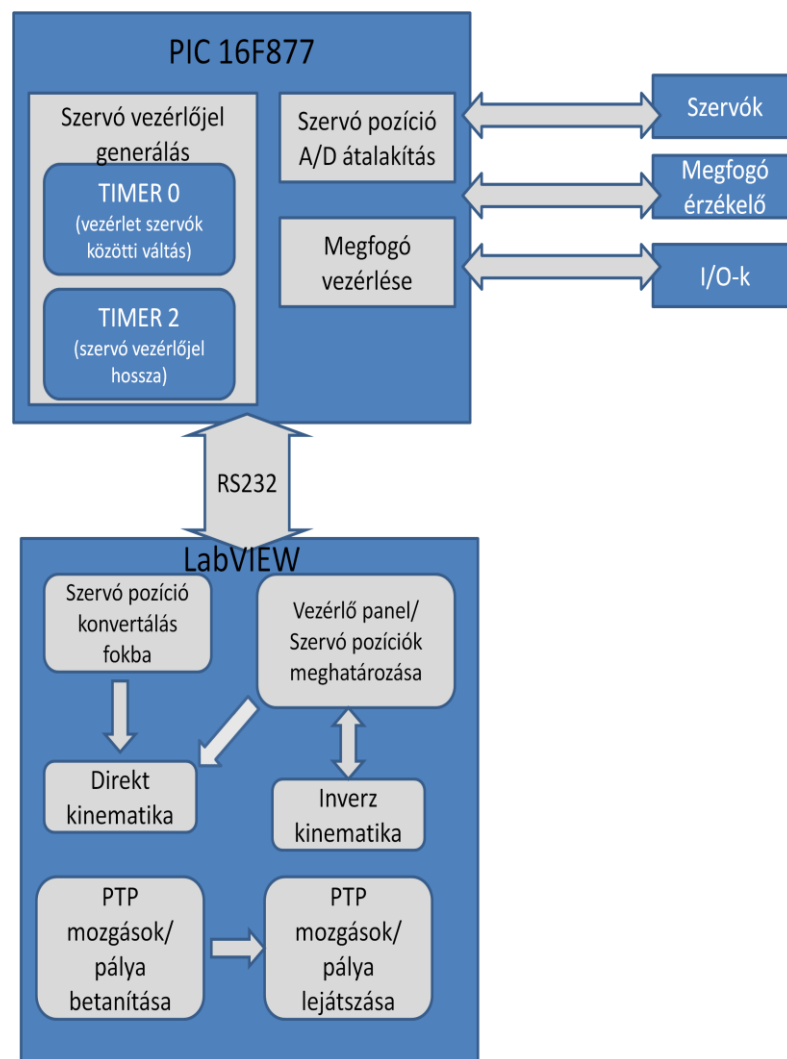
A szervó pozíciókon kívül más adatok is tárolódnak egy ponthoz tartozó sorban, ilyen a mozgás időtartama, a megfogó állapota és az outputok állapotai. A mozgás időtartama az a másodpercben értendő idő, ami alatt a robot az előző pontról az adott pontra átpozicionál. Ezt a pont megtanítása előtt kell beállítani minden mozdulatra. Ez nagyobb mozdulatoknál legalább kettő másodperc legyen, mert a hirtelen mozdulatok a robot tehetetlenségéből adódóan és a kevésbé masszív vázszerkezetnek köszönhetően kedvezőtlenek. A megfogó programozott nyitása és zárása is itt történik. Ilyenkor az adott pont csuklókoordinátái nem különböznek az előzőtől, csak a megfogó értéke változik (1=zárva vagy 0=nyitva). A kimenetek programozott vezérlése is ugyanezzel a bináris logikával történik az egyes sorokban. Az hogy a programozott PTP mozgássorozat egy bemenet változására kapcsoljon egy gombbal beállítható. Így válhat valóra a robot más rendszerekkel történő összekapcsolása.

Mivel a szervók csak egy adott sebességgel képesek egyik pozíciójukból a másikba átállni, ezért ahhoz hogy a PTP mozgások lejátszása egy folytonos, a beállított időtartamtól függő sebességű mozgás legyen, a mozgást sok kicsi pozícióváltásra kell felosztani. A mozgás megkezdése előtt a megadott időtartamból kiszámolja a program, hogy hány lépésre van szükség ahhoz, hogy 5ms-onként legyen léptetés. Ez után szervónként a kívánt pozíció és az aktuális pozíció különbségét elosztva a lépésszámmal delta értékeket határoz meg, amikkel minden 5 ms ciklusban változtatja majd az egyes szervók értékét. Ez maga a mozgás már.

Ha véget ért a mozgás kinullázódik a lépés számláló, és növekszik eggyel a pontszámláló értéke, aminek következtében a tömb következő sorában tárolt pontra ugrik ez a szubrutin. Ha a tömb végéhez ér, akkor kikapcsolja a PTP mozgássorozatot és a pont számlálót is nullázza, hogy előről kezdhető legyen az egész. Az egész tömb is gombnyomással törölhető, ilyenkor teljesen új mozgássorozat tanítható a robotnak. Ha a motorok ki vannak kapcsolva, akkor kézzel is beállíthatjuk a kívánt pozíciókat tanításhoz, mert ilyenkor a potenciométerekről leolvasott értékekkel dolgozik a ponttanító subVI.

Az alábbi linken vagy a CD-n elérhető videón a robot ilyen programozott mozgásokkal cukroz meg és kavar meg egy csésze teát:

<http://www.youtube.com/watch?v=ifXa5JAKu0A>



7. ábra: A vezérlés blokkvázlata

5. A robot kinematikája

A robotok kinematikájával nagy irodalom foglalkozik, ebből pár hasznos utalás megtalálható az irodalomjegyzékben. Most mi csak az elvégzett feladat megértéséhez szükséges alapokat fogjuk átvenni. A tisztelt olvasó számára Dr. Mester Gyula Robotkinematikával foglalkozó prezentációs anyaga kínálhatja a szükséges ismereteket tömör és érthető formában. Ez elérhető az irodalomjegyzékben szereplő hivatkozáson.

A robot csuklója már mozgathatóak külön- külön és a megfogó is működik, de nem tudjuk számszerűen még, hogy pillanatnyilag mi a megfogó pozíciója és orientációja, továbbá összetett mozgások kivitelezése sem lehetséges ez idáig.

Az ipari robotok esetében különböző koordinátarendszereket különböztetnek meg. Ezek a világ-, robot-, szerszám- és bázis koordinátarendszerek. Sokszor a világ- és robot koordinátarendszer egy és ugyanaz és a robot alapjának középpontjában helyezkedik el az origója, de a világ- koordináta rendszer lehet a roboton kívül is, mondjuk a szoba sarkában. A szerszám koordinátarendszer középpontja a TCP-ben (Tool Center Point / szerszám középpont, jelen esetben ez a megfogó) helyezkedik el, míg a bázis koordinátarendszer tetszőlegesen felvehető, például egy négyszögletű munkadarab sarokpontjában, ahol a koordinátarendszer tengelyei illeszkednek a munkadarab éleire.

