NAPRA-FORGÓ ROBOT AUTOMATIKUS VEZÉRLÉSSEL NAPENERGIA HASZNOSÍTÁSÁHOZ

SOLAR TRACKER ROBOT WITH AUTOMATED CONTROL FOR SOLAR ENERGY UTILIZATION

SZAKDOLGOZAT

BMEGEMIA4SD

Bódi László D4W2KH

Mechatronikai mérnök Bsc. 2N-AM0

2010. november. 31.

„Parta szállóttam, levonom vitorlám,

A szelek mérgét nemesen kiálltam.

Sok Charybdis közt, sokezer veszélyben

Izzada orcám.”

Berzsenyi Dániel: Osztályrészem

Jogi nyilatkozat

Alulírott Bódi László, a Budapesti Műszaki és Gazdaságtudományi Egyetem hallgatója, kijelentem, hogy ezt a szakdolgozatot meg nem engedett segítség nélkül, saját magam készítettem, és a szakdolgozatban csak a megadott forrásokat használtam fel.

Minden olyan részt, amelyet szó szerint, vagy azonos értelemben, de átfogalmazva más forrásból átvettem, egyértelműen, a forrás megadásával jelöltem.

Tartalomjegyzék

0. Bevezetés

A nem megújuló energiakészleteink fogyása következtében egyre jobban előtérbe kerülnek az alternatív energiaelőállítási módok, így például a napelemek is. Az általánosan használt napelemek hatásfoka 10% körül van, és a legújabb fejlesztésű napelem is, mely még csak kísérleti fázisban van, csak a ráeső fény energiájának 40%-át képes hasznosítani. Továbbá egy statikusan rögzített napelem energiatermelése nagy részben függ a Nappal bezárt szögétől. Ezen probléma kiküszöbölésére nyújt segítséget az általam tervezett szerkezet, azzal, hogy a napelemet mindig a fény intenzitásnak maximuma felé fordítja. Szakirányom a biomechatronika, melynek szerves része a biomimetika, azaz a biológiai rendszerek műszaki utánzása. Ez adta az ötletet a napra-forgó robot megvalósításához. Dolgozatomban a napelemek fejlesztésének jelenlegi állapotának elemzése, a biomimetika és az engem megihlető növény, a napraforgó bemutatása után a mechatronikai hármas alapján mutatom be szerkezetem megvalósítását.

Dolgozatom megírásához hathatós segítséget kaptam a BME Mechatronika, Optika és Gépészeti informatika tanszéke, valamint az Optikai Mérnökiroda Kft. részéről. Külön köszönet a Biomechatronika szakirányfelelősének, Dr. Aradi Petrának, konzulensemnek Dr. G. Szabó Istvánnak és munkatársának, Bujáki Krisztiánnak. Köszönettel tartozom Dr. Lipovszki Györgynek, a digitális szabályozás kidolgozásában nyújtott segítségéért és Dr. Sütő Zoltánnak, aki az elektronikai tervezőprogram használatában nyújtott segítséget.

0. Introduction

Due to the depletion of the non-renewable energy supplies, more and more emphasis is being put on the alternative energy production methods, such as solar panels. The efficiency of the commonly used solar cells is around 10%; and the latest development, still in an experimental stage, can only use the 40% of the incoming light's energy. In addition, a static fixed solar cell's energy production largely depends on the angle of it closes with the Sun. In an attempt to help solve this problem, I designed a structure to rotate the solar cell in the direction of light's maximum intensity. My specialization is the Biomechatronics, and its integral part called Biomimetics - technical imitation of biological systems - gave the idea to create such a structure. In my thesis, after giving an overview of Sun tracking methods, presenting Biomimetics, and sunflower, the plant which inspired me, I am going to present my structure's implementation, in a way based on the definition of Mechatronics.

While writing this thesis I have received valuable assistance of the Department of Mechatronics, Optics and Engineering Informatics of BME and the Optical Engineering Ltd. Special thanks to, Dr. Petra Aradi, Dr. István G. Szabó and Krisztián Bujáki. I am grateful to Dr. György Lipovszki, for assistance with the development of the digital control and Dr. Zoltán Sütő, who provided assistance in the use of the electronic design program.

0.1. Biomimetika

A biomimikri, vagy biomimetika, a természeti modellek, rendszerek, folyamatok és elemek vizsgálata, azon célból, hogy utánozni tudjuk vagy inspirációt nyerjünk belőle az emberi problémák megoldásához. A biomimikri és biomimetika kifejezések görög szavakból származnak: bios, mint „élet", és mimézis, mint „azt utánozni". Gyakran használnak egyéb rokon értelmű kifejezéseket, mint bionika, bio-inspiráció, és biognosiz.

Az emberek mindig a természethez fordultak inspirációért, hogy meg tudják oldani a problémáikat. A biomimikri egyik korai példája a madarak röptének tanulmányozása, hogy megvalósítsák az emberi repülést. Bár sohasem sikerült megteremtenie a „repülő gépet", Leonardo da Vinci (1452-1519) lelkes megfigyelője volt a madarak anatómiájának és repülésének. Számos jegyzetet és vázlatot készített megfigyeléseiről, valamint a vázlatokat készített különböző „repülő gépezetekről". A Wright-fivérek, akiknek végül sikerült létrehozni az első repülőgépet 1903-ban, az ihletet a galambok repülésből nyerték.

A „bionika" kifejezést a pszichiáter és mérnök Jack Steele alkotta. Azt jelenti, a definíciója szerint, hogy „a bionika olyan rendszerek tudománya, amelyek valamilyen funkciót természetről másolták". Maga a bionika úgy lépett be a Webster szótárba 1960-ban, mint „olyan tudomány, mely biológiai rendszerek vizsgálatával adatokat gyűjt a műszaki problémák megoldására".

Maga a Biomimetika kifejezés csak 1974-ben került be a Webster szótárba, azzal a definícióval, hogy „tanulmányozza a kialakulását, szerkezetét, működését a biológiai úton előállított anyagoknak és eszközöknek (például enzimek vagy selyem) és tanulmányozza a biológiai mechanizmusokat és folyamatokat (fehérjeszintézis, fotoszintézis), különösen abból a célból, hogy hasonló termékeket mesterséges, a természetest utánzó mechanizmusok segítségével szintetizálhassunk". [43]

Példák:[42][43]

- Tépőzár
- Cápabőrhöz hasonló mintázattal ellátott anyagú úszódressz
- Zimbabwei Eastgate Centre, mely termeszvár megfigyelése alapján épült, hűvös marad légkondicionálás nélkül
- Öntisztuló, vízlepergető ruha, ami a lótusz levél felszínét utánozza
- Dragon Skin hajlékony testpáncél, mely a Nílusi sokúszós csuka kültakaróját utánozza

0.1.1. Napraforgó

(Helianthus annuus) Amerikában őshonos egynyári növény. Nagy, összetett virágzattal rendelkezik, amelyben számos kis virág zsúfolódik össze. Ivartalan virágok alkotják a külső szirmot, melyek sárga, piros, narancssárga, vagy más színűek lehetnek. A fej belső kör alakú részében, vannak az úgynevezett lemez virágok, ezek hozzák létre az érett magot. A fejen a virágok spirális mintában vannak elrendezve, úgy, hogy egymással 137,5° zárnak be egymással, ez az aranymetszés szöge, így olyan egymást összekötő spirálos mintát állítanak elő, hogy a bal és jobb spirálok száma egymást követő Fibonacci-számok. Általában 34 spirál van az egyik irányban, és 55 a másikban, de egy nagyon nagy napraforgónál akár 89 is lehet az egyik irányban és 144 a másikban. Ez a virág fejében lévő minta valósítja meg a magvak leghatékonyabb csomagolását.

Napraforgók leggyakrabban 1,5 és 3,5 m közötti magasságúra nőnek meg, szakirodalom megemlíti, hogy 1567-ben, Padovában egy 12 m magas hagyományos, egyetlen fejű napraforgó termett. Ugyanezzel a vetőmag típussal majdnem 8 méter magasságot értek el egyéb helyeken, például a Madridban. A 20. század folyamán a 8 méteres magasságot értek el mind Hollandiában és Kanadában.[12]

0.1.1.1. Heliotropizmus

A virágzatnak a napfényre adott fototropikus reakciója van, amelyet heliotropizmusnak nevezünk. A fiatal napraforgó levelei és virágfeje követi a Nap irányának változását, tehát keletről nyugatra változtatja az orientációját a nappal folyamán, mert a virág és levelek szárai folyamatosan nőnek és ezzel mindig a fény felé fordítják a növényt. Ha növény megérett, a mozgás leáll. E jelenségnek oka, hogy a fény bontja a növények növekedési hormonját, így a fénynek kitett részek nem fognak nőni, viszont a kevesebb fényt kapó részek igen. A napraforgónál a szárat alkotó sejtek keresztirányban helyezkednek el. Ha ezek nőnek, mivel a sötét oldalon vannak, akkor mindig a fény felé fordítják a növényt. A mozgás egy idő után részévé válik cirkadián ritmusnak, így ha növényt 180 fokkal elforgatjuk, a régi mintát fogja követni pár napig, azaz levelek irányultsága inkább nyugatról keletre fog változni. [12]

0.2. Napelemek

A napelem olyan szilárdtest eszköz, amely a fénysugárzás energiáját közvetlenül villamos energiává alakítja. Az energiaátalakítás alapja, hogy a fény elnyelődésekor mozgásképes töltött részecskéket generál, melyeket az eszközben az elektrokémiai potenciálok, illetve az elektron kilépési munkák különbözőségéből adódó beépített elektromos tér rendezett mozgásra kényszerít, vagyis elektromos áram jön létre.[13][14]

0.2.1.1. Hatásfok [14]

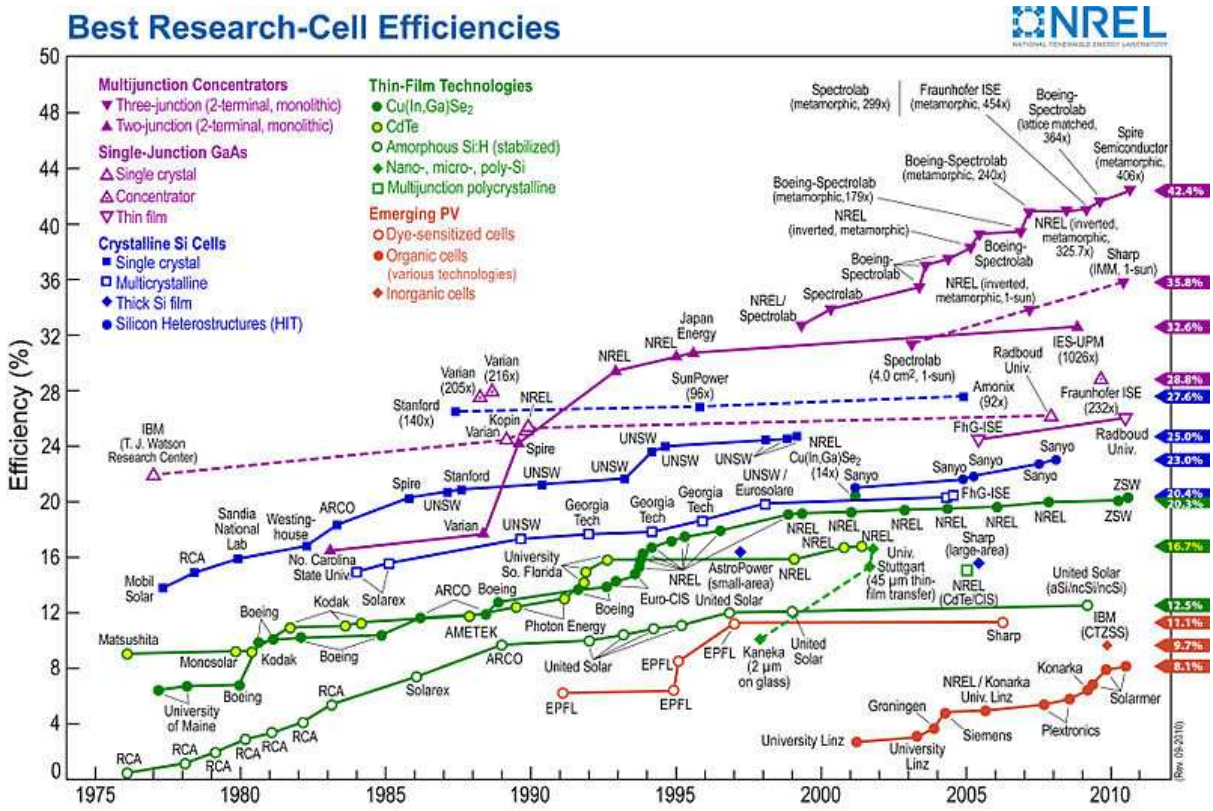
$$\eta = \frac{P_m}{E * A_c}$$

Ahol:

- P_m a fényelem által leadott maximális teljesítmény,
- E a napsugárzás energiája (W/m^2),
- A_c a napelem felülete (m^2)

0.2.1.2. Fejlesztések

Az alábbi diagramon láthatóak a kifejlesztett típusok és hatékonyságuk növekedése.



1. ábra: Legjobb napelem fejlesztések [45]

Látható, hogy kísérleti körülmények között az MC (multijunction concentrator) napelemek, már 42%-os hatásfok fölött járnak, viszont ezeket főleg műholdak energiaellátására fejlesztették ki, ahol az ár sokadrangú tényező, árhatásfokuk ($/W) alacsony. A gazdaságos PV (photovoltatic) napcellák hatékonysága még mindig 10% körül van.[45]

Ezeknek a hatásfokának javítása érdekében, az én szerkezetemhez hasonló, forgatási rendszereket hoztak létre, hogy kövessék a nap forgási irányát.

0.3. Napkövető rendszerek típusai

A napkövetők osztályozhatóak a tengelyek száma és iránya alapján. Szemben a nem mozgó kialakításokkal, az egytengelyesek éves termelése mintegy 30%-kal nagyobb, míg a kéttengelyeseké még további 6%-kal jobb.

0.3.1. Egy tengelyes követők

2. ábra:Wattsun HZ lineáris vízszintes napkövető Dél-Koreában és SunPower T20 döntött egytengelyes napkövető Nevadában

0.3.1.1. Vízszintes egytengelyű követők (HSAT)

A forgástengelye vízszintes Tengelyek rögzítőit a két végén meg lehet osztani többivel, ami csökkenti a telepítés költségeit, és így a forgatásukat is meg lehet oldani kis számú motorral. Ezen túlmenően, a visszalépéses algoritmussal bármilyen sűrűséggel lehet telepíteni őket egymás árnyékolása nélkül. Napcellái párhuzamosak a forgástengellyel.

0.3.1.2. Függőleges egytengelyű követő (VSAT)

A forgástengelyük függőleges, vagy kis szöget zár be a függőlegessel. Keletről nyugatra fordulnak a nappal folyamán. A mezők elrendezésénél figyelni kell a vetett árnyékokra, hogy elkerüljék a felesleges energia veszteséget, és optimalizálják a földhasználatot. Napcellái merőlegesek a forgástengellyel.

0.3.1.3. Döntött egytengelyes követő (TSAT)

Visszalépéses algoritmussal a forgástengelyükre merőlegesen bármilyen sűrűséggel , árnyékolás nélkül lehet telepíteni, viszont párhuzamosan a telepítést korlátozza a szélességük és döntöttségük.

0.3.1.4. Polár igazított egytengelyű követő (PASAT)

Ez egy tudományosan érdekes variáció döntött egytengelyűre, ahol a dőlésszöge megegyezik a szélességi fokkal. Nagy légellenállása miatt ritkán alkalmazzák.

0.3.2. Kéttengelyű követők

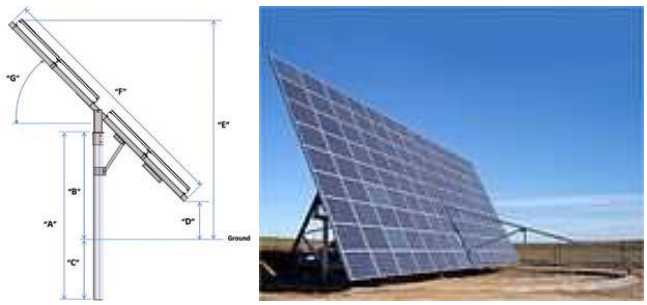
Két szabadsági fokkal rendelkeznek, tengelyeik általában derékszöget zárnak be egymással. A földre rögzített tengely nevezhető az elsődlegesnek, az ehhez kapcsolódó a másodlagos tengely. Számos elterjedt megvalósításuk van, besorolásuk az elsődleges tengely alapján történik. Két leggyakoribb az ún. Tip-Tilt és az Azimuth-altitude követők.

0.3.2.1. Tip-Tilt (csúcson-döntött) kéttengelyű követő (TTDAT)

Az elsődleges tengelye vízszintes, a másodlagos általában erre merőleges. A mezőkialakítások a nagyon rugalmasak. Visszalépéses algoritmussal bármilyen sűrűségben, egymás árnyékolása nélkül lehet telepíteni.

0.3.2.2. Azimuth-Altitude (irányszög-magasság) kéttengelyű követő (AADAT)

Az elsődleges tengelye függőleges, erre merőleges a másodlagos tengelye. A napelem nem az elsődleges tengelyen van, hanem attól távolabb van rögzítve, így a napelem nem a saját tengelye körül forog, hanem kört ír le az elsődleges tengely körül. A másodlagos tengely sima vízszintes forgástengely. Ezt a szerkezetet használják a nagy távcsövek forgatására is. Mezőelrendezésnél gondolni kell az árnyékokra, hogy elkerüljék a felesleges energia veszteséget, és optimalizálják földhasználatot. [44]



3. ábra: Patriot Solar Group 2 tengelyes követő és Azimuth-Altitude Kéttengelyű követő Toledo, Spanyolország

1. Feladat feldolgozása

Feladatom tehát, egy olyan kéttengelyes forgató automatika megalkotása, mely egy napelemet a fény intenzitás maximumába fordítja.