Inverz kinematika segítségével a robot képes e koordinátarendszerek tengelyeinek mentén mozgatni a szerszámot, illetve a tengelyek körül forogtatni. Ilyenkor a szerszám pozíciója és orientációja adott és ebből kell a vezérlő programnak meghatározni a csuklók állását.

Ha a csuklók külön mozgásával irányítottuk valahova a robotot, akkor viszont a csuklók állásának ismeretében kell kiszámolni a szerszám pozícióját és orientációját. Ezt hívják direkt kinematikai feladatnak.

5.1 A direkt kinematikai feladat megoldása

A direkt kinematikai feladat megoldásához általánosan homogén transzformációs módszert alkalmaznak. Minden csuklóba egy koordinátarendszert rendelnek, és sorban leírják az egymást követő koordinátarendszerek közötti transzformációs mátrixokat, (vagyis hogyan áll az i -edik ízület az $i-1$ -edikhez képest) ez által az egész kar pozíciója megadható. A homogén transzformációs mátrixok 4×4 -es mátrixok, melyek tartalmazzák a két koordinátarendszer közötti rotációt és a két koordinátarendszer origójának távolságát. Segítségükkel a kar végén lévő TCP pozíciója és orientációja határozható meg az alap koordinátarendszerben. Fontos szempont, hogy a koordinátarendszereket mindig ugyanazon elv szerint határozzuk meg, s az elmozdulások a legegyszerűbb formában szerepeljenek a transzformációs mátrixokban. [6] Ezeknek a feltételeknek a megvalósítására szolgál a Denavit és Hartenberg által kifejlesztett, róluk elvezett leírási módszer. A módszer lényege, hogy egyik koordinátarendszer a másikba két forgó és két haladó mozgással átvihető.

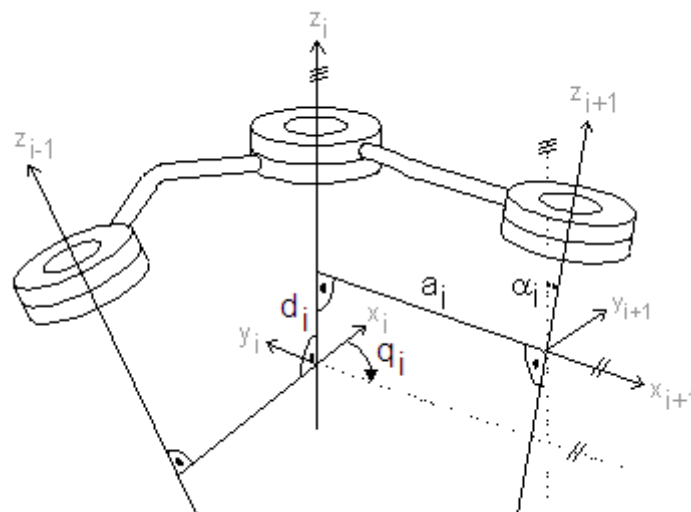
A koordináta-rendszerek felvételének szempontjai:

- z_i tengely legyen az ízület forgástengelye
- x_i tengely mutasson a z_{i-1} és az z_i tengely közös normálisába
- y_i tengely egészítse ki a koordináta-rendszert jobbsodrásúvá
- Párhuzamos forgástengelyek esetén azt a közös normálist választjuk, amelyik a megelőző izülethez rendelt koordináta-rendszer origóján halad át.[6]

Az $(i+1)$ -ik koordináta-rendszer viszonyát az i -ikhez képest négy értékkel definiáljuk:

- q_i jobbsavar irányú elfordulás a z_i tengely körül
- d_i a csuklótengely két normálisa közti távolság
- a_i az i -edik és $i+1$ -edik csukló tengelyek közös normálisának hossza
- α_i az i -edik és $i+1$ -edik csuklótengelyek közötti jobbsavar irányú szög.[6]

Ezek az ún. D-H paraméterek, melyek megértését a 8. ábra teszi szemléletesebbé.



8. ábra: D-H paraméterek³

Ezt a négy transzformációt külön transzformációs mátrixokban írjuk fel, majd ezeket összeszorozzuk mátrixszorzással. Ezt a feladatot a LabVIEW programban egy külön függvény végzi melynek bemenetei a felsorolt négy paraméter, kimenete pedig egy 4×4 -es mátrix. A robotkar esetében hat ilyen D-H transzformációra van szükség a direkt kinematikai feladat megoldásához. A hat referencia koordináta-rendszert a III/a melléklet szemlélteti, míg a szükséges D-H paramétereket a III/b melléklet.

E mátrixok szorzását itt nem részletezzük, de szolgáljon referenciaként a [6]. forrás.

³ http://e-oktat.pmmf.hu/webgui/www/uploads/tenkesnet/bpandur/7_elemei/image012.gif

A hat eredmény mátrix szintén összeszorozódik, és a végeredmény egy 4x4-es mátrix lesz, mely tartalmazza a végső koordinátarendszer helyzetét és orientációját az alap koordinátarendszerben. Ebből a mátrixból a végpont helyzete könnyen kiolvasható. A mátrix negyedik oszlopa a pont helyzetvektora, melynek első három értéke az x, y és z tengely menti távolságok az alap koordinátarendszer origójától. Az orientáció nem olvasható ki ennyire egyszerűen. A következő formulákkal határozható meg az eredmény mátrix bal felső sarkába illeszkedő 3x3-mas rotációs mátrixból ([5] Wöhlhaf: Introduction Robotics, 95. old.):

$$\underline{R} = \begin{pmatrix} x_x & y_x & z_x \\ x_y & y_y & z_y \\ x_z & y_z & z_z \end{pmatrix} \quad (2)$$

$$\alpha = \arctan \frac{x_y}{x_x} + k\pi \quad (3)$$

$$\beta = \arctan \frac{-x_z}{x_x \cos \alpha + y_x \sin \alpha} \quad (4)$$

$$\gamma = \arctan \frac{y_z}{z_z} + 2k\pi \quad (5)$$

Speciális eset, ha $\beta = k \frac{\pi}{2} \rightarrow x_x = x_y = y_y = z_y = 0$

$$\gamma = \begin{cases} \arctan \frac{-y_x}{y_y} - \alpha + 2k\pi & \text{ha } \beta = -k \frac{\pi}{2} \\ \arctan \frac{y_x}{y_y} + \alpha + 2k\pi & \text{ha } \beta = k \frac{\pi}{2} \end{cases} \quad (6)$$

Ezeket a számításokat is a számítógépes program végzi el minden ciklusban, majd az eredményeket egy 1x6-os világkoordináta vektorban határozza meg, melynek elemei x, y és z távolságok, valamint Roll(α), Pitch(β) és Yaw(γ) Euler szögek (forgás, billenés, csavarás) az alap koordinátarendszerhez viszonyítva.

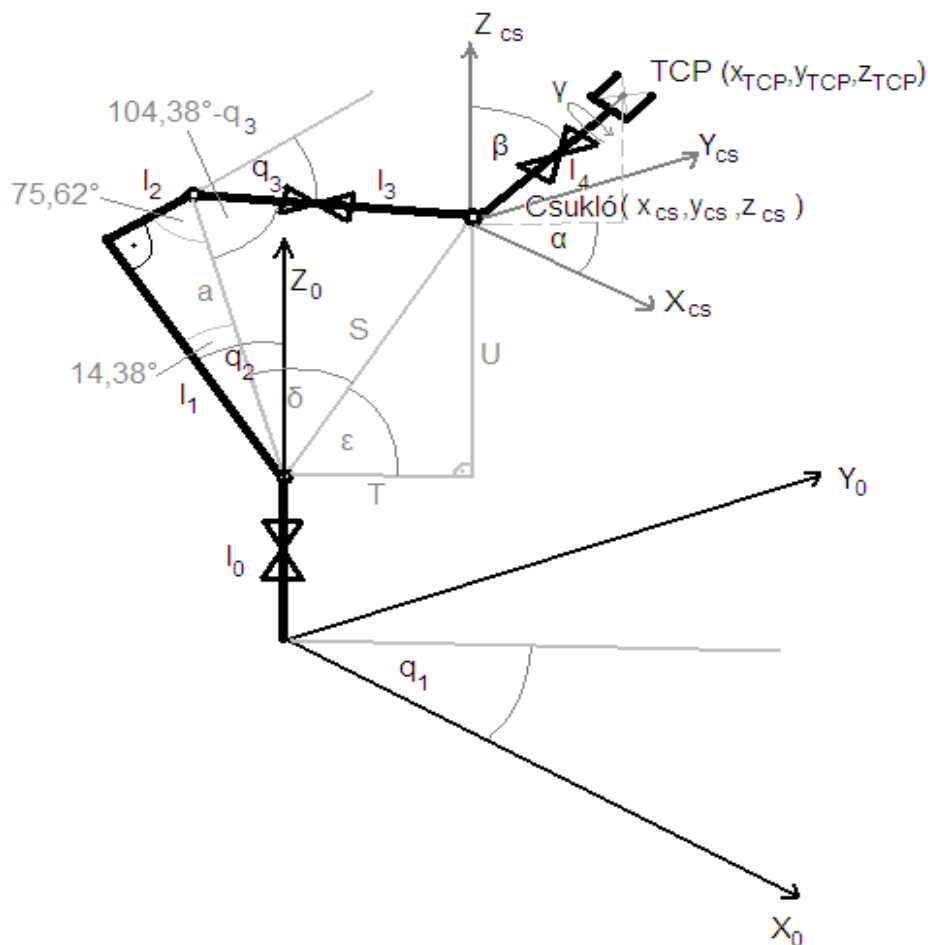
5.2 A robot inverz kinematikája

A vezérlőprogram ezt akkor használja, amikor TCP helyzetét és orientációját világkoordinátákkal adjuk meg és ebből kell meghatároznia a programnak a csuklók állását. Többféle módszer létezik az inverz kinematikai feladat megoldására, azonban nincsen egy abszolút megoldás sem. Minden egyes robot konfigurációra külön kell megoldani.

Az inverz kinematikai feladat két módon oldható meg: analitikus módszer és numerikus módszer. A numerikus megoldás deriválja a robotkar világ és csuklókoordinátái közötti összefüggést. A világkoordináták sebességvektora és a csuklókoordináták sebességvektora közötti összefüggést meghatározó Jacobi-mátrix és annak invertáltja segítségével számol.[4] Tekintve, hogy a robotban alkalmazott szervók csak egy féle sebességgel képesek mozogni, nem ezzel a módszerrel lett megoldva a feladat, hanem analitikusan.

Az analitikus módszer nagyszámú trigonometrikus egyenlet megoldásával jár, de pontosabb, mint a numerikus megoldás és meg is adja az összes lehetséges megoldást, melyek közül a munkatér adottságaira való tekintettel választhatunk.

Ennél a robotnál is az utolsó három ízület forgástengelye egy pontban, a csuklóban metszi egymást. A csukló koordinátáját a TCP helyzete egyértelműen meghatározza, a csukló pozíciója pedig q_1 , q_2 és q_3 ízületi változóktól függ.[6]



9. ábra: Az inverz kinematika analitikus megoldása

A 9. ábra felső részén látható a TCP és csukló viszonya az orientációs szögek függvényében. (A csuklóban felvesszünk egy az alap koordináta-rendszerrel azonos orientációjú koordináta-rendszert.)

A csukló koordinátái:

$$x_{cs} = x_{TCP} - l_4 \cos \alpha \sin \beta \quad (7)$$

$$y_{cs} = y_{TCP} - l_4 \sin \alpha \sin \beta \quad (8)$$

$$z_{cs} = z_{TCP} - l_4 \cos \beta \quad (9)$$

q_1 kiszámítására a megszokott arkusz tangens függvény helyett az atan_2 függvényt használjuk (LabVIEW-ban elérhető), mivel ez az előjelek alapján megállapítja azt is, hogy melyik tér negyedben van a szög:

$$\mathbf{q}_1 = \text{atan}_2(y_{cs}, x_{cs}) \quad (10)$$

q_2 és q_3 kiszámítása:

$$S^2 = a^2 + l_3^2 - 2a l_3 \cos(104,38^\circ - q_3) \quad (11)$$

ahol
$$S^2 = x_{cs}^2 + y_{cs}^2 + (z_{cs} - l_0)^2 \quad (12)$$

ebből
$$\mathbf{q}_3 = 104,38^\circ - \cos^{-1} \left(\frac{a^2 + l_3^2 - S^2}{2al_3} \right) \quad (13)$$

q_2 kiszámítása:

$$\varepsilon = \text{atan}_2(U, T) \quad (14)$$

ahol
$$U = z_{cs} - l_0 \quad (15)$$

és
$$T = \sqrt{x_{cs}^2 + y_{cs}^2} \quad (16)$$

Cossinus tételből:
$$\delta = \cos^{-1} \left(\frac{S^2 + a^2 - l_3^2}{2Sa} \right) \quad (17)$$

$$\mathbf{q}_2 = -((\delta + \varepsilon) - 90^\circ + 14,38^\circ) \quad (18)$$

Mivel az utolsó három ízület forgástengelye egy pontban metszi egymást ezért a

$$\underline{H}_4^{TCP} = \underline{H}_4^5 \cdot \underline{H}_5^6 \cdot \underline{H}_6^{TCP} \quad (19)$$

mátrix formailag azonos lesz azokkal a transzformációs mátrixokkal, amelyeket a Roll Pitch Yaw szögekkel történő forgatássorozatok eredményeként kaptunk.

q_1 , q_2 és q_3 ismeretében a

$$\underline{H}_1^4 = \underline{H}_1^2 \cdot \underline{H}_2^3 \cdot \underline{H}_3^4 \quad (20)$$

kiszámolható a transzformációs mátrixok szorzásával.