1.1.1.1. Eszköz részfeladatai

- Fénymérés: beeső fény intenzitásának a mérése, átalakítása analóg elektromos jellé,
- Analóg jel digitálissá konvertálása.
- A bejövő jelből aktuátorok mozgatásához megfelelő jel kialakítása
- A napelem mozgatása a megfelelő pozícióba.

1.1.1.2. Tervezés menete

Elinduláshoz tájékozódni kell a megoldási lehetőségekről, és össze kell gyűjteni a megvalósításhoz szükséges szakirodalmat.

Először meg kell határozni a forgatási mechanizmust, ki kell választani az aktuátorokat, majd meg kell tervezni a berendezést.

Ki kell választani a megfelelő vezérlő egységet. Ehhez analóg szabályzó helyett digitális mikrokontrollert érdemes alkalmazni, mert programozhatósága miatt többféle szabályozási eljárás is könnyen megvalósítható.

Ezután a fénymérő szenzor kiválasztását kell megejteni. A napelem maga erre nem alkalmas, mert a belőle kijövő feszültség a beeső fény függvényében tág határok közt, nemlineárisan változhat.

Az elektronikai kapcsolás és a programírás igazából csak egyidejűleg lehetséges, mert a kitalált programoknál változhat az elektronikai konfiguráció.

Ha már előrehaladott állapotban van a szoftver, ahhoz igazítva el lehet kezdeni az alkatrészek legyártását és összeszerelését.

Közben folyamatosan vezetni kell a tervezői füzetet, hiszen később a dolgozat megírásához jó alapot nyújt.

Ha már előrehaladott állapotban van a megvalósítás, akkor el lehet kezdeni a szakdolgozat megírását.

2. Hardver

2.1. Mechanika

2.1.1.1. Tervező program

Tervezéshez az *Autodesk Inventor* legújabb verzióját használtam. E program kezelését a *CAD alapjai* és ennek folytatása a *CAD modellezés* c. tárgyak elvégzésével sajátítottam el. Ez egy nagyon kényelmes és okos 3D tervező program, melynek készítésekor irdatlan nagy hangsúlyt fektettek a szoftverergonómiára. Egyik fő előnye, hogy egyetemi email címmel regisztrálva a teljes verzió letölthető az *Autodesk Education Community* honlapjáról [16], más *Autodesk* által készített tervező és modellező programokkal egyetemben (pl. *ElectroCAD, Maya, 3Ds Max*).

2.1.2. Mechanika terv

Elején egy nehezen megvalósítható, de az előzőekben nem említett, két, vízszintes forgató tengellyel rendelkező megoldást képzeltem el. Az ötletet aztán a konzulensemmel folytatatott konzultáció után vetettem el. Ő ajánlotta, hogy egyszerűen egy z irányú csapágyházra merőlegesen még egy csapágyházat illesszek rá, csúcsdöntött kialakítást eredményez. Ezután akadtam rá erre a szervomotorokból álló játékrobot ízületre [31], ez ihlette a konstrukció további tervezését:

4. ábra: Lynxmotion robot ízület

2.1.2.1. Motor típus kiválasztása

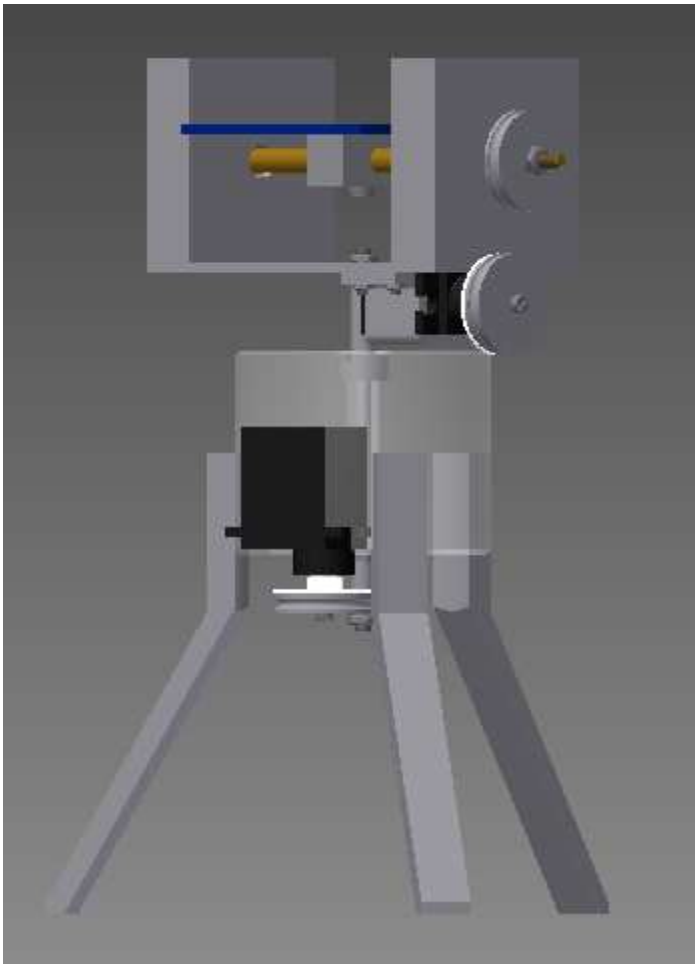
Ötletezés elején felmerült bennem, hogy sima RC motorokkal vagy léptető motorokkal oldjam meg a mozgatást, viszont mindkét megoldáshoz szükség lett volna valamilyen pozíciómérésre alkalmas szenzorra, amelynek a beépítése mellett a szabályzását is meg kellett volna oldanom. Ezért nyúltam az egyszerűbb megoldáshoz, az RC szervomotorokhoz, melyben a pozíció-mérést és - szabályzást megoldja a benne lévő elektronika, amihez nekem csak egy a szöghelyzettel összefüggő digitális jelet kell kiadnom.

2.1.2.2. Terv kialakítása

Először megterveztem a z irányú forgatást: a csapágyházat rúdanyagból alakítottam ki, azért mert körszimmetrikus és így lehet esztergálni, ami sokkal könnyebb gyártási művelet, mint a marás. Viszont a szervomotornak bemarással kellett csinálni fészket.

E után a második csapágyházat terveztem meg. Annak egy lapot kell majd forgatnia x irányban, így egy U-profil jellegű dolgot találtam ki, melynek középső eleme egy lap, két oldalfala pedig, melyben a csapágyak vannak, vastagabb négyzet alapú hasáb, hogy négypofás tokmányban megfogva esztergálni lehessen. Ehhez a csapágyházhoz alulról egy lemezalkatrésszel rögzítettem a szervót. A második csapágyház lapjának közepére fúrt lyukon keresztül az első csapágyház tengelyvállán a tengelyre vágott menettel és anyával rögzítettem. Az elfordulás ellen ragasztást alkalmaztam.

A szerkezet talajra való megállását három, az alsó csapágyházra bilinccsel rögzített láb adja.



5. ábra: Kész mechanika terv

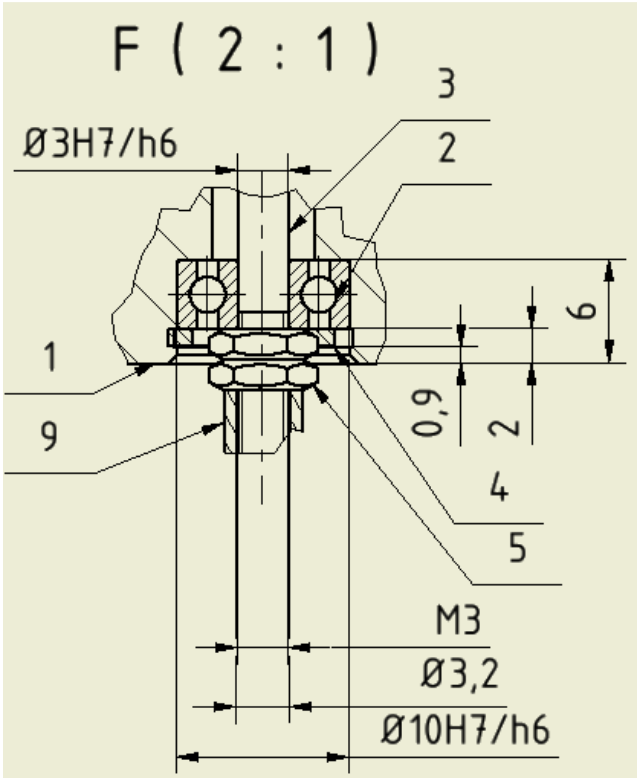
A gépezetet fontosabb elemeinek tervezéséről, szerkezeti kialakításáról, valamint ellenőrzéséről a továbbiakban lesz szó.

2.1.3. Hajtásrendszer

Először úgy gondoltam, hogy fogaskereket fogok használni a hajtás átadására, de miután utána jártam, hogy a fogaskerekek legyártási és szállítási ideje minimum 3-4 hét [30], ezért letettem a fogaskerékhajtásról. Így jött a képbe a szíj és dörzshajtás. Szakmai gyakorlaton egy forgatás megvalósításához már alkalmaztam mindkettőt, így nem volt ismeretlen számomra a kialakításuk. A szíjhajtást választottam. Amíg kis méretekről van szó, a szíjhajtás lehető legegyszerűbb megvalósítása o-gyűrűkkel történhet, mert alapból végtelenítve vannak, és jól bírják az igénybevételt. Érdemes 2mm vastag o-gyűrűkkel dolgozni, mert szinte minden átmérőben kaphatóak [29].

2.1.3.1. Csapágyelrendezés

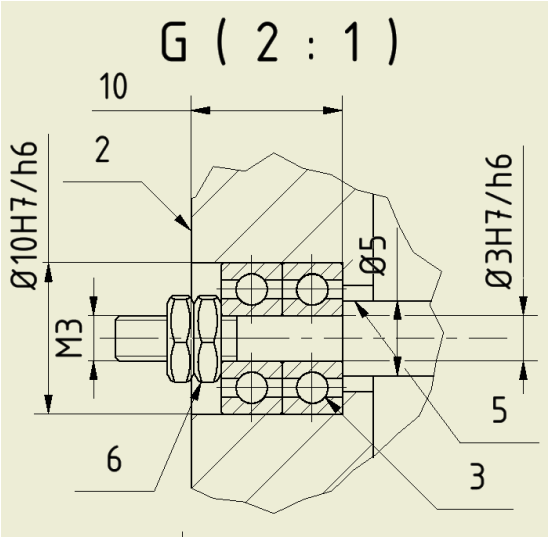
Golyóscsapágyat alapvetően nem választottam, hanem megkérdeztem milyen csapágyak vannak a műhelyben, és kezembe nyomtak egy dobozzal az SKF-623-as mélyhornyú golyóscsapágyból [2]. A mélyhornyú golyóscsapágy megfelel a feladatra, mert fel tud venni axiális terhelést is, így nem kellett axiális csapágyat beszereznem. Mivel az SKF 623-nak a belső futógyűrűjének belső átmérője 3mm, a külsője pedig 5,5mm (ld. 9. ábra), így a tengelyeimnek alapvetően vékonynak kellett lenniük. Viszont hosszúnak is a vastagságukhoz képest, egyrészt azért, mert az alsó házba bele akartam illeszteni a szervomotort, másrészt a felső házban a tengelyre akartam rakni a napelemet. Így az először kitalált alsó csapágyházban egyszeres O-elrendezést alkalmaztam: a csapágy külső futógyűrűjét egyik oldalon a csapágyházban kialakított fészek, a másik oldalon Seeger-gyűrű rögzíti. A belső futógyűrűjét a kinti oldalon alul tengelyre vágott menet és rácsavart anya, felül pedig egy tengelyváll rögzíti, így alakul ki az O-elrendezés:



6. ábra: Alsó ház csapágyrögzítés

Már a gyártás kezdetén láttam, hogy gondok lesznek majd a terheléssel, főleg, hogy axiálisan vannak terhelve ezért a későbbiekben szó lesz ezen csapágyak ellenőrzéséről is.

Mivel úgy gondoltam, hogy a csapágyak egyedül nem bírják a terhelést ezért a felső csapágyházban már tandem O-elrendezést alkalmaztam. A csapágy külső futógyűrűjét csak a belső oldalon a csapágyházban kialakított fészek fogja meg. A belső futógyűrűjét a kinti oldalon tengelyre vágott menet és rácsavart anya, belül pedig egy tengelyváll rögzíti, így alakul ki az O-elrendezés:



7. ábra: Felső ház csapágyrögzítés

A csapágyelrendezés kialakítását a *Simon és tsai.: Gépelemek 2* [4] c. tankönyv alapján készítettem. Az illesztéseket a *Házkötő István: Műszaki 2D-s ábrázolás* [5] c. jegyzetből választottam ki.

2.1.3.2. Hajtásátadás

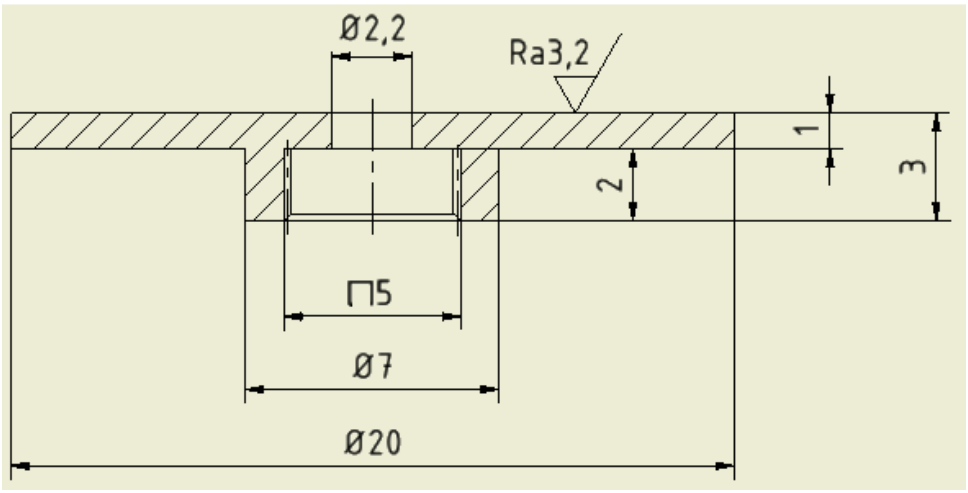
Az áttétel kiszámításához ismernünk kell a szervómotorok kimenő tengelyének szögelfordulás tartományát. Bár a szervomotorok kiválasztásáról és beméréséről később lesz szó, a motorok kb. 150°-180°-os tartománnyal rendelkeznek, tehát ehhez a szöghöz kellett méreteznem az áttételt.

A motor kimenő tengelye bordás tengely, alighanem szándékosan nem szabványméretű, hogy ne lehessen másik gyár alkatrészeit használni hozzá. Ezért nem lehet kapni egyik motorhoz sem hajtóművet, fogaskereket, mert minden gyártóéhoz más méret kéne, és az alapfelhasználó ezt RC modellek távvezérlésére akarja használni, amihez elég ez a szögtartomány is.

Viszont a motorhoz járt egy adag tengelykapcsolónak is nevezhető kiegészítő:

8. ábra: Szervó tengelykapcsolók

Ezek közül az egyik tárcsával rendelkezik. Ennek a tárcsának az átmérője 20mm, és erre akartam rárakni a hajtókereket, így annak az átmérőjét szükségképpen 20 mm-re választottam. Ha 2mm vastag o-gyűrűt akarok használni, akkor 1 mm mély hornyot kell bele vágnom, így a tárcsa hajtó átmérője 18 mm lett:



9. ábra: Tengelykapcsoló

Az alsó meghajtásnál, ami a Z irányú forgatást valósítja meg, teljes körbefordulásra van szükség. Ha a szervomotor maximális, 150°-os szögelfordulásból 360°-ot akarok csinálni, akkor 2,4-es áttétel kel, amihez a hajtott kerék átmérője 18mm/2,4=7,5 mm. Ha ehhez hozzáadjuk a horony miatt levágódó 2mm-t, 9,5 kapunk, így választottam hajtott kerék 1-nek a 10mm-es átmérőt (ld. 4. ábra).

A felső hajtásnál viszont nem kell ekkora szögelfordulás, mert a felső csapágyház kialakítása miatt nem tud teljes körben elfordulni, így jó lesz a 150°-os szögterjedelem is. Ezért a hajtott kerék 2-t 20mm átmérőjűnek választottam a hajtott kerék 2-t (ld. 5 ábra).

2.1.3.3. Tárcsák rögzítése

Mivel eléggé kis méretekről van szó, számomra a szokványos nyomatékátadási kialakítások gyártási megvalósításra problémás lett volna (konkrétan nem tudok olyan pontosan fúrni, marni és esztergálni), így a tervezés során ezeket nem alkalmazhattam.

A hajtókerekeket úgy rögzítettem, hogy a szervómotor kimenő bordás tengelyére húztam föl a velekapott tárcsás tengelykapcsolót, aminek a tetejét előzőleg vízszintese csiszoltam. A bordás tengely közepén van egy M2 furat, amihez a tengelykapcsolón is van 2,2mm-es átmenő furat, így a központosítást egy M2 es csavarral tudtam megoldani, úgy, hogy a hajtótárcsát középen szintén átfúrtam 2,2mmes fúróval. A nyomatékot a súrlódási erő a tárcsa és a tengelykapcsoló között nem tudja átadni, ezért ezt is meg kellett oldani: átfúrhattam volna még egy ponton a tengelykapcsolót és a tárcsát, hogy egy csavarkötéssel adjam át a nyomatékot, de inkább az egyszerűbb utat választottam és összeragasztottam őket. Mivel kis erőkről, és viszonylag nagy felületről van szó, ezért ez is megfelel a célnak.