$\underline{H}_1^{TCP}$ ismert a világkoordinátában megadott pozíció és orientációs adatokból, amivel vezérlünk. $\underline{H}_1^4$ invertálásával megkaphatjuk a

$$\underline{H}_4^{TCP} = \underline{H}_1^{TCP} \cdot (\underline{H}_1^4)^{-1} \quad (21)$$

mátrixot, mely a csukló és a TCP közötti transzformációkat tartalmazza. Ebből a mátrixból a már 5.1-ben ismertetett módon, a (2)-(6) képletek alkalmazásával kiszámolható a szükséges három forgatás.

Az inverz kinematikát szintén egy külön alprogram valósítja meg LabVIEW-ban. Amikor a csuklók egyenkénti mozgása helyett a világkoordinátában való mozgást választjuk, akkor a szervó értékeket ez a program adja meg. Amikor világkoordináta vezérlésről csuklóvezérlésre váltunk, akkor a robotkar egy helyben marad, tehát a szervó értékek átadódnak az inverz kinematikai programból. Amikor a szervó vezérlésről váltunk inverz kinematikára, akkor is a direkt kinematika eredményét képező világkoordináta vektorból kellene feltölteni az inverz kinematikai program világkoordináta kezdő értékeit, hogy a robot ugyanabból a pozícióból folytathassa a mozgást. Sajnos azonban erre még nincs lehetőség, mert ez a fent ismertetett inverz kinematikai megoldás megvalósítása még nem teljesen tökéletes. Az x-z síkban, ha orientációnál csak a Pitch szöget módosítjuk, akkor tökéletesen működik, erről videó is tanúskodik (CD melléklet vagy <http://www.youtube.com/watch?v=FavPEqxzzv4>), de a Roll és a Yaw szögekkel valami probléma még van. Ezért ha y irányba is mozgatjuk a robotot, illetve e szögekkel változtatjuk az orientációt, nem jól orientálódik a TCP. Ezért a direkt kinematikából átvett értékek az inverz kinematikában mégsem ugyanazt az orientációt fogják elérni, de a pozíció megfelelő lesz. Gyakorlatban ennek az az eredménye, hogy amikor csukló vezérlésről a világkoordinátában való vezérlésre váltunk, a robot hirtelen mozgással változtatja a helyzetét.

6. A robot, mint egy automatizálási rendszer része

Mint azt már tudjuk, a robot rendelkezik logikai ki- és bemenetekkel. Ezek segítségével a robot ténylegesen beilleszthető egy ipari, vagy inkább kísérleti automatizálási rendszerbe. Képes könnyű elemek manipulálására, de ha valódi PLC-hez kapcsoljuk szükség lesz egy kapcsolóáramkörre, ami megoldja a feszültségillesztést, mert a I/O-k a PIC 5 voltos logikai jelszintjével üzemelnek. A kimenetek még kisebb, 3 voltos feszültséggel működnek, mert ellenállások használatával kerüli el az esetleges rövidzárat.

A robot használati lehetőségeinek szemléltetésére készült egy szortírozó feladatot ellátó érzékelővel és futószalaggal felszerelt pad. A berendezés egy FIFO állványról érkező műanyag kockákat válogat szét szín szerint. A robot a FIFO adagolóból a szenzor elé helyezi a kockát, majd a mérés után a futószalagra. A futószalag két irányba mozdulhat el a kocka színétől függően. Az egyik irányba még egy terelő szabályozza a kocka útját, ami eszerint a futószalag végén bal, vagy jobb oldalt esik le. Így kerülnek a különféle kockák három különböző helyre.

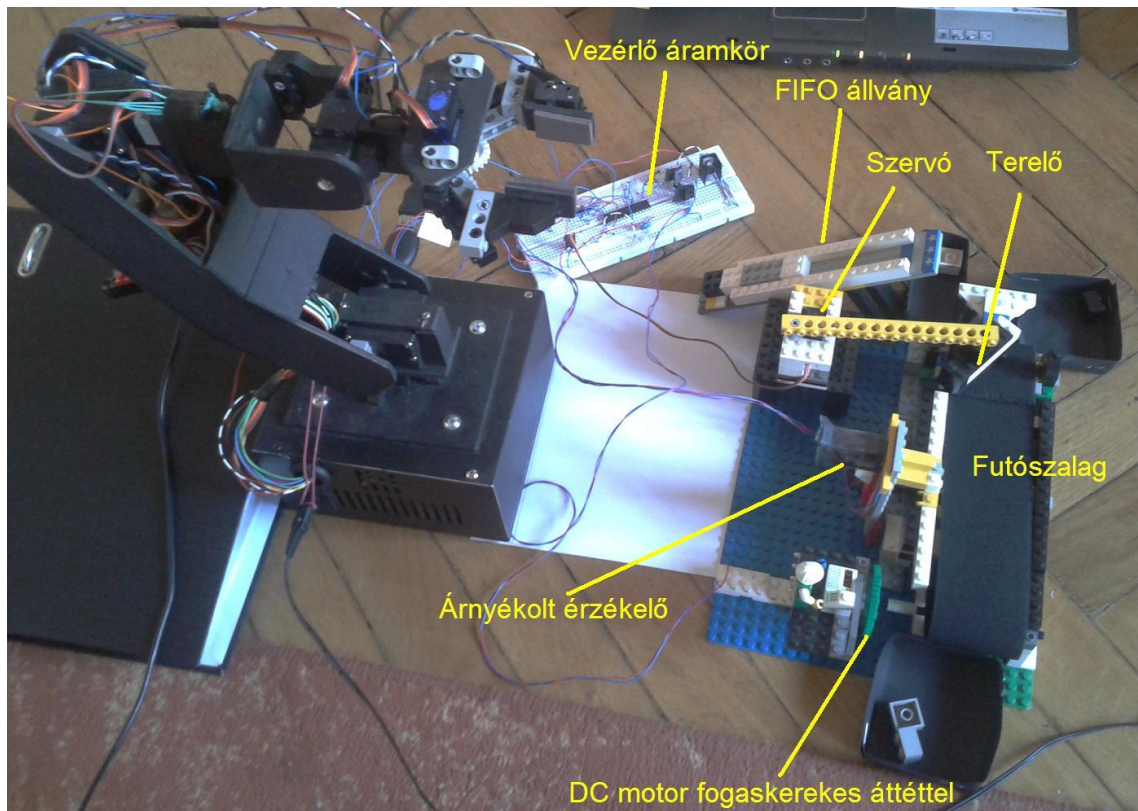
A kockák fehér, szürke és fekete színűek, és a szenzor ezekről a kockákról visszavert fény mennyiségét méri. Nyilvánvalóan minél sötétebb a kocka annál kevesebb fény verődik vissza. Maga a szenzor egy nagy fényerejű LED-ből és egy fotoellenállásból áll. A fotoellenállás egy 1kOhm-os ellenállással feszültségosztót képez, ami a vezérlést ellátó mikrokontroller analóg bementére van kötve. A fotoellenállás ellenállása nagyon világosban legkevesebb 70 Ohm, míg teljesen sötétben megközelítőleg 70 kOhm. Kezdetben sok problémát okoztak a változó környezeti fényviszonyok, illetve ennek kompenzálása. A végső megoldás a környezeti fény megközelítőleg teljes kizárása lett. Egy fekete kartonból készült alagút veszi körül a szenzort, aminek a végén feketére festett szivacs segíti a minél tökéletesebb illeszkedést az alagút fala és a kocka között, ezáltal minél kisebb mértékű fényt engedve be kívülről. így a mérés már elég megbízható lett. Hiba csak akkor fordul elő, ha a robot remegéséből adódóan mégsem elég tökéletesen illeszti a kockát a szenzor elé, és bejut valamennyi kinti fény.