A tengelyekre a hajtott kerekek központosítását szintén egy átmenő furattal, a tengelyre vágott M3-as menettel és dupla csavaranyával oldottam meg. A nyomatékátadást szintén ragasztott kötés biztosítja

2.1.3.4. Ragasztott kötés ellenőrzése

A ragasztáshoz Loctite nagyszilárdságú epoxi ragasztót [36] használtam, olyant, ami alkalmas egyaránt hőre lágyuló műanyag, réz és alumínium ragasztására is: Ez a Loctite Hysol 9514:

Ragasztótípus	1K epoxi-ragasztó
Keverési arány, térfogat (A:B)	Hőre keményedő
Keverési arány, súly (A:B)	Hőre keményedő
Használhatósági idő (fazékidő)	5 perc[®]
Rögzítési idő	30 perc[®]
Szín	Szürke
Viszkozitás	1.480 Pa.s
Nyírószilárdság (GBMS)	46 N/mm²
Lefejtési szilárdság (GBMS)	9,5 N/mm
Üzemi hőmérséklet	-55 °C – 200 °C
Kiszerelés	300 ml kartus 20 kg vödör

1. táblázat: Loctite Hysol 9514 ragasztó adatai

Ragasztott kötésnél: $\tau = \frac{F}{A}$ [4]. Ahol τ a nyíró feszültség, az A nyírt felület és F nyíró erő.

A szervómotor maximális nyomatéka a 2. táblázat alapján: $\mathrm{M_{szervo}} = 2{,}0\mathrm{kgcm} = 0{,}196\ \mathrm{Nm}$ Feltételezve a legrosszabbat a nyíró erő $F = \mathrm{M_{szervo}}/\frac{\mathrm{D_1}}{2} = 19{,}6\mathrm{N}$. A felület pedig $\mathrm{A} = \frac{D^2}{4} = 100mm^2$. Így a nyírófeszültség: $\tau = 0{,}196N/mm^2$, ami sok nagyságrenddel kisebb, mint a megengedett nyírófeszültség ($46N/mm^2$).

Hajtott tárcsákat a tengely menetes részére ragasztjuk, így a nyírt felület hengerpalást lesz: 10mm-es hosszt és 3mm-es átmérőt feltételezve: $\mathrm{A} = D * \pi * H = 96mm^2$. A nyíróerő pedig: $F = \mathrm{M_{szervo}}/\frac{\mathrm{d_1}}{2} = 65{,}3\mathrm{N}$. Így a nyírófeszültség: $\tau = 0{,}695N/mm^2$. Ami szintén sokkal kisebb a megengedettnél.

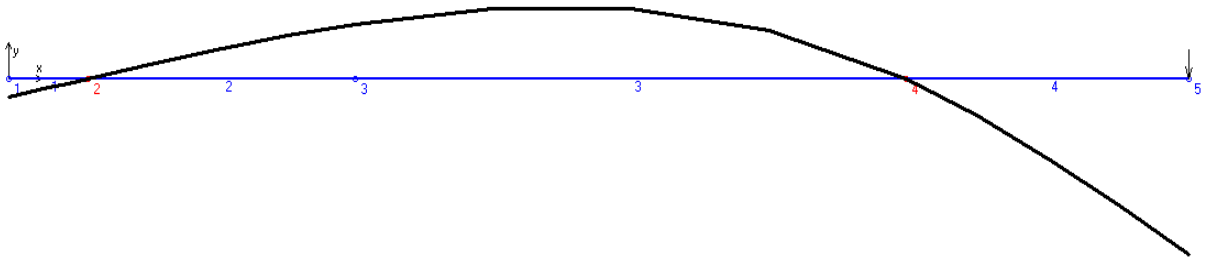
Ebből egyrészt az következik, hogy működni fog a nyomatékátadás, másrészt az, hogy olcsóbb és univerzálisabb ragasztó is megfelel a célnak.

2.1.3.5. Tengelyek ellenőrzése

A felső ház tengelyét felesleges ellenőrizni, a 3mm vastag réz tengely is simán elbírja a pehelykönnyű napelemet. Ha természetbeni körülményekre, azaz időjárásra méreteznénk, akkor számítani kéne a napelem légellenállása miatti a tengelyre ható erőkre, mondjuk a Magyarországon fújó maximális szélerőséget figyelembe véve.

Másrészt az alsó ház tengelyét szintén nem szükséges ellenőrizni nyomásra és húzásra, mert a két házat összekötő 5mm átmérő tengelyszakasz elbírja a 2N-nál kisebb súlyú felső házat.

Viszont van értelme radiális terhelésre ellenőrizni a tengely 1-et, ha feltételezzünk, hogy véletlenül megrántjuk szállítás közben a szerkezetet. Ekkor a tengely 1-re a felső ház súlyának többszöröse hathat. Ezt az állapotot a *Siker2009* program segítségével modelleztem, amely elérhető a *Műszaki mechanika tanszék* honlapjáról [40], vagy megtalálható a DVD mellékletben. A modellezett tengely 3mm átmérőjű, alpakka rúdból van [37], és a felső ház súlyának 10-szerese hat rá a végén. A deformáció a jobb láthatóság kedvéért 40-szeresére van nagyítva.



10. ábra: Tengely 1. lehajlása

A hajlító nyomaték a 4. csomópontnál a legnagyobb: Mh=360Nmm, így a feszültség $\tau = 106N/mm^2$, ami a szakítószilárdság ($R_m = \frac{360N}{mm^2}$) harmada, tehát kibírja a tengely ezt a terhelést is.

2.1.3.6. A napelem és a fototranzisztor rögzítése

A napelemet a tengely 2-re egy közepén átfúrt könnyű, műanyag hasábbal oldottam meg, amit a lyukon keresztül ráhúzhatok a tengelyre. A tengely palástját a közepén kicsit síkra köszörültem, és a hasábba a tengelyre merőlegesen fúrtam egy menetet, amibe csavart csavaroztam, hogy elfordulás ellen védve legyen. A hasábra a napelemet kétoldalú ragasztószalaggal rögzítettem.

A fototranzisztor rögzítésére is egy műanyag hasábot használtam. Egy átmenő, a fototranzisztor átmérőjével azonos furatot ejtettem a hasábon, majd a feléig a tranzisztor pereménél szélesebb zsákfuratot fúrtam. A furatra merőlegesen, oldalt kivágtam egy rést a kábelek kivezetésének, majd ezt is kétoldalú ragasztóval rögzítettem a napelemre.

2.1.3.7. O-gyűrű kiválasztása

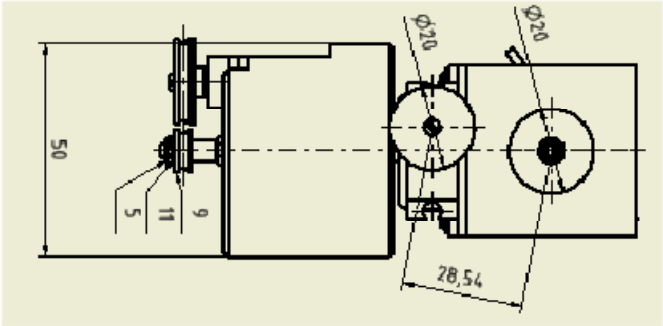
Összeállítási rajz alapján, könnyen meghatározható az elméleti, azaz nyújtott állapotú o-gyűrű mérete, az alábbi képlet alapján:

$$D_{elm} = \frac{\frac{(D_1 + D_2)}{2} \cdot \pi + 2 \cdot \sqrt{\frac{(D_1 - D_2)^2}{4} + L^2}}{\pi} - 2$$

Ahol $D_1; D_2$ a kerékátmérők és L a középpontok közti távolság. A kerekekbe 1mm mély horony van, így az o-gyűrű középátmérőjét kapnánk meg, ha 2 mm vastagot választunk, viszont a szabvány méret a belső ármérő, ezért az eredményből le kell vonnunk 2 mm-t.

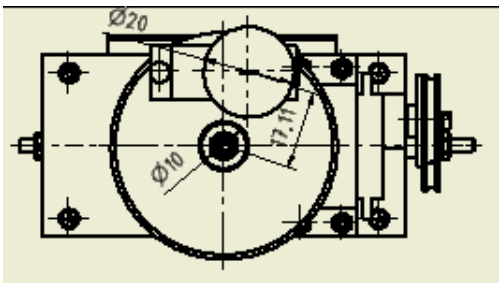
A megfelelő szabványos méretet úgy kell kiválasztani, hogy a megkapott elméleti átmérőnél, maximum 10%-kal kisebb méretet kell választani, mert az o-gyűrű ennyit még tud nyúlni, és így lesz megfelelő a feszítettség a szögelfordulás átadására.

Hajtás 1-nél a két átmérő 20-20 mm, és a köztük lévő távolság 28,54 mm, így az elmélet átmérő 38,18 mm, a választott ő gyűrű mérete: Ø2xØ35.



11. ábra: Hajtás 1.

Hajtás 2.-nél a két átmérő 20 és 10 mm, és a köztük lévő távolság 17,11 mm, így az elmélet átmérő 26,46 mm, a választott o-gyűrű mérete: Ø2xØ24



12. ábra: Hajtás 2

Az o-gyűrűket az O-ring Kft. katalógusából választottam. [29]

Szabvány: A DIN 3760

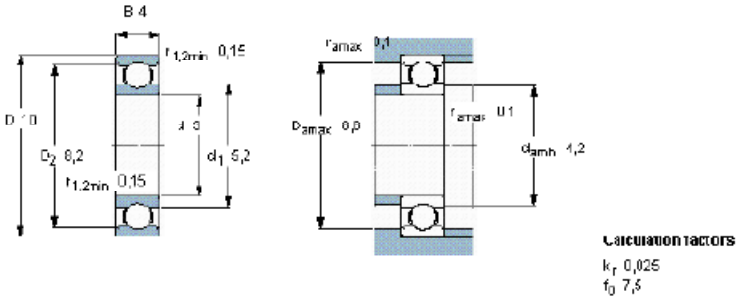
Minőség: 72 NBR 872

2.1.3.8. A golyóscsapágy ellenőrzése

Mivel a csapágyakat nem választottam, hanem azt használtam, ami volt, nem tudtam előre méretezni a csapágyakat a várható terhelésre, sőt ezekre a csapágyakra kellett felépíteni a napraforgó mechanikáját. Így úgy gondolom, hogy érdemes utólag ellenőrizni kész felépítmény alapján, főleg a csapágyház 1-ben lévő csapágyak terheltségét. Számításaim alapját egy a *Miskolci Egyetem* szerverén található dokumentum adta [6].

Deep groove ball bearings, single row, unsealed

Principal dimensions			Basic load ratings		Fatigue load limit	Speed ratings		Mass	Designation
			dynamic	static		Reference speed	Limiting speed		
d	D	B	C	C_0	P_u				
mm			kN		kN	r/min		kg	-
3	10	4	0,54	0,18	0,007	130000	80000	0,0015	623



13. ábra: SKF 623 katalóguslapja [2]

Egyenértékű statikus alapterhelés:

$$P_0 = P = m_{csapagyhaz2} \cdot g = 0{,}16kg \cdot 9{,}81\ N/kg = \ 1{,}57\ N$$

Statikus teherbírás:

$$S_0 = \frac{C_0}{P_0} = \frac{180}{1{,}57} = 115$$

Névleges élettartam (millió körbefordulás):

$$L_{10} = \left(\frac{C}{P}\right)^P = 40689497{,}1$$

Névleges élettartam (években, ahol n=10/h):

$$L_{10_y} = \frac{10^6}{60 \cdot n} L_{10} = 7762800\ év$$

Tehát, a csapágy alig van terhelve, így nem kell tartanunk a csapágy meghibásodása miatt adódó leállásoktól.

2.1.4. Szervomotor

A szervók impulzusszélesség-vezérelt motorok, amelyek adott szöghelyzet felvételére képesek. Ez azt jelenti, hogy a motornak egy impulzusszélességgel lehet megadni, hogy milyen pozícióba álljon be.

14. ábra: Szervómotor [34]

Minden szervóba 3 vezetéket kell bekötni. Ebből kettő a táp és a föld. Tápfeszültség 4,8-7,2V között lehet, a feszültség növelésével növekszik a szervomotor nyomatéka és sebessége, viszont csökken az élettartama. A harmadik kábel a vezérlésért felelős, erre kötjük a vezérlő alapjelét, melynek maximális effektív értéke 5V feszültség lehet.

A vezérlőelektronika első része egy impulzusszélesség-feszültség átalakító. Ez megméri a kapott impulzus szélességét, majd annak alapján egy adott feszültséget generál. A vezérlőbe be van égetve a minimum impulzusszélesség (minIW) és a maximum impulzus szélesség (maxIW), e kettő közti kapott értéket tudja értelmezni. A maximális impulzusszélesség elérésekor a generált feszültség közel azonos a tápfeszültséggel. A motor nem ezzel a feszültséggel lesz meghajtva, ez csak egy referenciafeszültség a szabályzáshoz. A szervó kimenő tengelyére egy potenciométert helyeztek el, melynek három csatlakozása közül egyik a pozitív tápfeszültségre, a másik pedig a földre van kötve. A tengely elfordulásával változik a potméter ellenállása, így a kimenő lábán a szöghelyzettől függő, a tápfeszültség és a föld közötti referenciafeszültség jön létre. A szervomotorban lévő analóg vagy digitális szabályzó az impulzusszélességből generált referencia feszültségből a kimenő tengely aktuális szögpozíciójától függő potenciométerből kijövő feszültséget vonja ki, majd a különbségből állapítja meg a szabályzójelet, mely közvetlenül vezérel egy H-hidas IC-t, ami a motornak szükséges áramot adja.

A vezérléshez szükséges impulzusszélességek szervónként eltérőek. Ami kvázi szabványnak is tekinthető, az, hogy minden szervo 1500µs széles impulzus hatására áll középállásba. Kis mozgásterű szervók, amik -60° és +60° között tudnak mozogni, általában 1200-1800us közötti impulzushosszal mozognak, míg a nagy mozgásterű szervók, melyek -90° és +90° közötti tartományban képesek mozogni, 650us és 2350us közötti impulzusszélességgel irányíthatóak.

Az analóg és digitális szervómotor között az a különbség, hogy analóg szervónál, az impulzust nem elég egyszer kiadni, mert így a motor csak egy nagyon rövid ideig kapna

tápfeszültséget, és ezen a rövid idő alatt nem képes elérni a kívánt pozíciót. Ha az impulzus konverter nem kap jelet, akkor a motort sem gerjeszti, tehát a rendszer elernyed. Vagyis folyamatosan küldenünk kell a jelet a motornak, amíg el nem éri a kívánt szögpozíciót, vagy amíg a tengelyt terhelő nyomaték fennáll. A digitális szervóknál elég egy impulzus, mert a digitális szabályzás megjegyzi a referencia feszültséget, és addig hajtja a motort, amíg az el nem éri a kívánt pozíciót. Persze az impulzushiányos időben lévő elernyedés ezekre is jellemző, tehát terhelő nyomatéknál egy digitális szervomotornak is folyamatosan kell küldenünk a megfelelő szélességű jelet.

Másik különbség a digitális és az analóg szervók között, hogy az analógoknak van felfutás a nyomatékában és sebességében, a digitálisak esetében viszont a szabályzó, amint változik a jel, teljes nyomatékkal és sebességgel indítja meg a motort.

A bevett gyakorlat az, hogy analóg szervókat 50Hz körüli impulzus szélesség modulált jellel (PWM) vezéreljük, a kitöltöttsége pedig 10-20% között szokott lenni. Onnan érzékelhető, hogy az impulzusmentes időszakban a szervó elernyed, hogyha a szervónak fix impulzusokat adunk és kézzel megpróbáljuk eltekerni a tengelyét, akkor el kezd remegni. A digitálisak többet bírnak, 300Hz körüli PWM jelet is adhatunk neki, ami azért jó, mert így kevesebb lesz a nyomaték nélküli időszak is.[9]

2.1.4.1. Szervomotorok kiválasztása

Lehető legolcsóbb szervót kerestem, aminek viszonylag nagy a nyomatéka, és fém fogaskerekei és golyóscsapágyai vannak, így találtam rá a Szán modell [15] csepeli modellboltban 2700Ft.-értkapható Corona-RC CS-929-MG Analóg motorra, ami harmad annyiba kerül, mint a belvárosi boltokban kapható hasonló teljesítményű motorok. Ebből vettem kettőt. Az adatait az alábbi táblázat tartalmazza:

Operating voltage range 4.8V~6.0V	
Operating temperature range	-20'c to +60'c
Operating speed	0.14sec/60' degree
Stall torque	1.8kg.cm / 24.99 oz. in
Running current	200mA/60? no load
FDead band width	≤3mS
Operating travel	40' /one side pulse traveling 400S
Motor type	cored metal brush
Potentiometer type	2 slider/direct drive
Dimensions	22.5x11.5x24.6mm(0.89x0.45x0.97)"
Weight	12.5g
Ball bearing	MR85
Gear material	metal
Connector wire length	215mm(8.46in)
Connector wire stand counter	24
Connector wire gauge	28AWG

2. táblázat: Corona-RC CS-929-MG (Analóg) [32]

Ha már ott voltam, vettem 600Ft.-al drágábban egy példányt ennek a digitális verziójából, a Corona-RC DS-929-MG-ből is egyet, kipróbálásra. Már az első teszteknél kiderült, hogy hibát követtem el. Az analógok nyomába sem ér a digitálisnak: lassan állnak be a megadott helyzetbe (nagy a szabályozási idő), gyengébbek, túl nagy a túllendülésük, és tényleg kisebb nyomtatékkal indulnak el, ezzel szemben a digitális gyors, pontos (minimális túllendülés, rövid szabályozási idő), erősebb és tényleg lehet nagyobb PWM jellel vezérelni, ami sokkal jobb felbontást tesz lehetővé.

A digitális adatait az alábbi táblázat tartalmazza:

Operating voltage range 4.8V~6.0V	
Operating temperature range	-20'c to +60'c
Operating speed	0.11sec/60' degree
Stall torque	2.0kg.cm / 28 oz. in
Running current	200mA/60? no load
FDead band width	≤3mS
Operating travel	40' /one side pulse traveling 400S
Motor type	cored metal brush
Potentiometer type	2 slider/direct drive
Dimensions	22.5x11.5x24.6mm(0.89x0.45x0.97)"
Weight	12.5g
Ball bearing	MR85
Gear material	metal
Connector wire length	215mm(8.46in)
Connector wire stand counter	24
Connector wire gauge	28AWG

3. táblázat: Corona-RC DS-929-MG (Digitális) [33]

2.1.5. Alkatrészek gyártásának lépései

Alkatrészeimet az OMI gépműhelyében található szerszámok segítségével gyártottam le. A két legfontosabb gép, amit használtam az eszterga és a maró gép. Továbbá használtam még fúrógépet, köszörűt, csavarozót és kézi szerszámokat is, mit pl. kézi fűrész, sorjázó, reszelő, kalapács. Az alkatrészek mögötti zárójelben az első szám jelenti, hogy az összeállítási rajzon, milyen tételszámmal szerepel, a második szám pedig azt, hány darab kell belőle, utána az előgyártmány van megnevezve.