A futószalagot mozgató DC motor kétirányú mozgását H híd teszi lehetővé, valamint egy 7806-os feszültségszabályozó IC állítja elő a robottal közös 12 Voltos tápból a motor számára szükséges 6 Voltot. A futószalag végén lévő terelőt egy olyan szervó motor mozgatja, mint ami a robot megfogóját is.

Ennek a rendszernek is egy PIC 16F873a mikrokontroller valósítja meg a vezérlését, de célja az, hogy egy PLC működését utánozza. Ez a külső áramkör csak egy ideiglenes tesztpanelon lett összeépítve, nem lett áramkörbe forrasztva. A fejlesztése során ez is használt soros kommunikációt a számítógéppel, hogy a szenzor által mért értékek

láthatóak legyenek a monitoron. Ehhez nem készült LabVIEW program, csak a Windowsos HyperTerminal program jelenítette meg a PIC által küldött információkat.

Ez a mikrokontroller van hozzákapcsolva a robot ki- és bemeneteihez. A program kommunikál a robottal és működése szekvenciális. Ez fog indító jelet küldeni a robot első inputjára, hogy a robot elkezdheti a kocka megfogását és szenzor elé pozicionálását, majd vár addig, amíg a robot visszaigazolást nem küld arról, hogy már ezt megtette (Output1). Ekkor kapcsol be az érzékelő és végez a mikrokontroller méréseket, amiket utána átlagol. A robot rövid várakozás után továbbviszi a kockát a futószalagra, aminek mozgásirányát és a terelőjét is már beállította a PIC. A robot elengedi a kockát és visszatér kiinduló helyzetébe, majd jelet küld a PIC számára, hogy készen áll a művelet megismétlésére (Output2) és vár a PIC-től érkező újabb indító jelre. A vezérlés folyamatábrája a IV mellékletben megtalálható.



10. ábra: A szortírozó áttekintése

A szortírozó működése megtekinthető videón a dolgozat CD mellékletéről, vagy a http://www.youtube.com/watch?v=7_Zf-UQBQV4 linken.

7. A robot vezérlése 3D kapacitív interfésszel

A robot működésének és inverz kinematikájának látványos szemléltetése (a programozott mozgáson kívül), amikor a felhasználó nem csak virtuális gomb nyomogatásával manipulálhatja a robotot, hanem valamilyen fizikai interfész segítségével irányíthatja azt. Erre egy kitűnő lehetőséget kínál a Kyle McDonald által bemutatott, háromdimenziós kapacitív interfész.[7] Ez egy olyan alufóliával bevont kartonlapokból álló érzékelő, amely képes nyomon követni az ember kezének pozícióját. De hogy működik?

Mindegyik alufólia lap kapacitásként viselkedik, ami képes töltést tárolni, ha feszültséget kapcsolunk rá. Mindegyik fólia egy mikrokontroller egyik lábára van kötve, ami biztosítja a feszültséget és az áramot a fóliák feltöltéséhez. Ahogy a kezünkkel közelítünk a szenzorlap felé a testünk elektronikus kölcsönhatásba lép vele és változtatja a kapacitását. Minél nagyobb a kapacitása, annál tovább fog tartani a lemeznek, hogy teljesen feltöltődjön.

A mikrokontroller méri, hogy mennyi idő kell a lemez feltöltéséhez. A feltöltési idő változása megfelel a kéz távolságának a fóliától. Ez a viszony azonban erősen exponenciális, ezért a mért értékeket LabVIEW-ban linearizálni kell.

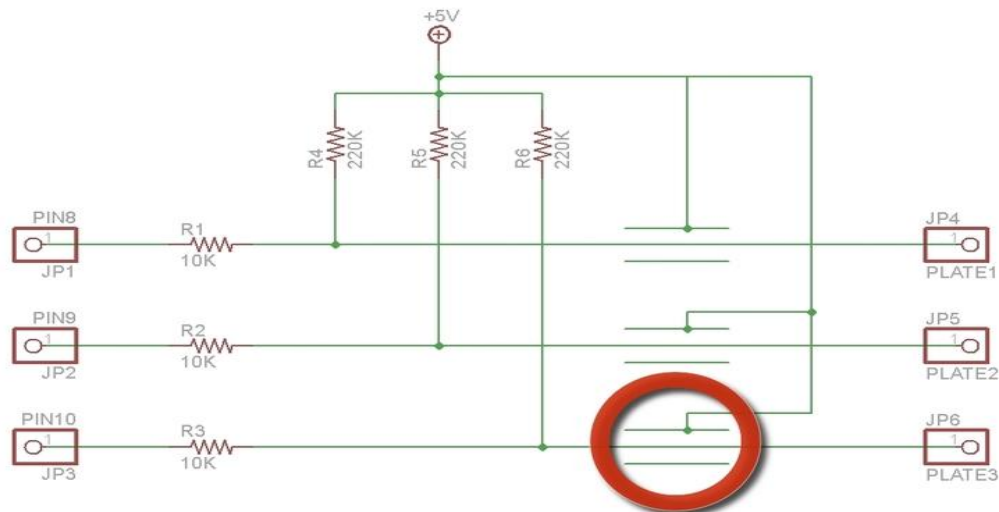
A lemezeket a mikrokontrollerrel összekapcsoló vezetékek árnyékolnak, hogy a vezeték ne viselkedjen antennaként és tegye lehetetlenné a mérést. A megfelelő működéshez szintén fontos, hogy a mikrokontroller jól le legyen földelve.