15. ábra: Marógép

16. ábra: Eszterga

Csapágyház1 (1;1; D60 plexi rúd)

1. Levágni egy 45 mm hosszú darabot, hogy legyen rajta ráhagyás.
2. 3 pofás tokmányba befogni.
3. Felhúzni az előlapját oldalazó késsel.
4. Beközpontozni központfúróval.
5. 6-os fúróval teljes hosszon átfúrni, vigyázva arra, hogy a fúrót ne tartsuk benne sokáig, mert a plexi nem hővezető és felmelegedvén megolvadhat.
6. Belsőátmérőző-késsel d10H7-re 6 mm mélyen kiesztergálni, mikrométer segítségével.
7. Belsőleszúró-késsel 0,9 mm mélyen és 1,1 mm hosszon hornyot esztergálni.
8. Az él letöréseket oldalazó-késsel kialakítani.
9. Kivenni a tokmányból, megfordítani és visszarakni.
10. Felhúzni ezt a végét is oldalazó-késsel síkra, és lemenni 40 mm hosszig.
11. 7-től 10-es pontig megcsinálni ezen az oldalon is.

Tengely1 (3:1; D6x80)

1. Befogni patronba.
2. Oldalazó-késsel felhúzni síkra az elejét.

3. Külsőátméretező-késsel felhúzni 3h6-ra az oldalát, 5mm-es hosszon mikrométer segítségével.
4. M3-as magméretét, ami 2,9 mm, szintén külsőátméretező-késsel felhúzni 4 mm mélyen.
5. Oldalazókéssel letörni a magméret elejét, menetmetszéshez.
6. M3-as menetmetszővel 4 mm mélyen menetet vágni.
7. Kivenni a tokmányból.
8. Megfordítani, és megfogni úgy, hogy 10 mm kilógjon belőle.
9. Felhúzni az oldalát síkra úgy, hogy 70 mm legyen a távolság a másik tengelyváltó között.
10. Központfúrni az elejét.
11. Kijjebb húzni a munkadarabot úgy, hogy 55 mm lógjon ki belőle.
12. Központfuraton megtámasztani támasztóékkel.
13. Külsőátmérőző-késsel d3h6-ra 53 mm felhúzni mikrométer segítségével.
14. Majd M3-as magméreten, azaz d2,9-re 18 mm mélyen felhúzni külső átmérőző késsel.
15. Oldalazókéssel letörni a magméret elejét, menetmetszéshez.
16. M3-as menetmetszővel 18 mm mélyig menetet vágni.

Tengely2 (2;1; D6x150)

1. Befogni patronba.
2. Oldalazókéssel felhúzni síkra az elejét.
3. Külsőátméretező-késsel felhúzni 3h6-ra az oldalát, 15mm-es hosszon mikrométer segítségével.
4. M3-as magméretét, ami 2,9 mm szintén külsőátméretező-késsel felhúzni 8 mm mélyen.
5. Oldalazókéssel letörni a magméret elejét, menetmetszéshez.
6. M3-as menetmetszővel 4 mm mélyen menetet vágni.
7. Kivenni a tokmányból.
8. Megfordítani, és megfogni úgy, hogy 10 mm kilógjon belőle.
9. Felhúzni az oldalát síkra úgy, hogy 125 mm legyen a távolság a másik tengelyváltó között.
10. Beközpontozni az elejét központfúróval.
11. Kijjebb húzni a munkadarabot úgy, hogy 30 mm lógjon ki belőle.
12. Központfuraton megtámasztani támasztóékkel.
13. Külsőátmérőző-késsel d3h6-ra 25 mm mélyen felhúzni mikrométer segítségével.
14. Majd M3-as magméreten, azaz d2,9-re 18 mm mélyen felhúzni külső átmérőző késsel.
15. Oldalazó-késsel letörni a magméret elejét, menetmetszéshez.
16. M3-as menetmetszővel 18 mm mélyig menetet vágni.

Hajtó szíjkerék (9;2; D20 rúd)

1. Befogni 3 pofás tokmányba.
2. Felhúzni az elejét síkra oldalazó késsel.
3. Központ fúrni.
4. 3,2-es fúróval közepén átfúrni kb. 6 mm mélyen.
5. R1-es rádiuszkéssel az elejétől 2 mm távolságra 1 mm mélyre körhornyot esztergálni.
6. 1,9-es leszúró késsel az elejétől 5,9 mm-re teljes átmérőn leszúrni.

Hajtott szíjkerék 1 (10;1; D10 rúd)

1. Befogni tokmányba.
2. Felhúzni síkra az elejét oldalazó késsel.
3. Központfúrni.
4. 16 mm mélyen d3,2-es fúróval kifúrni.
5. Külsőátmérőző-késsel 9,5 mm hosszon d5-re esztergálni.
6. R1-es rádiuszkéssel a válltól 2 mm távolságra 1 mm mélyre körhornyot esztergálni.
7. 1,9-es leszúró késsel a válltól 5,9 mm-re teljes átmérőn leszúrni.

Hajtott szíjkerék 2 (5;1; D20 rúd)

1. Befogni tokmányba.
2. Felhúzni síkra az elejét oldalazó késsel.
3. Központfúrni.
4. 10 mm mélyen d3,2-es fúróval kifúrni.
5. Külsőátmérőző-késsel 3,7 mm hosszon d5-re esztergálni.
6. R1-es rádiuszkéssel a válltól 2 mm távolságra 1 mm mélyre körhornyot esztergálni.
7. 1,9-es leszúró késsel a válltól 5,9 mm-re teljes átmérőn leszúrni.

Tengelykapcsoló példányszám (x;2; szervóhoz kapott tengelykapcsoló)

1. Felső lapját 1 mm vastagra lecsiszolni dörzspapírral.

Oldallap (2;2; # 12 lap)

1. Levágni 2 db 45 x 45-ös négyzetet.
2. Befogni marógépnek a satujába mindkettőt egyszerre, lapjával szembe.
3. Ütőkéssel feltisztítani mindkettőnek az elejét.
4. Megjelölni a szemben lévő sarkokat.
5. Kivenni és az óra járásával ellentétes irányába megfordítani, mert a marógép arra lejt.
6. Ütőkéssel felhúzni az oldalukat úgy, hogy 40 mm magasak legyenek.
7. Ki kell fogni.
8. Egy derékszögű idomhoz a lemart oldalt odatámasztani és így befogni a satuba, majd a derékszöget eltávolítani.
9. Feltisztítani az oldalát ütőkéssel.
10. 8-9-es pontig ugyanezt megismételni a másik munkadarabbal.

11. Megjelölt sarkuknál fogva összefogni és beszorítani a satuba úgy, hogy derékszögű sarkuk legyen lent.
12. Ütőkéssel felhúzni a másik oldalukat is 40 mm-re.
13. Befogni az egyiket 4 pofás tokmánnyal az eszterga gépbe.
14. Központfúrni.
15. 7-es fúróval teljes hosszon, középen átfúrni, majd belső átmérőző késsel d10H7-et felhúzni 10 mm mélyen mikrométer segítségével.
16. Belső lyukat letörni oldalazó késsel.
17. A 12-16-ig megismételni a másikkal is.

Tartó (6;1; 10 lap)

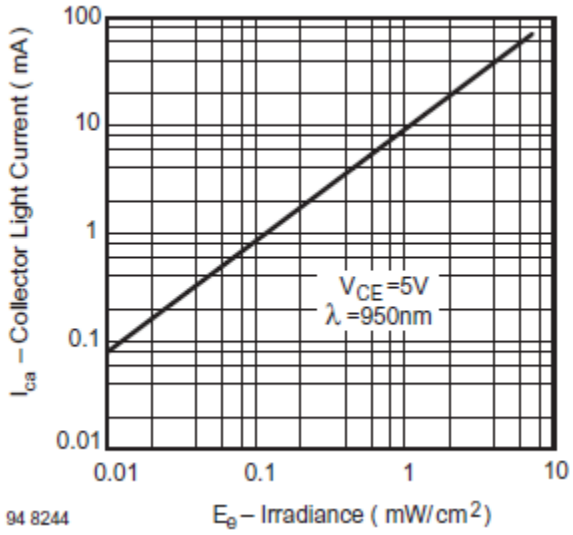
1. Lefűrészelni egy 10 x 20-as darabot.
2. Átfúrni a lapjára merőlegesen 6-os fúróval.
3. Majd erre merőlegesen az oldalán befúrni M3-as magméreten, d2,5-s fúróval a lyukig befúrni.

17. ábra: Munka közben

2.2. Elektronika

2.2.1. Fényérzékelő szenzor kiválasztása

Lehető legolcsóbb megoldást keresvén a fototranzisztorok közt keresgéltem magamnak fényérzékelő szenzort, így akadtam rá egy lineáris infravörösre működő példányra, a BPV11F-re, a *Lomex* [11] honlapján. [17]



18. ábra: BPV11F megvilágítás-kollektor áram diagram [17]

Mivel a Nap infravörös tartományban is sugároz, a feladatra tökéletesen megfelel, csak a tesztelést teszi nehézzé, mert nem lehet fénycsővel vagy LED-es lámpával kipróbálni. A diagramon látható, hogy ha a kollektor-emitter feszültség 5V, amivel alapvetően a MCU-t is meg fogom hajtani, akkor ezen lineáris megvilágítás-kollektor áram összefüggést kapunk, ami igazából a végső feladat megvalósítása szempontjából teljesen felesleges, mert nem akarok vele szabályozni, csak a maximális érték helye lesz a fontos, így nem is kell linearitás, csak szigorúan monoton összefüggés, hogy csak egy helyen vegye fel a maximumát.

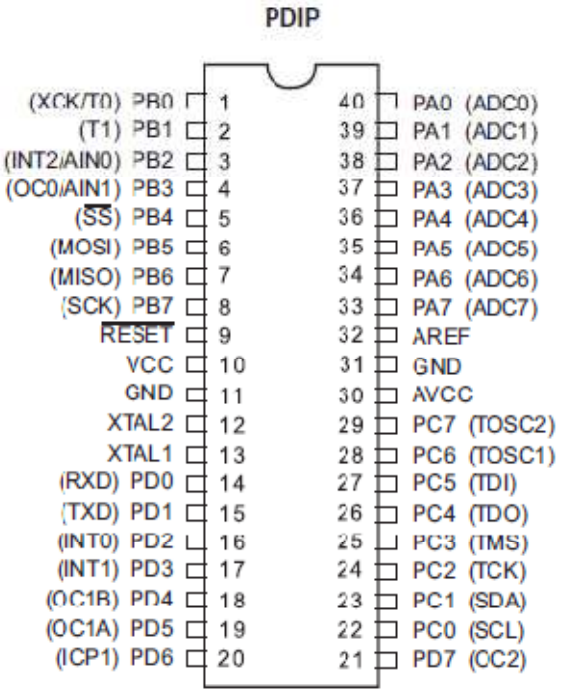
2.2.2. Áramkör és nyáktervező

Sokféle program ingyenes verzióját töltöttem le, de egyiket kényelmetlenebb volt használni, mint a másikat, pl. *Diptrace, Utilboard, PCB layout, PCB artist, Eagle, Orcad*. Még a *Diptrace* tűnt a legkezelhetőbbnek, viszont az csak nyáktervező program. Emlékeztem rá *Analóg elektronika (BMEVIAUA009)* egyik utolsó laborján tanultunk egy programról, amivel viszonylag egyszerűen el lehetett jutni a kapcsolási rajzon keresztül a kész nyáktervig. Ez pedig nem volt más, mint a *KiCAD*. Miután nem emlékeztem a kezelésére, ezért egy konzultáció keretében Dr. Sütő Zoltán tanár úr megtanított a kezelés alapfogásaira.

A *KiCAD* programot az *Automatizálási és Alkalmazott Informatikai Tanszék* honlapjáról lehet letölteni, vagy megtalálható a DVD mellékletben.[28]

2.2.3. A mikrokontroller kiválasztása

Mikrokontroller kiválasztásánál nem volt nehéz dolgom, az Atmel [10] és a Microchip mikrokontrollerjei jöhettek szóba, mert ezekhez értek valamennyire és ezekről található a legtöbb anyag az interneten. Atmel ATmega16 DIP tokozású MCU-ját választottam, mert könnyű kezelni, sokféle funkció van beépítve, könnyen építhető és fejleszthető, közepes mennyiségű lába van, a melyeket még épp lehet kezelni egy próbapanelen, az ára is kedvező, a hozzá járó fejlesztő környezetet kényelmes és egyszerű használni és abban is sok funkció van pl. szimulátor, ráadásul ehhez tudtam könnyen ISP programozót építeni.



19. ábra: AVR Atmega16 PDIP pin kiosztása

Főbb funkciói és paraméterei [20]:

- Maximum 16MHz órajel és 16MIPS (millió utasítás per szekundum)
- 40 láb, ebből 32 programozható
- 16KB Flash memória
- RISC architektúra: 131 utasítással
- Analóg komparátor és 8 láb analóg bemenet multiplexelt 10 bites A/D átalakítóval
- 4 számláló, 4 PWM
- USART, SPI, JTAG

2.2.4. Programozó

Hobbielektroni.hu-n található cikk [1] alapján építettem meg az programozómat, ami egy STK-500-as klónja. A kapcsolási rajz, a nyákterv és az élesztést segítő driverek a DVD melléletben megtalálhatóak. Ez egy ISP (In system programable) eszköz, ami azt jelenti, hogy soros kommunikációval, viszonylag kevés lábat használva lehet programozni a mikrokontrollert, akár beültetés után is.

20. ábra: AVR-Doper, USB-s ISP programozó

Szükséges lábak a költekezők:

- MOSI (Master Output Slave Input)
- MISO (Master Input Slave Output)
- SCK (Serial Clock)
- RESET
- VCC
- GND

A MOSI és MISO lábakon folyik a soros kommunikáció, az SCK adja a szükséges órajelet, a GND és VCC a tápellátást, RESET pedig azért kell, mert az AVR-eket folyamatos reset alatt lehet programozni.

2.2.4.1. Fejlesztői környezet

A mikrokontrollerhez ingyenesen használható az *AVR Studio 4* fejlesztői környezet, ami egy nagyon kényelmes és okos program, melyben C-ben vagy Assemblyben lehet programozni. Miután ember közelibb nyelv, és a pontos működés és időzítés nem fontos, így én az előbbit választottam. Az *AVR Studioban* lehet a csatlakoztatott fejlesztőkártyával az MCU-t felprogramozni és debuggolni is - ha a kártya képes arra -, de lehetőség van arra is, hogy a beírt kód alapján a kontroller működését szimuláljuk.

Az *AVR STudio 4* elérhető innen ingyenesen [19], vagy megtalálható a DVD mellékleten.

3. Szoftver

3.1. Szoftver elemek

Napraforgót működtető szoftver ismertetése először tekintsük át az AVR Atmega16 programozásának alapjait, és programom két fontos elemének, a PWM-nek és az ADC-nek a beállítását.

3.1.1. Programozási alapok

Az Assembly mellett C jellegű nyelven lehet programozni a mikrokontrollereket, amivel a főbb funkciókat különböző regiszterekben lévő bitek 1-esbe vagy 0-ba állításával lehet elérni. Fontos, hogy törekedjünk az egyszerűségre és a bonyolultabb függvények kikerülésére.

A táblázatok és képek nagy része az Atmega16 katalógusából [20] származik.

Fontos megjegyezni, hogy a következő oldalak nélkül nehezen lenne értelmezhető a programozási rész, így le kellet írnom, mindent, ami ezen funkciók programozásához szükséges.

3.1.1.1. Alapbeállítások:

Két könyvtárat kell a feladathoz, a programhoz mindenképp mellékelni. Az egyik az I/O funkciókért felelős, a másik a várakozó szubrutinokat tartalmazó könyvtár. A MCU típusának megfelelő könyvtárat nem kell mellékelni, mert amikor új projektet csinálunk az AVR Studioban, ki kell választanunk, milyen mikrokontrollerünk van, és a program alapból mellékeli a megfelelő header fájlt. Másrészt érdemes definiálni, hogy az MCU hány MHz-en jár, bár ezt a felprogramozásnál is meg lehet tenni az alapbeállításokat konfiguráló ablaknál az AVR Studio 4-ben.

```
#include <avr/io.h>
#include <util/delay.h>

#ifndef F_CPU
#define F_CPU 800000UL //8MHz orajel
#endif
```

3.1.1.2. Bitműveletek:

Minden speciális regiszternél minden bitnek külön neve van, hogy, amit a preprocesszor megfelelő számra cserél le a fordítás során. Ennek két előnye van: [24]

1. Könnyen el tudjuk érni a funkciókat, és a kódot is könnyebben értelmezzük: DDRD=0x44; vagy DDRD=68; nem biztos, hogy első látásra értelmezhető, viszont arról, hogy DDRD|=(1<<PD3)|(1<<PD3); már látszik, hogy a 7. és 3. lábról van szó.
2. Könnyebb szállíthatóság: ha a kódunkat más típusú mikrokontrolleren akarjuk használni és számokat írtunk bele a regiszterbe, akkor át kell nézni és át kell írni a regiszterek tartalmát, mert egyáltalán nem biztos, hogy ugyanazt a funkciót ugyanazzal a bittel tudjuk elérni. Viszont a preprocesszor a megfelelő header fájlt használva át tudja írni a szövegek helyére a megfelelő számokat.