Mindegyik alumínium fólia egy 10 k Ω -os ellenálláson keresztül van a PIC lábára kötve, valamint egy 220 k Ω felhúzó ellenállással az 5 V-os tápra (11. ábra).

A méréshez először ki kell sütni a kondenzátorokat. Ehhez a PIC aktuális lába logikai alacsony szintre kapcsol, vagyis földre. Ekkor a 10 k Ω -os ellenálláson keresztül kisül a kondenzátor, valamint a 220 k Ω -os ellenálláson keresztül folyó áram is erre folyik tovább. Néhány μ s várakozás után, mikor a lemez biztosan kisült, a PIC lába bináris bemenetként kezd el működni és a program méri azt az időt, mire újra feszültség jelenik meg a lábon. Amíg ugyanis töltődik a kapacitás, addig arra felé folyik az áram, de amint megtelt és már teljesen szigetel akkor mégis inkább a 10k ellenálláson folyik keresztül a PIC lába felé (11. ábra).

A mérési eredmény több mérés átlaga, melyek az 50 Hz-es elektromos hálózat egy periódusának ideje alatt történnek. Arra hogy a teljes 20 ms periódus alatt végig végezzünk méréseket azért van szükség, mert ha véletlenszerűen történik a mérés, lehet hogy az AC hálózat pillanatnyi értéke éppen csúcson van, másik méréskor pedig egy alacsonyabb értéken. Ekkor ha valamelyik hálózatról működő eszköz erőtere bezavarna

a mérésbe akkor összevissza eredményeket kapnánk. Így viszont állandóan számolunk a lehetséges hibával és átlagként egyenletes eredményeket kapunk.



11. ábra: 3D interfész kapcsolása⁴

A mért értékek a 4.4 fejezetben ismertetett módon jutnak a LabVIEW programba. Az RC kör időállandójáról valós számítás vagy mérés nincsen, bár oszcilloszkóppal érdemes lenne megvizsgálni a jelet, hogy megtudjuk mégis mekkorák a szóban forgó kapacitásértékek. A PIC program által mért értékek alapján a program kalibráció után dönti el, hogy milyen tartományban változnak a mérési értékek, majd ezután egyszerűen ezt a tartományt linearizálja és átskálazza a kívánt vezérlési tartományra, és ezekkel az értékekkel vezérli a robot inverz kinematikáját számoló alprogramot. Célszerű kisebb munkateret meghatározni a robotnak, mint amit valóban mozgunk a kezünkkel, mert ennek a szenzornak a pontossága még nem túl megbízható valamint nem is tudjuk pontosan milyen koordinátára mozgatjuk a kezünket.

Egy megoldatlan probléma, hogy amint most már két kommunikációs portot használ a számítógép, a kommunikáció a robot és a komputer között indokolatlanul belassul. Ez a szokottnál is sokkal darabosabbá teszi a robot mozgását. A probléma okát a dolgozat megírásáig nem sikerült felderíteni.

Videó a szenzorról működés közben: <http://youtu.be/eNQ4SWTZkHA>

⁴ <http://guide-images.makeprojects.org/igi/vnMhkLZVqZmA2rQG.large>

8. Összefoglalás és következtetések

A dolgozatban áttekintettük a robot fizikai felépítését és működését. Megismerkedtünk a szervók működésével és vezérlőjelük előállítási módjaival. Olvashattuk hogyan figyeli a program a szervók pozícióját és képes ez alapján mozgáspálya rögzítésére és visszajátszására, továbbá programozott Point To Point mozgássorozat lejátszására vezérelve a robot kimeneteit is és figyelembe véve a bemeneteket. A dolgozat ismertette a robot direkt és inverz kinematikájának megoldását is, melynek segítségével vezérlehető világkoordináták megadásával is a robot. Végezetül megnéztük hogyan illeszthető be a robot egy valósághű automatizálási rendszerbe, valamint hogyan lehet egy nem hétköznapi interfésszel vezérelni a robotot.

Természetesen a robot ezek közül minden területen fejleszthető és korrigálható. Látható a mellékelt videókban, hogy a robot mozgása kissé darabos. Ezt a szervók alacsony felbontásának köszönhető. Erősebb, nagyobb felbontású és pontosabb szervókkal ezen a mozgáson sokat lehetne finomítani, de ez elég költséges fejlesztés lenne.

Az inverz kinematikai alprogram korrigálásával és fejlesztésével meg lehetne valósítani, hogy ne csak világkoordinátákkal vezéreljük a robotot, hanem akár a szerszám koordináta rendszerét, vagy bázis koordinátarendszereket alapul véve is. Továbbá egy ilyen fejlettebb inverz kinematikai számítás segítségével lehetőség lenne nem csak PTP, hanem lineáris illetve köríves mozgások programozására is.

A fejlesztési lehetőségek igazából határtalanok, melyek túlmutatnak a dolgozat keretein. Például a robot vezérelhető lenne wifi hálózaton keresztül okostelefonról, akár gombok, vagy a telefon gyorsulásérzékelőinek a segítségével. Vagy interneten keresztül elérhető lenne a robot vezérlése, melyet webkamerák tennének használhatóvá. Az utóbbira a LabVIEW komplett megoldást kínál, de az előbbihez másféle programozási ismeretekre is szükség lenne.

Ez a sok fejlesztési lehetőség idővel el is szakad ettől a robottól és kiderül, hogy minden területen vannak lehetőségek kisebb-nagyobb projektekre. Ez ugyan nyilvánvaló, mégse végez hasonló munkákat kedvtelésből minden mérnökhallgató, pedig épp ez az amitől igazán tudnának ők is fejlődni az egyetemi tanulmányok mellett is. Remélem sikerült elérni a dolgozat egyik célkitűzését, hogy kedvet ébresszen hasonló szerkezetek építésére és programozására más diákokban is, akik olvasták ezt a dolgozatot. Bízom benne, hogy a dolgozat segítséget is tud nekik nyújtani az elindulásban és mérnökként ők is többször juthatnak hozzá az olyan alkotó munka öröméhez, melyek eredménye az, hogy a gép forog, az alkotó pihen.