Ha például a D port 4. lábát akarjuk kimenetté tenni, akkor a DDRD regiszterben kell a 3 bitet egyesbe billenteni, mert 2^3=4, ezért kell csinálni egy bájtot, amiben egy egyest eltolok balra PD4-gyel (ami helyett a preprocesszor behelyettesíti a 3-at), majd azt vagyolni kell a DDRD eddigi tartalmával, hogy az előző beállítások ne vesszenek el. Röviden, C-ben leírva:

```
DDRD|=(1<<PD4);
```

Ha pedig törölni akarunk a bitet, akkor először csinálnunk kell egy maszkot, azaz egy olyan bájtot, amiben minden egyes bit 1-es értékű, kivéve a kiválasztott bitet, ami nulla. Ezt úgy csináljuk, hogy fogom a bájtot, és egy egyest eltolok balra PD4-gyel (ami helyett a preprocesszor behelyettesíti a 3-at), az egészet negálom, tehát ahol egyes volt ott nulla lesz a többinél pedig egyes. Ha ezt éselem az eredeti regiszter tartalmával, akkor minden bit átíródik, kivétel a PD4 es, mert az törlődik. Röviden, C-ben leírva:

```
DDRD&=~(1<<PD4);
```

3.1.1.3. Típusok:

Az Atmega16 regiszterei 8 bitesek, tehát ha 255-nél nagyobb számot akarunk tárolni, akkor ahhoz több regiszter kell Assembly-ben, de a C-ben a preprocesszor megoldja a konverziót és az allokációt, ha a megfelelő típust választottuk. Egyik alapkönyvtárában a <stdint.h> -ben, vannak a meghatározott hosszú egész típusok definíciói: [27]

```
typedef signed char int8_t
```

A táblázat tartalmazza a típusok neveit és méreteit:

Név	Típus	Max. érték	Min. érték
signed char	int8_t	-128	127
unsigned char	uint8_t	0	255
signed int	int16_t	-32768	32767
unsigned int	uint16_t	0	65535
signed long int	int32_t	-2147483648	2147483647
unsigned long int	uint32_t	0	4294967296
signed long long int	int64_t	-9,2e+18	9,2e+18
unsigned long long int	uint64_t	0	1,8e+19

4. táblázat: Meghatározott szélességű egész típusok x]

3.1.1.4. Ciklusok:

Ugyanúgy lehet használni, mindegyik ciklust, de a főfüggvénynek érdemes örökké futnia, ekkor használjuk ezt a megadási módszert a while vagy for ciklussal:

```
while(1)
   {}
 for(;;)
   {}
```

3.1.1.5. Várakozás

Rövid várakozásokra vannak beépített függvények a <util/delay.h> könyvtárban, de hosszú várakozásra ezeket nem lehet használni, mert csak lefagyasztják az MCU-t. Ezért kell függvényt írni rá.

```
void wait_ms(uint16_t ms)
{
     while ( ms )
     {
          _delay_ms(1);
          ms--;
     }
}
```

A fönti függvény annyi ideig futtatja le azt a ciklust, amiben egy ms-os várás van, amekkora számot írunk bele az argumentumába.

3.1.1.6. Projekt beállítások

Új projekt létrehozásához a Project menü New project gombját kell megnyomnunk, ott típusnak az AVR GCC-t kell választanunk, ha C-ben akarunk programozni, majd meg kell adnunk a projekt nevét, és egy mappát, ahol a fájlokat tároljuk. Majd debug platformot kell választanunk, ami AVR Simulator 1 vagy 2 lehet, ha nincs debuggerünk, és meg kell adnunk a mikrovezérlő típusát, ami nálam ATmega16 volt. Utána a Project menü Configuration options gombját megnyomva hozzuk elő a projekt beállításokat. Itt kell beállítani az órajelet, amivel a szimulátor működni fog. Ha megírtuk a programot, akkor a Build Active Configuration gombbal lehet lefordítani, ez létrehozza a bináris .hex fájlt a projekt mappában. Ha szimulálni akarunk, akkor Build and Run gombot kell megnyomnunk. Ha fel akarjuk programozni a mikrokontrollert, akkor a Con gombra kattintva tudunk csatlakozni a programozóhoz: feljön egy ablak, ahol ki kell választani a programozó típusát – nálam ez STK-500 -, és a portot, amin csatlakozik a számítógéphez, ezt a mezőt érdemes Auto-n hagyni, ha csak egy ilyen eszköz van csatlakoztatva. Ha ezt megettük, akkor feljön az STK-500 kezelőablaka. Itt először a Main fülön a Read Signature gombbal kell ellenőrzi, hogy kapcsolódik-e a MCU-hoz, ha igen, akkor a Fuses fülön a SUT_CKSEL legördülő menüben, kell kiválasztani az órajelforrás típusát, nálam az belső 8MHz-es, 4ms-os indulási idővel. Ez a beállítás elmentődik a projektben, így nem kell minden programozáskor beállítani. Majd a Program fülön, a Flash memóriába kell felprogramozni a kiválasztott .hex fájl a Program gombbal.

Ha programozóhoz nem tudunk csatlakozni, akkor próbáljuk meg kihúzni és újra bedugni az eszközt, ha ez nem működik, akkor nézzük, meg a jumbereket a programozón: milyen módon használjuk, milyen tápról, ha így sem megy, akkor nagy valószínűséggel elromlott a programozó. Ha a MCU jelét nem tudjuk olvasni, vagy nem tudjuk programozni, akkor nézzük meg, hogy jól van-e bedugva, kap-e tápot a mikrovezérlő. Fontos megemlíteni, hogy ha gyenge áramforrással, pl. elemmel próbálunk egy nagy áramfelvételű kapcsolás táplálni – két szervomotor meghajtása már az -, akkor nem fog működni a programozás.

3.1.2. I/O konfigurálása

Az Atmega16-ban 3 regiszter felelős az I/O beállításokért, a DDRx, a PORTx és a PINx. Ahol az x-ek helyére az egyes kimeneti bájtoknak a betűjelét kell behelyesíteni, ami nálunk lehet A,B,C és D.

A DDRx (Data Direction Register) felelős a lábak ki- vagy be-meneti irányultságának meghatározására. Ha az egyik láb DDRx bitje 0, akkor a láb kimenetként fog működni, ha 1 akkor viszont bemeneteként.

A PORTx regiszter felelős a láb 3 állapotának (tri-state) beállításáért. Ha a láb DDRx bitje 0, azaz inputra van beállítva, és PORTx regiszterét egyesbe billentjük, akkor a láb nagy impedanciás bemenetként fog működni; ha nullán hagyjuk, akkor sima bemenet lesz. Viszont, ha DDRx bitje , és a PORTx regisztere is 1, akkor magas jelszintű kimenet lesz, értelemszerűen, ha a PORTx 0, akkor alacsonyszintű kimenet lesz.

A PINx regiszterrel lehet kiolvasni, az egyes lábak bemeneti értékeit, ha DDRx-el inputba, és PORTx-et 0-ba állítottuk.

Az alábbi táblázat foglalja magába az előbb leírtakat, a megértéséhez szükséges még, hogy a SFIOR (Special Function I/O Register) PUD bitjének 1-esbe billentésével lehet kikapcsolni az összes bemeneti ellenállást.

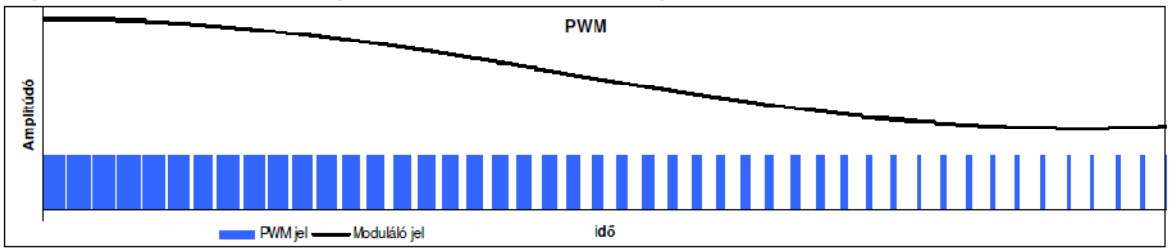
Table 20. Port Pin Configurations

DDxn	PORTxn	PUD (in SFIOR)	I/O	Pull-up	Comment
0	0	X	Input	No	Tri-state (Hi-Z)
0	1	0	Input	Yes	Pxn will source current if ext. pulled low.
0	1	1	Input	No	Tri-state (Hi-Z)
1	0	X	Output	No	Output Low (Sink)
1	1	X	Output	No	Output High (Source)

5. táblázat: I/O lábak beállításai

3.1.3. Impulzusszélesség moduláció

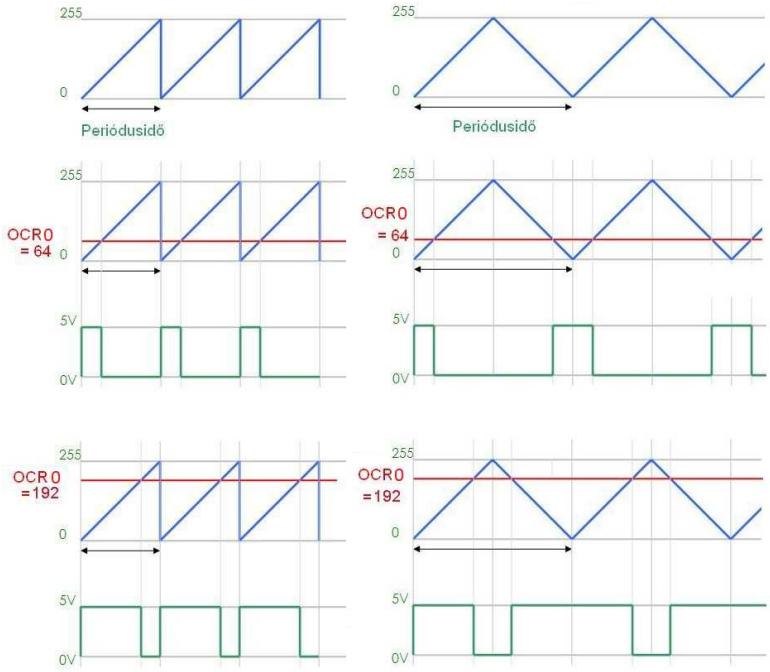
Az impulzusszélesség-modulált jel (PWM) olyan digitális, azaz állandó amplitúdójú jel, mely adott frekvencián változó kitöltöttséggel (pulzushosszal) rendelkezik, tehát azt tudjuk változtatni, hogy egy periódusidő alatt mennyi ideig legyen magas szinten a jel.



21. ábra: Impulzus szélesség moduláció [38]

Létezik még impulzusamplitúdó-moduláció (PAM), impulzusfrekvencia-moduláció (PFM), és impulzuskésleltetés-moduláció (PDM). Az utóbbinál az amplitúdó mellett a pulzus szélesség is állandó, itt a pulzusok távolságát tudjuk változtatni. [38] A mikro vezérlőnk hardveresen csak PWM-et tud előállítani, de ez nekünk elég is, mert ez kell a szervómotor vezérléshez. Megfigyelhetjük, hogy a PWM jel kitöltöttségével arányos analóg jelet kaphatunk, ha integrátorkapcsolást kötünk eléje, így PWM-mel lehet teljesítményt is vezérelni, pl. lámpák fényerejét, sima DC motorok sebességét is lehet vele szabályozni, amelyeknél nem is kell integrátor, mert a tekercselés impedanciája integráló hatású - persze érdemes nagyobb frekvenciát alkalmazni.

Az Atmega16-ban 4 PWM jelet tudunk hardveresen előállítani a benne lévő 3 számláló segítségével. Ez azt jelenti, hogy a mikrovezérlő folyamatosan tudja kiadni ezt a jelet, anélkül, hogy szoftveresen figyelnénk rá, mert a számlálók a CPU-tól függetlenül működnek. A Timer0 és Timer2 egy-egy 8bites, a Timer1-nek viszont van két 10bites PWM generátora. A PWM jelet mind a négy esetben úgy állítja elő, hogy amikor elindítja a számlálót, bebillenti a kimenetet egyesbe vagy nullásba (attól függően, hogy invertált vagy nem invertált jelet akarunk kapni). A számláló értéke folyamatosan összehasonlítódik az OCRx-ben (Output Compare Register) lévő számmal; ha egyezik az érték, akkor negálja a kimenetet, majd elszámol a maximális értékig. Itt válik ketté a két PWM típus, amit a mikrokontroller a számlálójával tud csinálni, mert vagy nullára ugrik, és újrakezdi; vagy elkezd lefele számolni, és amikor megint az OCRx értékhez ér, akkor megint negálja a kimenetet. Az előbbi, ami a farkasfog mintát használja, a gyors; az utóbbi, ami a háromszög alakú mintát, a fázishelyes PWM. Észrevehető, hogy fázishelyes esetben a jel frekvenciája éppen fele akkora, mint a gyors PWM mód esetén.[23][26][39]



22. ábra: Timer0 gyors és fázis helyes PWM-e [39]

3.1.3.1. Timer0

A Timer0 PWM-je nem alkalmas a feladatra, de azért tekintsük át a beállítását és használatát a könnyebb megértés érdekében. A jelet a OC0 lábra küldi ki majd az MCU, ami a B port 3-as lába.

A számlálóhoz tartozik egy előosztó. Az előosztás mértékének megadásával tudunk különböző bemenő órajelet előállítani CPU órajeléből, ezzel tudjuk állítani a számlálás sebességét. Az előosztók lehetnek: 1, 8, 64, 256, 1024.

Így a PWM frekvenciája:

$$f_{pwm} = \frac{f_{cpu}}{N \cdot M}$$

Ahol N az előosztó értéke, és M 256, ha gyors PWM-et akarunk; és 510, ha fázishelyeset.

A Timer0-at a TCCR0 regiszter segítségével állíthatjuk be:

Bit	7	6	5	4	3	2	1	0	
	FOC0	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00	TCCR0
Read/Write	W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

23. ábra: Timer/Counter Control Regiszter-TCCR0

A WGM0x (Waveform Generation Mode) bitek felelősek a jelgenerálási módozatok beállításért:

Table 38. Waveform Generation Mode Bit Description

Mode	WGM01 (CTC0)	WGM00 (PWM0)	Timer/Counter Mode of Operation	TOP	Update of OCR0	TOV0 Flag Set-on
0	0	0	Normal	0xFF	Immediate	MAX
1	0	1	PWM, Phase Correct	0xFF	TOP	BOTTOM
2	1	0	CTC	OCR0	Immediate	MAX
3	1	1	Fast PWM	0xFF	BOTTOM	MAX

6. táblázat: WGM0x bitek beállításai

A COM0x (Compare Match Output Mode) bit felelős az OC0 kimeneti láb viselkedésért:

Table 39. Compare Output Mode, non-PWM Mode

COM01	COM00	Description
0	0	Normal port operation, OC0 disconnected.
0	1	Toggle OC0 on compare match
1	0	Clear OC0 on compare match
1	1	Set OC0 on compare match

Table 40. Compare Output Mode, Fast PWM Mode

COM01	COM00	Description
0	0	Normal port operation, OC0 disconnected.
0	1	Reserved
1	0	Clear OC0 on compare match, set OC0 at BOTTOM, (non-inverting mode)
1	1	Set OC0 on compare match, clear OC0 at BOTTOM, (inverting mode)

Table 41. Compare Output Mode, Phase Correct PWM Mode

COM01	COM00	Description
0	0	Normal port operation, OC0 disconnected.
0	1	Reserved
1	0	Clear OC0 on compare match when up-counting. Set OC0 on compare match when downcounting.
1	1	Set OC0 on compare match when up-counting. Clear OC0 on compare match when downcounting.

7. táblázat: COM0x bitek beállításai

A CS0x (Clock Select) bitek pedig az elő osztó meghatározására szolgálnak:

Table 42. Clock Select Bit Description

CS02	CS01	CS00	Description
0	0	0	No clock source (Timer/Counter stopped).
0	0	1	$clk_{I/O}$/(No prescaling)
0	1	0	$clk_{I/O}$/8 (From prescaler)
0	1	1	$clk_{I/O}$/64 (From prescaler)
1	0	0	$clk_{I/O}$/256 (From prescaler)
1	0	1	$clk_{I/O}$/1024 (From prescaler)
1	1	0	External clock source on T0 pin. Clock on falling edge.
1	1	1	External clock source on T0 pin. Clock on rising edge.

8. táblázat: CS0x bitek beállításai

Tehát, ha 8Mhz-es órajelen 1 MHz-es, gyors, nem invertált PWM-et szeretnék, melynek kitöltöttsége 50%, akkor ezt kell beleírnom a programba:

```
TCCR0|=(1<<WGM00)|(1<<WGM01)|(1<<COM01)|(1<<CS00);
DDRB|=(1<<PB3); //Timer0 lab kimenetnek
OCR0=128; //50%-os kitoltottség
```

3.1.3.2. Timer1

A Timer1 egy 16 bites számláló, melynek sok alkalmazása lehetséges. Én csak a PWM jel előállításához szükséges beállításokat ismertetem. A Timer0-nak előosztója van, ami miatt csak a CPU frekvenciájának adott hányada lehet a PWM frekvenciája. Viszont a Timer1-nél be tudunk allítani egy TOP értéket, amivel megadhatjuk, hogy a számláló meddig számláljon el (a Timer0 mindig elszámolt 255-ig), amivel pontos frekvenciákat tudunk beállítani. A számláló maximális értéke 2^16-1, tehát ennél kisebb számot lehet megadni. Ezt a TOP értéket, mivel 16bites, az ICR1H és ICR1L (Input Capture Register 1 Low/High) regiszterekben lehet definiálni, ezeket össze lehet foglalni C-ben a ICR1 regiszterbe, mert helyettünk a fordító megoldja a tárolást.

A beállított frekvencia érték lesz mindkét Timer1 által generált PWM frekvenciája, a kitöltöttségüket két különálló regiszterkettős (mert 16 bites a komparátor és 10bites a PWM felosztása) határozza meg: OCR1AH és OCR1AL; és OCR1BH és OCR1BL (Output Compare Register). Ezeket C-ben össze lehet foglalni egy-egy regiszterbe: OCR1A és OCR1B.

3.1.3.3. Kiszámítás

A Timer1 PWM frekvenciájának kiszámítása az alábbi képlettel lehetséges:

$$f_{pwm} = \frac{f_{cpu}}{K_1 \cdot N \cdot (K_2 + TOP)}$$

Ahol N az előosztó értéke, ami 1, 8, 64, 256, 1024 lehet. A TOP a számláló maximális értéke, ami egy 16 bites szám. A K1 és K2 értéke a PWM típusától függ: 2 és 0, ha fázishelyes; és 1 és 1, ha gyors PWM-et akarunk.

Ha a mikrovezérlő 8MHz-en jár és az előosztót 64-nek választottuk, és 50Hz-es gyors PWM jelet akarunk kapni:

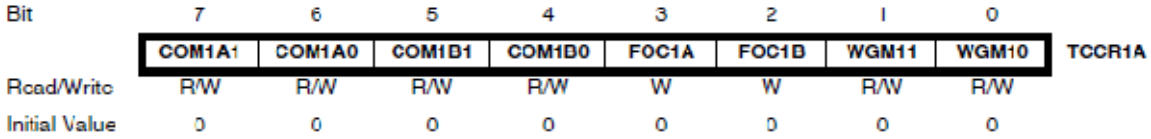
$$50Hz = \frac{8MHz}{1 \cdot 64 \cdot (1 + TOP)}$$

Ebből következik, hogy a TOP érték:

$$TOP = 2499$$

3.1.3.4. Beállítás

A Timer1 beállításért 2 regiszter is felelős: TCCR1A és TCCR1B (timer/Counter Control Register):



Bit	7	6	5	4	3	2	1	0	
	COM1A1	COM1A0	COM1B1	COM1B0	FOC1A	FOC1B	WGM11	WGM10	TCCR1A
Read/Write	R/W	R/W	R/W	R/W	W	W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

24. ábra: TCCR1A regiszter

A COM1Ax és COM1B (Compare Output Mode) regiszterek a Timer1 kimenő lábai (OC1A és OC1B) viselkedésének meghatározásáért felelősek. A mi szempontunkból fontos, hogy itt lehet beállítani, hogy invertált legyen, avagy sem a PWM jelünk.

Table 45. Compare Output Mode, Fast PWM

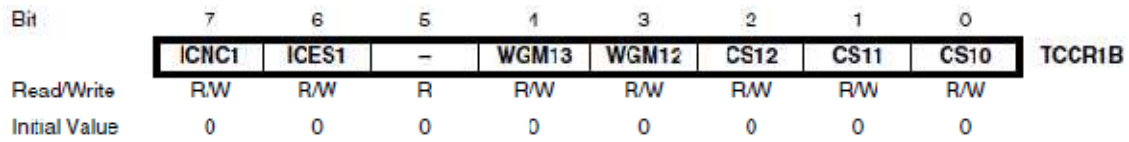
COM1A1/COM1B1	COM1A0/COM1B0	Description
0	0	Normal port operation, OC1A/OC1B disconnected.
0	1	WGM13:0 = 15: Toggle OC1A on Compare Match, OC1B disconnected (normal port operation). For all other WGM13:0 settings, normal port operation, OCnA/OCnB disconnected.
1	0	Clear OC1A/OC1B on compare match, set OC1A/OC1B at BOTTOM, (non-inverting mode)
1	1	Set OC1A/OC1B on compare match, clear OC1A/OC1B at BOTTOM, (inverting mode)

Table 46. Compare Output Mode, Phase Correct and Phase and Frequency Correct PWM

COM1A1/COM1B1	COM1A0/COM1B0	Description
0	0	Normal port operation, OC1A/OC1B disconnected.
0	1	WGM13:0 = 9 or 14: Toggle OCnA on Compare Match, OCnB disconnected (normal port operation). For all other WGM13:0 settings, normal port operation, OC1A/OC1B disconnected.
1	0	Clear OC1A/OC1B on compare match when up-counting. Set OC1A/OC1B on compare match when downcounting.
1	1	Set OC1A/OC1B on compare match when up-counting. Clear OC1A/OC1B on compare match when downcounting.

9. táblázat: COM1nx bitek beállításai

FOC1n (Force Output Compare) bitek csak, akkor aktívak, ha nem PWM módban használjuk a Timer1-et, ezért ezeket nem részletezem.



Bit	7	6	5	4	3	2	1	0	
	ICNC1	ICES1	–	WGM13	WGM12	CS12	CS11	CS10	TCCR1B
Read/Write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

25. ábra: TCCR1B regiszter

A 4 db WGM1x (Waveform Generation Mode) bit a két regiszterben van szétosztva. Ezek a bitek felelősek a jelgenerálási módozatok beállításáért, tehát itt tudjuk beállítani a PWM jel típusát:

Table 47. Waveform Generation Mode Bit Description(1)

Mode	WGM13	WGM12 (CTC1)	WGM11 (PWM11)	WGM10 (PWM10)	Timer/Counter Mode of Operation	TOP	Update of OCR1x	TOV1 Flag Set on
0	0	0	0	0	Normal	0xFFFF	Immediate	MAX
1	0	0	0	1	PWM, Phase Correct, 8-bit	0x00FF	TOP	BOTTOM
2	0	0	1	0	PWM, Phase Correct, 9-bit	0x01FF	TOP	BOTTOM
3	0	0	1	1	PWM, Phase Correct, 10-bit	0x03FF	TOP	BOTTOM
4	0	1	0	0	CTC	OCR1A	Immediate	MAX
5	0	1	0	1	Fast PWM, 8-bit	0x00FF	BOTTOM	TOP
6	0	1	1	0	Fast PWM, 9-bit	0x01FF	BOTTOM	TOP
7	0	1	1	1	Fast PWM, 10-bit	0x03FF	BOTTOM	TOP
8	1	0	0	0	PWM, Phase and Frequency Correct	ICR1	BOTTOM	BOTTOM
9	1	0	0	1	PWM, Phase and Frequency Correct	OCR1A	BOTTOM	BOTTOM
10	1	0	1	0	PWM, Phase Correct	ICR1	TOP	BOTTOM
11	1	0	1	1	PWM, Phase Correct	OCR1A	TOP	BOTTOM
12	1	1	0	0	CTC	ICR1	Immediate	MAX
13	1	1	0	1	Reserved	–	–	–
14	1	1	1	0	Fast PWM	ICR1	BOTTOM	TOP
15	1	1	1	1	Fast PWM	OCR1A	BOTTOM	TOP

10. táblázat: WGM1x bitek beállításai

CS1x bitekkel lehet beállítani az előosztót:

Table 48. Clock Select Bit Description

CS12	CS11	CS10	Description
0	0	0	No clock source (Timer/Counter stopped).
0	0	1	$clk_{I/O}$/1 (No prescaling)
0	1	0	$clk_{I/O}$/8 (From prescaler)
0	1	1	$clk_{I/O}$/64 (From prescaler)
1	0	0	$clk_{I/O}$/256 (From prescaler)
1	0	1	$clk_{I/O}$/1024 (From prescaler)
1	1	0	External clock source on T1 pin. Clock on falling edge.
1	1	1	External clock source on T1 pin. Clock on rising edge.

11. táblázat: CS1x bitek beállításai

Az ICN1 és ICES1 bitek nem kellenek a PWM jelhez, ezeket szintén nem részletezem.

Tehát, ha a mikrovezérlő 8MHz-en jár és az előosztót 64-nek választjuk; és 50Hz-es, gyors, 10 bites PWM jelet akarunk kapni az OCR1A lábon, melynek kitöltöttsége 50%, akkor ezt kell beleírnom a programba:

```
TCCR1A|=(1<<COM1A1)|(1<<COM1B1)|(1<<WGM11);
TCCR1B|=(1<<WGM13)|(1<<WGM12)|(1<<CS11)|(1<<CS10);
DDRD|=(1<<PD5); //OCR1A lab kimenetnek
ICR1=2499; //max. ertek
OCR1A=1249; // 50%-os kitoltottseg
```

3.1.4. Analóg-digitális átalakító

Az Atmega16-ban van egy 10 bites fokozatos közelítésű analóg-digitális átalakító, ami egy 8 csatornás multiplexerre van kötve, így 8 egyvégű feszültség bemenete lehet, melyek a port lábai. Az egyvégű bemenetek másik viszonyítási feszültsége a föld. Az eszközben van még továbbá 16 különböző kombinációjú kivonó bemenet - ebből kettő kombinációban még a konverzió előtt választható erősítéssel (0, 20 és 46dB) -, amiben a negatív bemenet mindig az ADC1. [25]

3.1.4.1. Fokozatos közelítésű analóg-digitális átalakító (SAR ADC)

A fokozatos közelítésű analóg-digitális átalakító (angolul Successive Approximation ADC) a visszacsatolt AD átalakítók egyik típusa. A bemenetére érkező jelet ciklikus összehasonlítással (közelítéssel) határozza meg. A jel átalakítása maximum annyi lépes, ahány bites a felbontás (8 bites felbontás esetén tehát 8 lépésből áll). Előnye az amplitúdó független konverziós idő és olcsónak mondható ára. [41]

3.1.4.2. Beállítása

Az ADC beállításáért két regiszter felelős: az ADMUX (ADC Multiplexer Selection Register) és ADCSRA (ADC Control and Status Register A), a konverzió értékét szintén két regiszter tárolja, mert 10 bites a felbontása: ADCH és ADCL, amit C-ben egy regiszterben is lehet kezelni, ami az ADC.

Bit	7	6	5	4	3	2	1	0	
	REFS1	REFS0	ADLAR	MUX4	MUX3	MUX2	MUX1	MUX0	ADMUX
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

26. ábra: ADMUX regiszter

A REFSx (Reference Selection) bitek felelősek a referenciafeszültség kiválasztásáért:

Table 83. Voltage Reference Selections for ADC

REFS1	REFS0	Voltage Reference Selection
0	0	AREF, Internal Vref turned off
0	1	AVCC with external capacitor at AREF pin
1	0	Reserved
1	1	Internal 2.56V Voltage Reference with external capacitor at AREF pin

12. táblázat: REFSx bitek beállítása

Az ADLAR() bit felelős a konverzióeredmény tárolásának igazításáért a két tároló regiszterben (ADCH és ADCL). Ha ADLAR 0, akkor az eredmény jobbra lesz igazítva, azaz az érték két felső bitje lesz csak az ADCH-ban. Ha 1 az értéke, akkor balra lesz igazítva, azaz a felső 8 bitje lesz tárolva az ADCH-ban, a maradék kettő az ADCL-ben. Ha C-ben programozunk, akkor nyugodtan hagyhatjuk 0-ban, mert a fordító segítségével tudjuk egyben kezelni a két regisztert. Viszont az A/D konverzió alsó két bitje általában eléggé zajos, ezért egyszerűbb nem foglalkozni velük. Ha balra rendezzük az eredményt és csak az ADCH-val törődünk, akkor ugyan csak 8-bites lesz a konverter felbontása, de kevésbé lesz zajos az eredmény.

Az MUXx bitek felelősek a csatorna és az erősítés kiválasztásért:

Table 84. Input Channel and Gain Selections

MUX4..0	Single Ended Input	Positive Differential Input	Negative Differential Input	Gain
00000	ADC0	N/A		
00001	ADC1			
00010	ADC2			
00011	ADC3			
00100	ADC4			
00101	ADC5			
00110	ADC6			
00111	ADC7			
01000	N/A	ADC0	ADC0	10x
01001		ADC1	ADC0	10x
01010		ADC0	ADC0	200x
01011		ADC1	ADC0	200x
01100		ADC2	ADC2	10x
01101		ADC3	ADC2	10x
01110		ADC2	ADC2	200x
01111		ADC3	ADC2	200x
10000		ADC0	ADC1	1x
10001		ADC1	ADC1	1x
10010		ADC2	ADC1	1x
10011		ADC3	ADC1	1x
10100		ADC4	ADC1	1x
10101		ADC5	ADC1	1x
10110		ADC6	ADC1	1x
10111		ADC7	ADC1	1x
11000		ADC0	ADC2	1x
11001		ADC1	ADC2	1x
11010		ADC2	ADC2	1x
11011		ADC3	ADC2	1x
11100		ADC4	ADC2	1x

13. táblázat: MUX4..0 bitek beállítása: Bemeneti csatorna és erősítés

Bit	7	6	5	4	3	2	1	0	
	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0	ADCSRA
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

27. ábra: ADCSRA regiszter

Az ADEN (A/D Enable) bittel lehet engedélyezni a konvertert. Ha 0, akkor az A/D ki van kapcsolva, ha 1, akkor be. Ha konverzió közben váltjuk át, akkor a konverziónak vége szakad.

Az ADSC (ADC Start Conversion) bittel lehet elindítani a konverziót, ha egyest írunk bele, akkor elkezdődik, és amikor kész lesz, akkor visszavált nullára.

Az ADETE (ADC Auto Trigger Enable), az ADIF (ADC Interrupt Flag) és az ADIE (ADC Interrupt Enable) biteket nem részletezem, mert nem akarom triggerelni a jelet és a feladat megvalósításhoz nem szükséges az ADC megszakítás lekezelése.

Az ADPSx bitek beállításával lehet leosztani az órajelet, a konverter órajelének a meghatározáshoz, amivel konkrétan a mintavételezési időt tudjuk megadni:

Table 85. ADC Prescaler Selections

ADPS2	ADPS1	ADPS0	Division Factor
0	0	0	2
0	0	1	2
0	1	0	4
0	1	1	8
1	0	0	16
1	0	1	32
1	1	0	64
1	1	1	128

14. táblázat: ADPSx bitek beállítása

3.1.4.3. Használata

Ha 8MHz es CPU frekvencián mondjuk 16 us-os mintavételezéssel az ADC0 és ADC1 láb közti feszültséget erősítés nélkül akarjuk balra igazítottan meghatározni, ahol a referencia feszültséget az AREF láb adja, akkor ezekkel a beállításokat kell beleírni a programba:

```
ADMUX=(1<<REFS0)|(1<<ADLAR)|(1<<MUX4);
ADCSRA=(1<<ADEN |(1<<ADPS2)|(1<<ADPS1)|(1<<ADPS0);
```

Innen az analóg feszültség kiolvasása, egy függvénnyel:

```
int8_t adc()
{
ADCSRA = (1 << ADSC);

while(!(ADCSRA & (1<<ADIF)));

ADCSRA|=(1<<ADIF);

return(ADCH);
}
```

Ebben elindítjuk a konverziót, majd egy while ciklussal figyeljük, hogy kész van-e, majd ha kész, akkor kitöröljük az A/D Interrupt Flaget, hogy ne okozzon gondot; majd visszatérünk a konverzió értékével.

3.1.4.4. Konverzió eredménye

A konverzió értéke az alábbi képlet alapján számolható:

$$ADC = \frac{(V_{pos} - V_{neg}) \cdot G \cdot K \cdot 512}{V_{ref}}$$

Ahol ADC a konverzió eredménye, G az erősítés, Vpos és Vneg a pozitiv és negativ bemeneti feszültség. Ha egycsatornás sima konverzióról van szó, akkor K 2, a Vneg pedig 0; ha pedig komparálás, akkor K 1. Az alábbi táblázat tartalmazza néhány lehetséges értéket, és a negativ számok ábrázolását a mikrokontrollerben.

Table 82. Correlation between Input Voltage and Output Codes

V_{ADCn}	Read code	Corresponding Decimal Value
V_{ADCm} + V_{REF}/GAIN	0x1FF	511
V_{ADCm} + 511/512 V_{REF}/GAIN	0x1FF	511
V_{ADCm} + 510/512 V_{REF}/GAIN	0x1FE	510
...	...	...
V_{ADCm} + 1/512 V_{REF}/GAIN	0x001	1
V_{ADCm}	0x000	0
V_{ADCm} - 1/512 V_{REF}/GAIN	0x3FF	-1
...	...	...
V_{ADCm} - 511/512 V_{REF}/GAIN	0x201	-511
V_{ADCm} - V_{REF}/GAIN	0x200	-512

15. táblázat: Bemeneti feszültségek és kimeneti értékek digitális komparálásnál

3.1.4.5. További lehetőségek

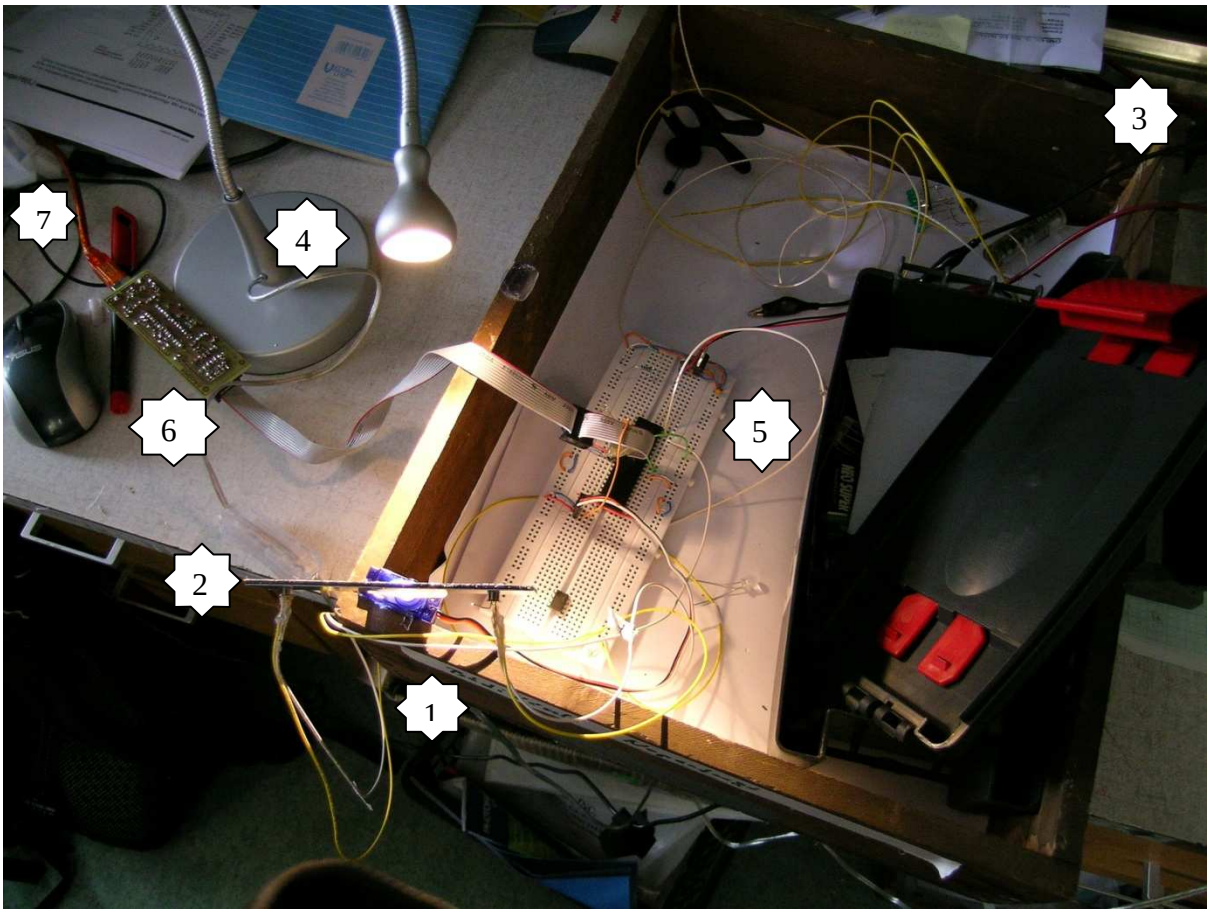
Használható az AD konverterre egy bonyolultabb függvény is, amivel ugrálni tudunk a csatornák közt, az argumentumába írt szám segítségével: [25]

```
void init_adc()
{
          ADMUX=(1<<REFS0);
          ADCSRA=(1<<ADEN)|(1<<ADPS2)|(1<<ADPS1)|(1<<ADPS0);
}

uint16_t adc(uint8_t ch)
{
          ADMUX&=0xF0;//elozmenyek torlese
          ch&=0x0F;// csak a mux bitek
          ADMUX|=ch;

          ADCSRA|=(1<<ADSC);//inditas

          ADCSRA & (1<<ADIF)));//varakozas

          ADCSRA|=(1<<ADIF); //interrupt flag torlese

          return(ADC);
}
```

3.2. Programozás

3.2.1. Fejlesztői környezetem

Ahhoz, hogy a programot megfelelően tudjam megírni és tesztelni, készítenem kellett egy fejlesztői összeállítást. A szervomotort felragasztottam a ládám belső oldalára, a szervomotorhoz járó egyik mozgató karra ráragasztottam ez előzőleg fototranzisztorok rögzítése miatt kifúrt dibond lapot. A szervomotort és az áramköri elemeket bedugtam a próbapanelbe, az 1. kapcsolási rajz szerint, és összekötöttem a programozót a számítógéppel egy USB kábel, és a mikrokontrollerrel egy szalagkábel segítségével.