Irodalomjegyzék

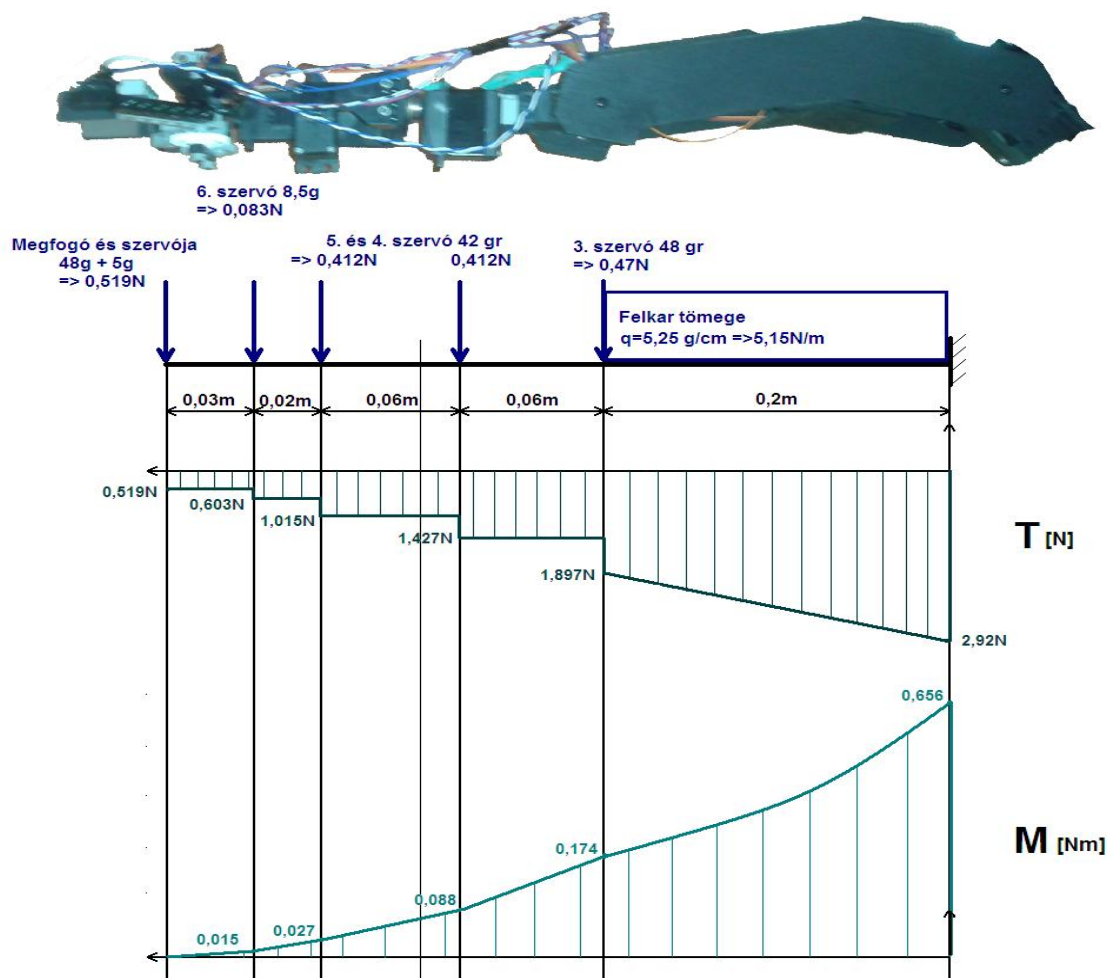
- [1] Dr. Rudas Imre, Dr. Bencsik Attila: Robottechnika, 1999, Bánki Donát Műszaki Főiskola
- [2] Servo tutorial http://www.pyroelectro.com/tutorials/servo_motor/index.html
- [3] Microchip Technology Inc: PIC16F87X Data Sheet, 2001, DS30292C
- [4] Dr. Mester Gyula: Robotmanipulátorok kinematikája, Szegedi Tudományegyetem SZÉF,
http://siva.banki.hu/jegyzetek/Mechatronikai_alapismeretek/Ip_Robotrendszerek/Robotkinematika-1.pdf
- [5] Prof. Dr.-Ing. Konrad Wöllhaf: Introduction Robotics, HS-Weingarten, March 16, 2010
- [6] Pécsi Tudományegyetem Pollach Mihály Műszaki és Informatikai kar: Robotok irányítása
<http://e-oktat.pmmf.hu/robotiranyitas>, 7. rész
- [7] Kyle McDonald: DIY 3D Controller
<http://www.instructables.com/id/DIY-3D-Controller/?ALLSTEPS>

Mellékletek

I. Melléklet: A robot szervóinak választása és mechanikai ellenőrzésük

(I/a) Táblázat: A választott szervók

Szervó száma	Tipus	Súly	Nyomaték [kg*cm]	Nyomaték [Nm]
1	TowerPro SG5010	42 g	5,2	0,509
2	TowerPro MG995	48 g	11	1,078
3	TowerPro MG995	48 g	11	1,078
4	TowerPro SG5010	42 g	5,2	0,509
5	TowerPro SG5010	42 g	5,2	0,509
6	ECO Micro 11 pro	8 g	2,4	0,235
Megfogó	HEXTRONIK HXT900	5 g	1,6	0,156

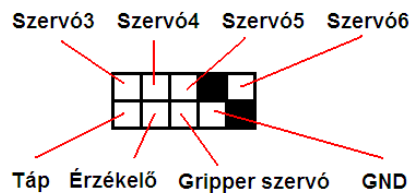


(I/b) Szervó ellenőrző számítás nyíróerő és hajlító nyomatéki ábrája.