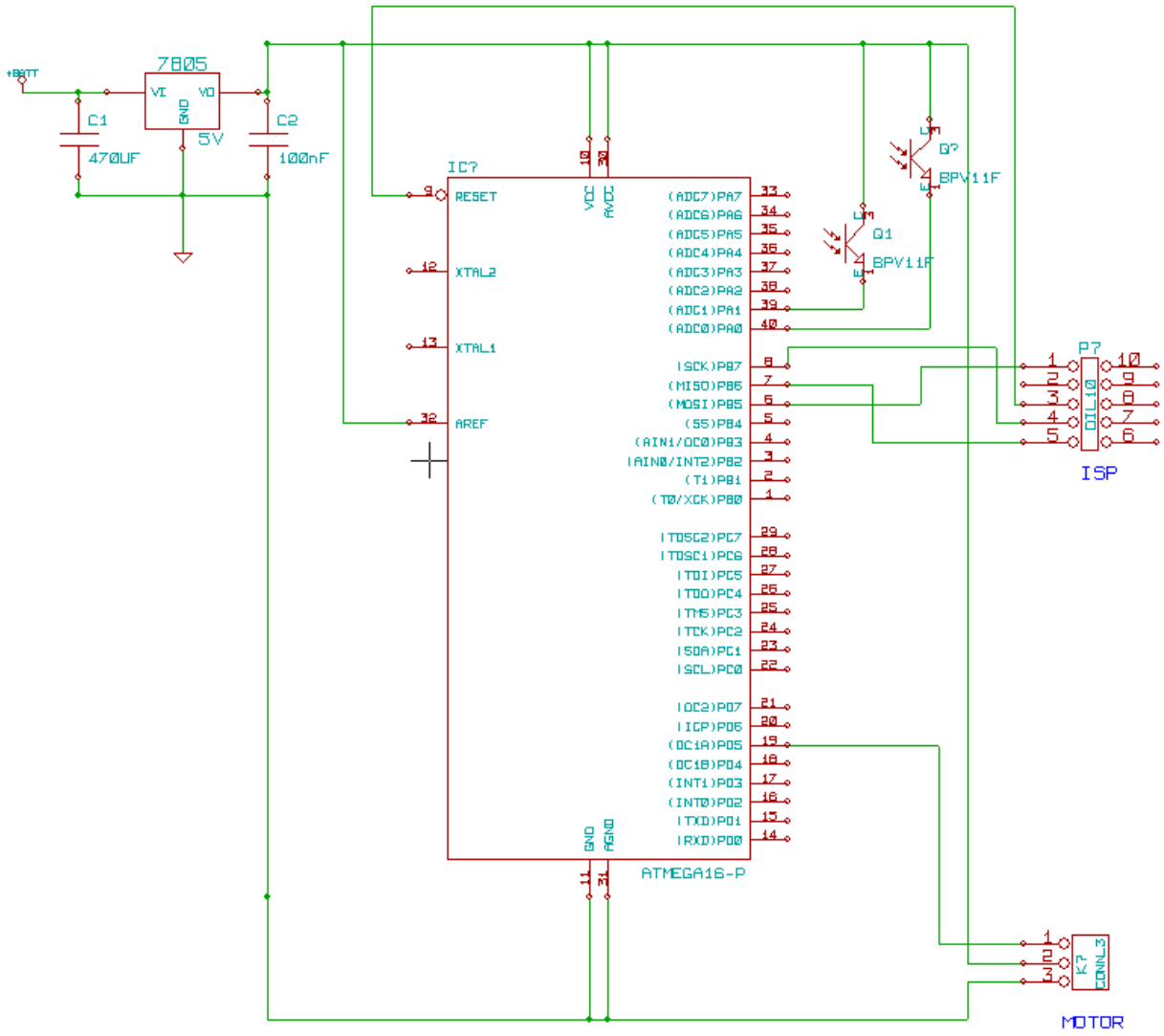


28. ábra: Fejlesztői környezetem

Fejlesztői környezet tartalma:

1. Szervomotor
2. Szervomotor tengelyére erősített dibond lap a fototranzisztorokkal
3. Tápegység
4. Lámpa
5. Próbapanel
6. Programozó
7. Laptop

Ehhez az alábbi kapcsolást használtam:



29. ábra: Kapcsolási rajz 1.

Melyben egy 5V-os feszültségstabilizátor szolgáltatja a megfelelő tápfeszültséget, a fototranzisztorokat PA0, és PA1 lábakra kötöttem be, valamint a PD5-re kötöttem rá a szervomotor vezérlő kábelét. Referencia feszültségnek 5Vot választottam, mert ez esik a tranzisztorokon is, így ezt kötöttem be az AREF lábra.

Ezzel az összeállítással megvan a fejlesztés iránya: ha elmozdítom a lámpát, akkor a program célja, hogy beforgassa a lapot a lámpa sugárzására merőlegesen. E cél megvalósítására sok ötletem és gondolatom támadt, az első volt a legkomplexebb és a legkevésbé megvalósítható. Viszont mielőtt nekiállhattam programozni a szervomotorok bemérését is meg kellett ejtenem.

3.2.2. Szervomotorok bemérése

Pontosan meg kellett határozni, hogy 8MHz-en működő mikrokontrollerrel generált 50Hz-es 10 bites PWM-nél milyen kitöltési értékeknél milyen szöghelyzetet vesznek fel a szervók.

Ehhez az alábbi programot használtam, amelyben az OCR1A értékeket kell változatni a bemérés során [26]:

```
#include <avr/io.h>
#include <util/delay.h>

void wait_ms(uint16_t ms)
{
        while ( ms )
        {
                _delay_ms(1);
                ms--;
        }
}

void main()
{
   TCCR1A|=(1<<COM1A1)|(1<<COM1B1)|(1<<WGM11);
   TCCR1B|=(1<<WGM13)|(1<<WGM12)|(1<<CS11)|(1<<CS10);
   ICR1=2499;  //fPWM=50Hz
   DDRD|=(1<<PD4)|(1<<PD5);

while(1)
   {
      OCR1A=97;   //bal veghelyzet
      wait_ms(500);
      OCR1A=213;  //kozephelyzet
      wait_ms(500);
      OCR1A=330;  //jobb veghelyzet
      wait_ms(500);
   }
}
```

Megvett három szervómotor:

1. Corona-RC DS-929-MG (Digitális)
2. Corona-RC CS-929-MG
3. Corona-RC CS-929-MG (Analóg)

Mért értékek:

Szervómotor	1	2	3
bal véghelyzet(1)	80	90	107
közép helyzet(1)	195	195	195
jobb véghelyzet(1)	310	310	300
tartomány(°)	150	150	150

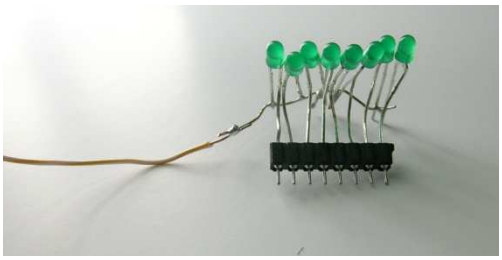
16. táblázat: Szervók PWM kitöltési tényező-szöghelyzet értékei

3.2.3. Próbálkozások

3.2.3.1. Első gondolat

A kezdeti elképzelés az volt, hogy két szenzort használva fogok fényt mérni, azaz az dibond két oldalán mért fényerősséget kivonom egymásból, majd e különbségből állapítok meg egy szabályzó jelet. Ezt azért tehetem meg, mert a teszteléshez lámpát használok, aminek kúp szögben terjed a kibocsájtott fénye. A Napból viszont párhuzamos fény jön, így későbbiekben meg kellett volna határoznom egy ideális szöget, amit a két szenzor bezár, hogy értelmes különbséget kapjak.

Az első gondolatom az volt, hogy az A/D átalakítót komparátor módban használom, hogy kapjak egy különbség értéket. Először még az A/D működés helyességéről akartam meggyőződni, így csak azt írtam meg. Továbbá a port B mind a nyolc lábára kikötöttem egy LED-et a megjelenítéshez, mivel az STK-500-as nem tud debuggolni.



30. ábra: LED sor

Ehhez be kellett állítanom az A/D konvertert, hogy az PA0 és a PA1 közötti feszültséget, erősítés nélkül balra eltoltan tárolja el, továbbá a port B lábait kimenetté kellett tenni. PA0 és PA1 lábat nem kellett bemenetté tenni, mert az, az alapbeállítás. Az A/D átalakítást az előzőekben ismertetett adc(ch) függvénnyel (amit 8bitesre jelölt egészre módosítottam és csak az ADCH adja vissza) végeztem, ahol ch értéke 8, ami a megfelelő komparátor módot jelenti:

```
#include <avr/io.h>
int8_t adc(uint8_t ch){};//kifejtve lásd fent
void main()
{
     ADMUX=(1<<REFS0)|(1<<ADLAR);
     ADCSRA=(1<<ADEN |(1<<ADPS2)|(1<<ADPS1)|(1<<ADPS0);

     DDRB=255;//B labak kimenetek

while(1){PORTB=adc(8);}
}
```

Valamiért ez a megoldás nem váltotta ki a hozzáfűzött reményeket, a pozitív különbségeket rendesen megjelenítette, viszont a negatívokat nem: néha rendesen kiírta az eredményt néha egyszerűen kiírta, hogy mínusz egy. (A negatív számok ábrázolása kettes komplemenssel történik, ami a 15. táblázatban látható). Így jött a második gondolat.

3.2.3.2. Második gondolat

Második gondolatom az volt, hogy hanyagolom a negatív különbségeket és az A/D komparátort, és sima két egycsatornás A/D után egy HA feltétellel először eldöntöm melyik nagyobb, majd abból vonom ki a kissebet. Majd e különbséget jelenítem meg. Itt 8 bites jelöletlen egészre módosítottam az adc(ch) függvényt és csak az ADCH-t adattam vissza vele.

```
#include <avr/io.h>
uint8_t adc(uint8_t ch){};//kifejtve lásd fent
int_8t a,b;
void main()
{
     ADMUX=(1<<REFS0)|(1<<ADLAR);
     ADCSRA=(1<<ADEN |(1<<ADPS2)|(1<<ADPS1)|(1<<ADPS0);

     DDRB=255;//B lábak kimenetek

     while(1)
     {
     a=adc(0); b=adc(1);
     PORTB=a>b?(a-b):(b-a);
     }
}
```

Ez a módszer bevált, sikeresen jelenítette meg az értékeket mérés során, úgyhogy ezt fejlesztem tovább. Tehát programot átfogalmaztam: a mikrokontroller vezérléséhez szükséges beállításokat és az A/D inicializálását, a header fájlokat és a wait_ms() függvény deklarációját beleraktam egy napra.h nevű header fájlba, ami megtekinthető a mellékletben.

A pwm_a 0°-hoz tartozó OCRx érték, valamint a pwm_a+tart a másik véghelyzethez tartozó érték, így tart a mozgási tartományt jelenti.

Ezzel a programmal egy digitális P jellegű szabályzást akartam megvalósítani, ami azt csinálja, hogy először beáll középhelyzetbe, majd a két szenzor jelének különbségének a p hányadát adja hozzá az OCR1 aktuális értékéhez, majd vár 100ms-ot, aztán újrakezdi a különbségképzéstől. Elméletileg ez a program bemozgatta volna a- p értéktől is függő- ideális helyzetbe, mert addig megy, amíg a hozzáadandó hányados nem 0. Azért nem lehet tetszőleges p értéket megadni, mert ha túl kicsi, akkor túl lesz szabályozva és sokáig fog ide-oda ugrálni az ideális helyzethez körül. Viszont ha túl nagy, akkor nem tudja megtalálni az ideális helyzetet, mert kisebb számot egészesen osztva nagyobbal nullát kapunk eredményül, így minél nagyobb a p, annál nagyobb az eltérés az ideálistól. Pontosan megfogalmazva 2p lesz a maximális eltérés. Tehát a p értekkel a beállási sebességet és a beállási pontosságot egyaránt határozzuk meg, és sajna p csak egész érték lehet.

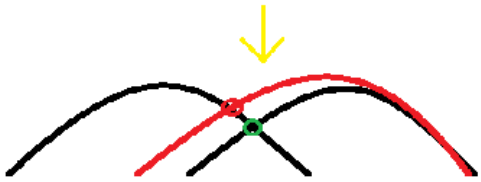
3.2.3.3. Második gondolat forráskódja

```
#include <napra.h>
void main()
{
     init();
     uint8_t pwm_a=97, tart=233, t1, t2, ki, p=8;
     OCR1A=pwm_a+tart/2; //középre beallas
     wait_ms(100);

     while(1)
     {
          t1=adc(0);
          t2= adc(1);

          if(t1>t2)
          {
               ki=(t1-t2);
               ki/=p;    //P faktor
               OCR1A-=ki;
          }
          else
          {
               ki=(t2-t2);
               ki/=p;    //P faktor
               OCR1A+=ki;
          }
     wait_ms(100);
     }
}
```

Ez a szabályzás néha működött: azaz volt, amikor beállt rendesen a lámpa felé, viszont volt, amikor egyáltalán nem; és volt olyan is, amikor egyik helyzetből a másik véghelyzetbe kezdte csapkodni a motort. Konkrétan végigpróbálgattam a p értékeket 1-től 32-ig (a tart=233 miatt, följebb már nem volt értelme menni), de egyik se bizonyult sikeresnek. Ezért segítségez folyamodtam.

3.2.3.4. Konzultáció

Bementem Dr. Lipovszki György tanár úrhoz, aki a Digitális szabályozás c. tárgyat tanította nekünk, hogy segítségét kérjem a problémám megoldáshoz. Ő mondta, hogy kétszenzoros szabályzásnál fontos, hogy a két szenzor karakterisztikája megegyezzen, mert itt nem a jelleg görbe maximumát találom meg, hanem a két görbe metszéspontját, ahol ugyanakkora az értékük. Tehát, ha kicsit is különböznek, akkor a két görbe metszéspontja nem az ideális helyzetnél lesz, hanem attól távol.

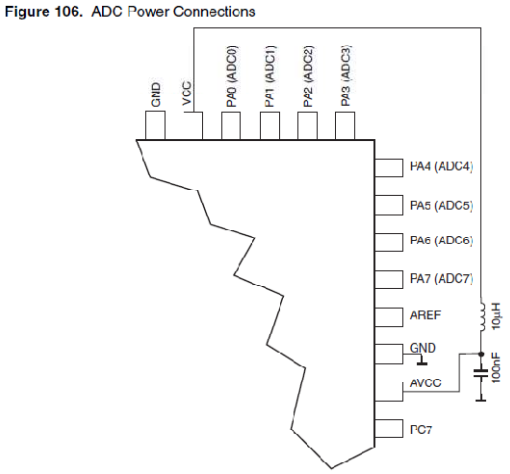


31. ábra: Konzultáció

Mivel a Nap viszonylag lassan mozog, ezért azt ajánlotta, hogy egy szenzorral végezzem el a szabályozást, mégpedig úgy, hogy bizonyos időközönként meghatározzuk, hol van a Nap és abba az irányba fordítjuk a napelemet. A Nap helyzetét, meg egyszerűen, úgy határozzuk meg, hogy meghatározott mennyiségű osztással végigmegyünk a teljes tartományon, és minden osztásnál megmérjük a fény erősségét. Majd ha ez megvan, akkor kiválasztjuk a legnagyobbat és arra a helyre beállunk. Ezt a pozíciót, a következő végigmérésig tartjuk.