II. Melléklet: A vezérlőpanel és a kábeleinek kiosztása

II/a/1: Vezetékek a vezérlőpanelre forrasztva

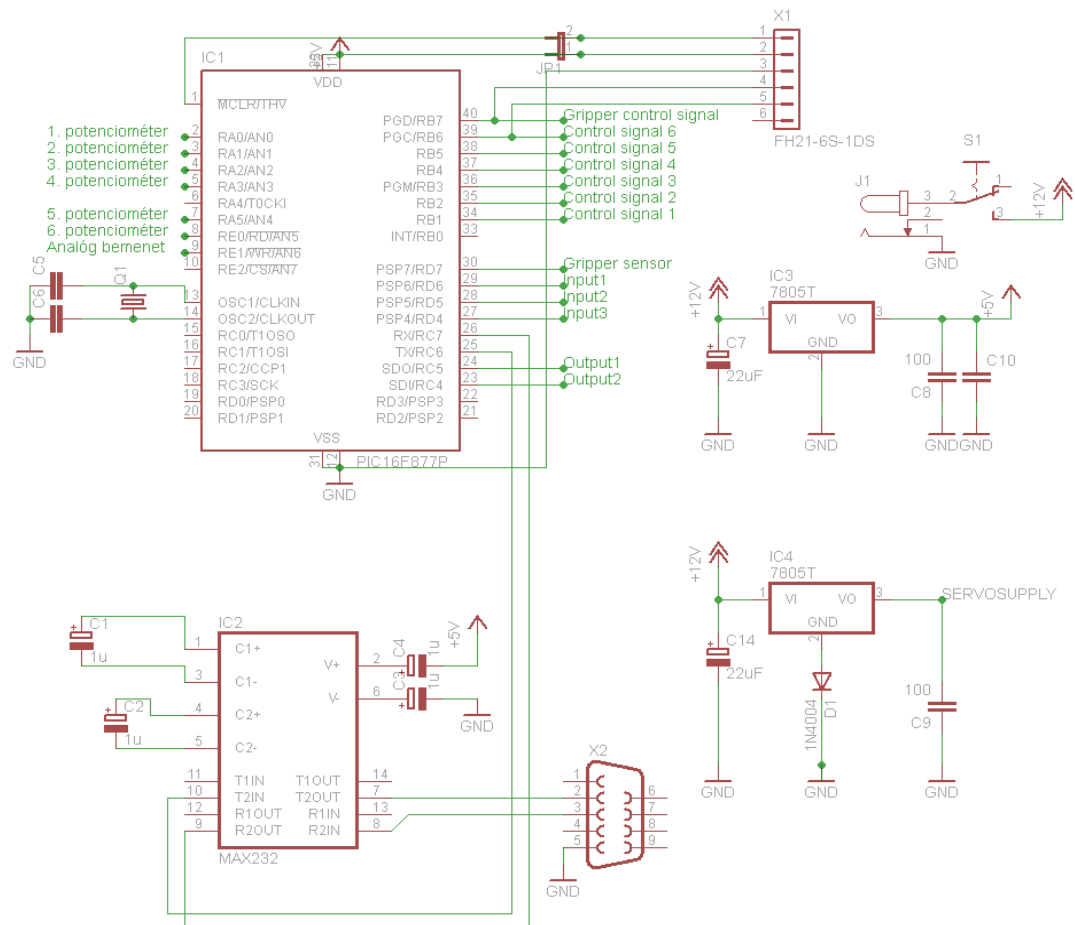
Különálló vezetékek		Kötegelt vezetékek					
Piros	Fekere	Piros	Fehér1	Zöld1	Barna	Fehér2	Zöld2
Szervó 3	Szervó 6	Táp	Érzékelő	Gripper szervó	GND	Szervó 4	Szervó 5



II/a/2: A vezérlő kábelek csatlakozójának kiosztása

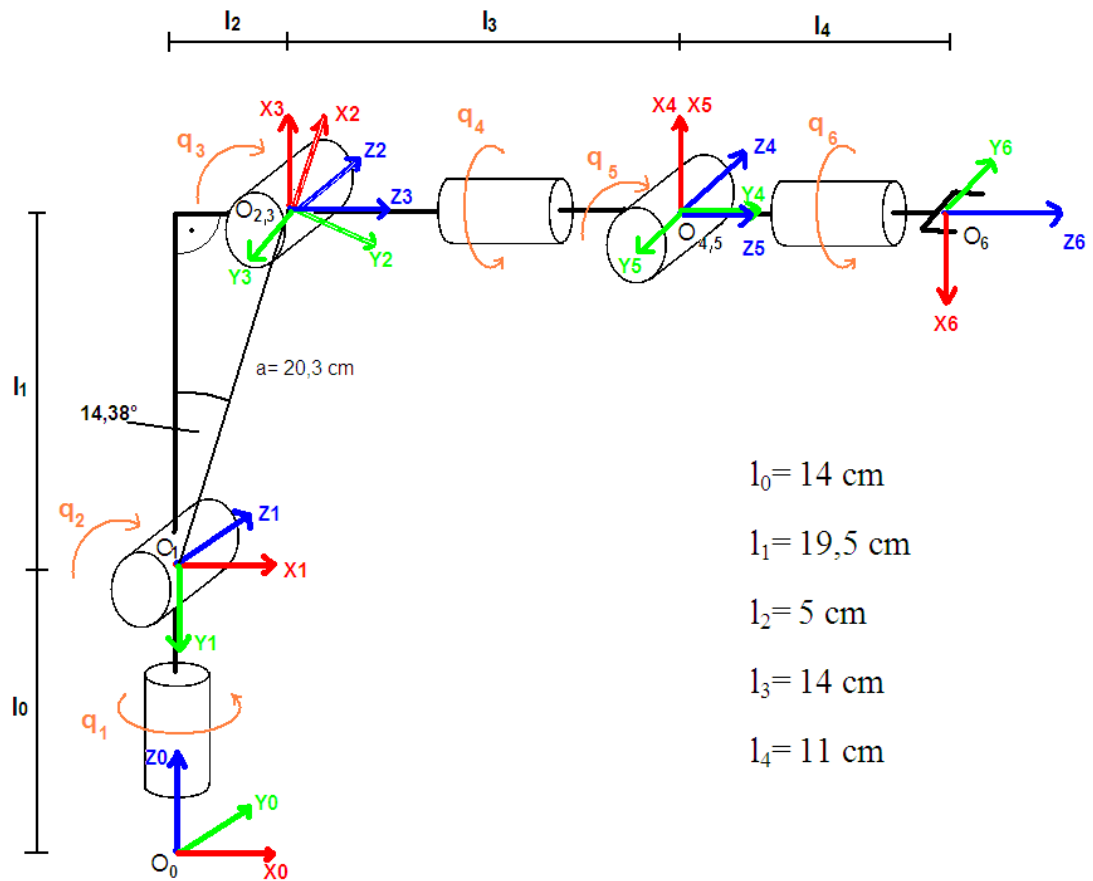
II/a/3: Potenciométerre forrasztott vezetékek

Fehér	Zöld	Kék	Sárga
3. Potenciométer	4. Potenciométer	5. Potenciométer	6. Potenciométer



II/b: Vezérlőpanel kapcsolási rajza

III. Melléklet: A robot direkt kinematikája



III/a: A robotkar referencia koordinátarendszerei és méretei

III/b: A robot D-H paramétereit

i	q_i	d_i (cm)	a_i (cm)	α_i
1	q_1	14	0	-90°
2	$q_2 - 75,616^\circ$	0	20,13	0°
3	$q_3 - 14,381^\circ$	0	0	-90°
4	q_4	14	0	90°
5	q_5	0	0	-90°
6	$q_6 + 180^\circ$	11	0	0°

IV. Melléklet: A szortírozó vezérlésének folyamatábrája