3.2.4. Működő program

A megvalósításhoz módosítanom kellett a kapcsolási rajzott, úgy, hogy csak az egyik fototranzisztort használja, amelyet a PA0-s lábra kötöttem be. Másrészt a kapcsolásba beleraktam a pontosabb mérésekhez szükséges elemeket: egy-egy nagyobb kondenzátorral (C4 és C5) kötöttem össze a föld és a táp lábakat, valamint egy tekercs segítségével az alábbi, Atmega16 katalógusában [16] lévő az zajszűrő kapcsolását valósítottam meg. Harmadrészt bekötöttem az ISP lábakat, valamint egy tápcsatlakozót is. A kész kapcsolási rajz a mellékletben megtalálható.



32. ábra: Zajszűrő kapcsolás

A program maga azt csinálja, hogy a szervomotor mozgási tartományát felosztja n részre, - végleges verzióban 10 részre; az elején beáll a tartomány aljára majd lépkedve végigfordul a résztartományokon, és minden megállásnál megméri a fény intenzitását. Majd a mért értékek közül kiválasztja a legnagyobb intenzitású helyet és beáll oda. Tárolhattam volna n nagyságú statikus tömbben is a mért értékeket, majd ezek közül a körbe forgás után egy maximumérték-keresést elvégezve választottuk volna ki a legjobb helyet. Ez is tökéletesen működött kisebb elemszámú tömböknél, viszont nem biztos, hogy nagyobb osztásnál is működött volna, amit nem próbáltam ki, mert kitaláltam, hogy a maximumkeresést már a méréskor is elvégezhetjük, ha az épp aktuális mért értéket közvetlenül a mérés után összehasonlítjuk az eddigi maximummal.

3.2.4.1. Működő program forráskódja

```
#include <napra.h>

void main()
{
init();

while(1)
   {
     uint8_t i, pwm_a=97, tart=233, tarolo, max, maxhely, oszto;
     oszto=10;

     i=0;
     max=0;
     maxhely=0;
     tarolo=0;

     while(i<(oszto+1))
     {
          OCR1A=pwm_a+i*tart/oszto;
          wait_ms(1000);
          adc();
               tarolo=ADCH;
          if(tarolo>max)
          {
               maxhely=i;
               max=tarolo;
          }
          i++;
     }

     OCR1A=pwm_a+maxhely*tart/oszto;
     wait_ms(10000); //hosszu varakozas
     }
}
```

A végleges program forrása annyiban tér el ettől, hogy ott az első motor beállása után következik a második motor beállítása is, majd utána a hosszabb várakozás.

4. Összeszerelési tapasztalatok

Az idő szűkössége miatt nem volt lehetőségem az eltervezettnek megfelelően legyártani az alkatrészeket, így például az alsó házba nem került bele a szervót tartó lépcsős bemarás. A szervomotort a lábakat is rögzítő bilinccsel szorítottam oda a házhoz.

Az o-gyűrűk beszerzésére sem volt időm, így o-gyűrű szálakat vágtam méretre, majd végtelenítettem őket Loctite rugalmas pillanatragasztóval. Ez a módszer is remekül bevált. Már 15 másodperccel a ragasztás után, csak az o-gyűrű másfélszeres megnyúlásánál szakadt el a ragasztás.

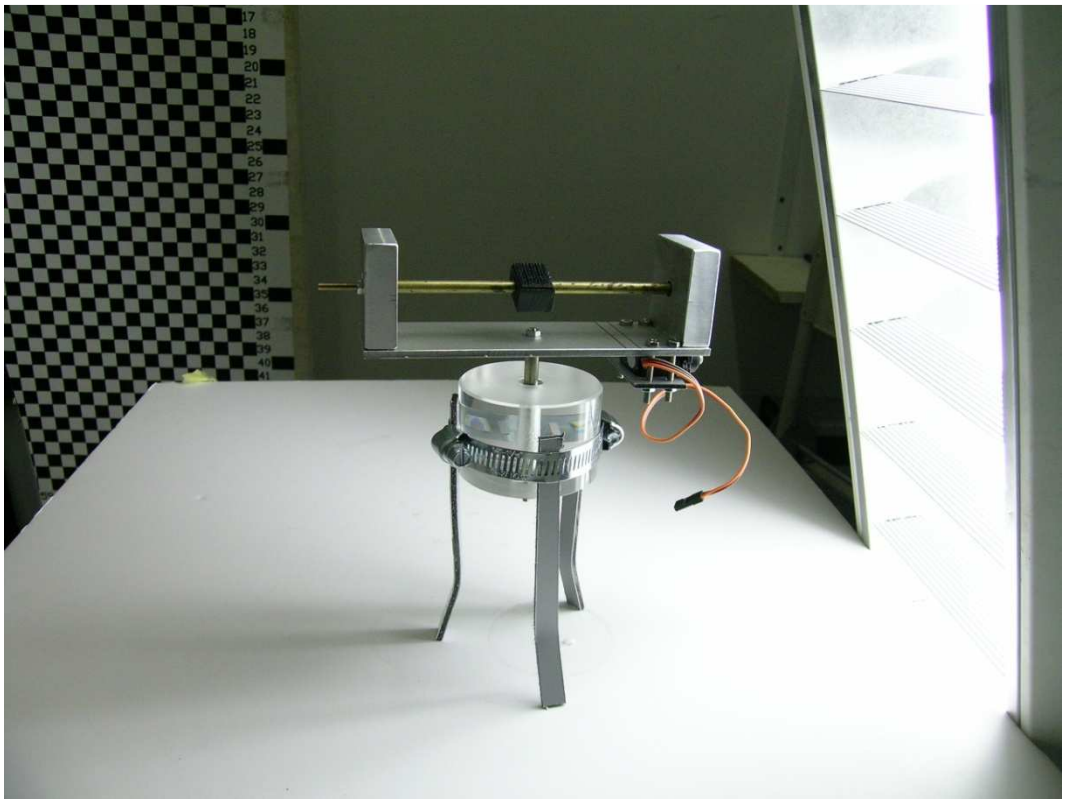
A hajtó és hajtott tárcsák epoxi-ragasztóval történő rögzítése tökéletesen működik: 6 perccel a ragasztás után, már csak anyagroncsolással, azaz harapófogóval tudtam volna őket szétszedni.

A másik szervónak a felső házhoz való rögzítése során problémám akadt: harmadik próbálkozásra sem sikerült megfelelően hajlított lemezalkatrészt készítem, ezért egy négy ponton átfúrt viszonylag rugalmas dibond lappal és csavarkötésekkel rögzítettem a ház aljára a motort.

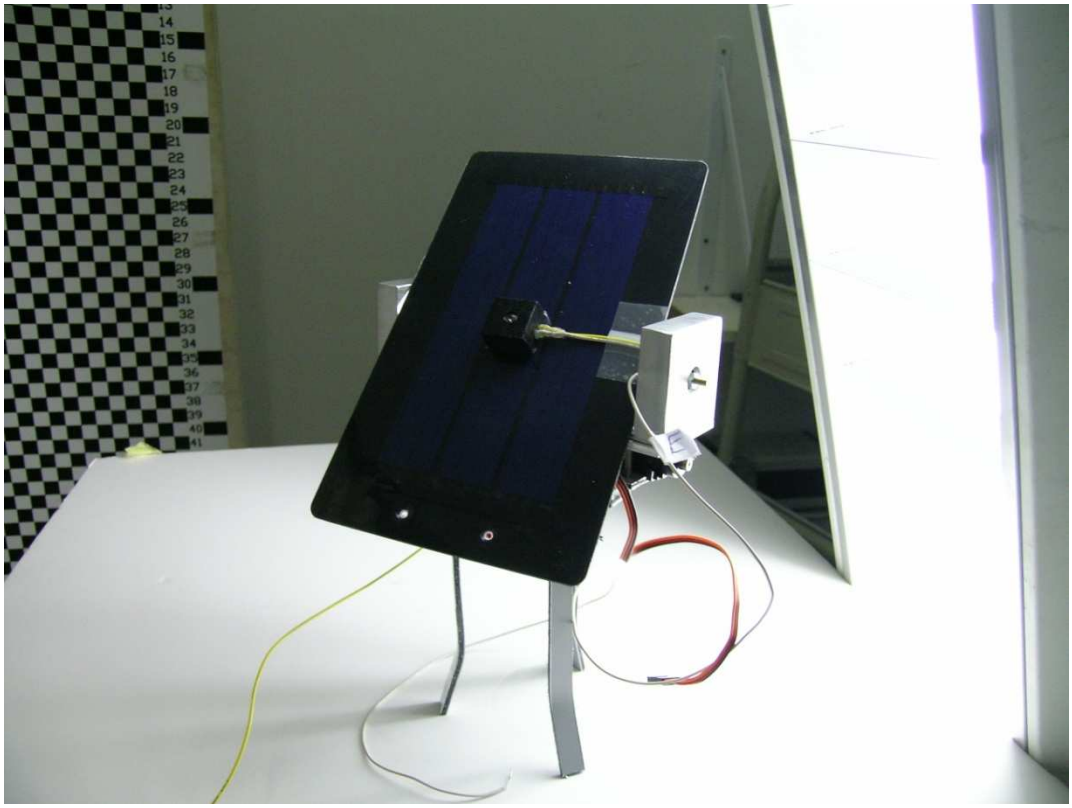
A napelemet a könnyebb és biztonságosabb szállíthatóság érdekében tépőzáras ragasztószalaggal rögzítettem a napelem tartó hasábhoz.

A többi alkatrész, azaz a csapágyházak szerelése tervszerűen ment. Fontos megjegyeznem, hogy az alsó csapágyháznál jöttem rá arra, hogy O-elrendezés ide vagy oda, a csapágyak egymással szembenéző belső futógyűrűinek felületét mindenképp meg kellett volna támasztanom egy távtartó csővel, mert amikor erősebben rászorítottam az anyát, eltörtem az egyik csapágyat. A Seeger-gyűrű viszont kell, mert szereléskor kiesik a csapágy. Legközelebb mind a négy ponton meg fogom fogni a csapágyat, viszont a megfelelő elrendezés kialakítása végett az egyik megfogásnál, hagyok majd egy kis játékot, ami így nem teszi túlhatározottá a csapágyrögzítést.

A nyák legyártása a szakdolgozat megírásakor még nem történt meg, viszont próbapanelen összeállítottam a kapcsolást. Miután rájöttem, hogy egy sima 9V-os elem nem bírja el a két szervomotor áramfelvételét, a tápellátást a régi laptopom töltőjének átalakításával oldottam meg. Tapasztalatom itt az, hogy értékes áramköri elemek megvédésre érdemes minél több egyenirányítót használni, és be kell építeni egy megfelelően méretezett biztosítékot, mert az egyik szervomotor rossz bekötésénél sikerült kiégetnem a feszültség stabilizátort.



33. ábra: Összeállítás napelem nélkül



34. ábra: Összeállítás napelemmel

5. Összefoglalás

Nagyon örülök, hogy az Optikai mérnökiroda Kft.-nél írhattam a szakdolgozatomat, mert technikai felszereltsége és az ott dolgozók segítőkészsége nagy előnyt jelentett a munkámban.

A programozási részben leírtak alapján egyszerűbb robotikai probléma megoldható. Például ADC-vel vonalkövetés optokapuval, fal és akadály elkerülés ultrahangos távolságmérővel, továbbá a PWM jellel és egy H-hidas IC-vel lehet RC és léptető motorokat is vezérelni.

Szakdolgozatomban nem alkottam új dolgot, de megmutattam, hogy értek a mechatronika három területéhez és az egyedi gyártástechnológiához. Az elvégzett munka kevesebb, mint a harmada telt a konkrét fejlesztéssel, a többi idő az alkatrészek beszerzésével és legyártásával és a szakdolgozat megírásával telt el. Sok gyakorlati tudást szereztem az alkatrészek elkészítése során, mindazonáltal legközelebb a hardver legyártását erre szakosodott cégekre fogom bízni, amivel időt és pénzt spórolok meg.

Összeállítottam egy táblázatot a legyártási és fejlesztési költségek körülbelüli értékéről. Az órabérem alternatív költségét 1000Ft-ban állapítottam meg:

Megnevezés	Típus	Mennyiség	Ár	Költség
Szervomotor	db	3	3000	9000
Mikrokontroller	db	1	1000	1000
Elektronikai alkatrészek	db	1	3000	3000
Mechanikai tervezés	óra	15	2000	30000
Elektronikai tervezés	óra	5	1000	5000
Programozás	óra	112	1000	112000
Legyártás	óra	168	1000	168000
Szakdolgozat megírása	óra	112	1000	112000
Összesen:				440000

17. táblázat: Költségek

5.1. Továbbfejlesztési lehetőségek

A napra-forgó robotot szerkezetét úgy lehetne továbbfejleszteni, hogy időjárásállóvá tesszük, azaz a méretezzük a természetbeni körülményre: szélerősségre, UV sugárzásra és esőre. Tehát meg kell oldani a szerkezet eső elleni leszigeteltségét; a hajtásrendszert méretezni kell orkán erejű szélre is; a szíjhajtást, mert az UV sugárzás tönkreteszi a gumikat és nem biztosít megfelelő szögmerevséget, ki kell váltani fogaskerekes visszahatás nélküli hajtóművel. Továbbá, ha visszahatásmentes hajtóművet alkalmazunk, akkor nem kell folyamatosan járatni a motorokat, hogy megtartsák a terhelés ellenében a szögpozíciót, így energiát is spórolhatunk.

Másrészt egy nagy napelem forgatása helyett lehetne alkalmazni egy nagy parabolatükröt, ami a fényt egy kisebb napelemre fokuszálja, így használhatunk drágább, de nagyobb hatásfokú napelemet is. A mellesleg a fókuszált fénnyel hajthatunk egy Stirling-motort vagy akár melegíthetünk vizet is vele.

18. táblázat: A parabola alakú napkollektor Stirling motorral [14]

Harmadrészt a szervomotork RC motorokra való cserélésével és szögelfordulás mérő szenzorral kisebb pénz ráfordításával erősebbé és pontosabbá tehető a szerkezet.

A szoftvert kétféleképpen is tovább lehet fejleszteni. Egyrészt megfelelő szabályozási algoritmus kidolgozásával ki lehet iktatni a maximumkeresési időszakokat, továbbá ha túl gyenge a napsugárzás, akkor alvó üzemmódba kapcsolja magát: lehető legkisebb légellenállású alakot veszi fel és a mikrokontrollert SLEEP üzemmódra állítja. Ez különösen jól jöhet viharos időben. Másrészt a szoftverbe be lehet építeni azt is, hogy folyamatosan kövesse a Nap mozgását, anélkül, hogy szabályozva lenne, azzal, hogy telepítéskor megmondjuk neki melyik szélességi fokon milyen irányba helyeztük el. Egy adatbázis és egy óra segítségével ezekből az adatokból pontosan tudja, hol van a Nap. Továbbá, ha a Nap lement, akkor alvó üzemmódba kapcsolja magát az energia spórolás érdekében.

6. Mellékletek

6.1. Ábrajegyzék

6.2. Csatolmányok

- Programforrás
- A4SD-10-100: Összeállítási rajz
- A4SD-10-101: Csapágyház 1
- A4SD-10-103: Tengely 1
- A4SD-10-107: Szervomotor
- A4SD-10-108: Tengelykapcsoló
- A4SD-10-109: Hajtott kerék 1
- A4SD-10-110: Hajtó kerék
- A4SD-10-112: Szorító
- A4SD-10-115: Hajtott kerék 2
- A4SD-10-200: Csapágyház 2
- A4SD-10-201: Alaplap
- A4SD-10-202: Oldal
- A4SD-10-205: Tengely 2
- A4SD-10-300: Kapcsolási rajz
- A4SD-10-400: Nyákterv

6.3. DVD melléklet tartalma

- Szakdolgozat .pdf formátumban
- SKF 623 katalógus lapja
- AVR Atmega16 mikrokontroller dokumentációja
- L7800 feszültség stabilizátor dokumentációja
- BPV11F fototranzisztor dokumentációja
- AVR-Doper USB ISP programozó elkészítéséhez szükséges anyagok
- AVR Studio 4
- 3D modell összes fájlja
- Géprajzok, kapcsolási rajz, NYÁK terv
- Programforrás és lefordított hex fájl
- KiCAD elektronikai tervező program
- Siker2009 VEM program
- Tengely 1 méretezésének eredményei

6.4. Irodalomjegyzék

[1] Avr-doper: usb-s isp programozó; http://uc.hobbielektronika.hu/kapcsolasok_avr-doper_usb-s_isp_programozo.html; 2010.

[2] SKF, Egysorú mélyhornyú golyóscsapágyak, http://www.skf.com/portal/skf/home/products?maincatalogue=1&lang=en&newlink=1_1_1; 2010

[3] Tóth-Nagy-Marosfalvi, Gépelemek I., 2005

[4] Simon és tsai., Gépelemek 2., 2008

[5] Házkötő István, Műszaki 2D-s ábrázolás, 2006

[6] Gördülőcsapágyak élettartam számítása: http://www.uni-miskolc.hu/~wwwmach/tantargyaink/003b_gepelemek1/ge1_4fel_segedlet.pdf; 2010

[7] BME GT3 tanszék honlapja; http://gt3.bme.hu//index.php?option=com_wrapper&Itemid=143; 2010

[8] Kéttengelyes követő; http://www.patriotsolargroup.com/dataSheets/20kWDual-axisPVTracker.pdf; 2010

[9] Szervó motorok vezérlése; http://myprojects.hu/pic/szervo-motorok-vezerlese.html; 2010

[10] Atmel Products - AVR Solutions – Datasheets; http://www.atmel.com/dyn/products/datasheets.asp?family_id=607; 2010

[11] Lomex; http://www.lomex.hu/; 2010

[12] Sunflower; http://en.wikipedia.org/wiki/Sunflower; 2010

[13] Napelem – Wikipédia; http://hu.wikipedia.org/wiki/Napelem; 2010

[14] Napenergia – Wikipédia; http://hu.wikipedia.org/wiki/Napenergia; 2010

[15] RC modell Web áruház; http://www.rcmodell.com/shop/product_info.php/cPath/67_251/products_id/2940; 2010

[16] Autodesk Education Community; http://students.autodesk.com/; 2010

[17] BPX43-3 Datasheet pdf - NPN-Silizium-Fototransistor Silicon NPN Phototransistor – Siemens; http://www.datasheetcatalog.com/datasheets_pdf/B/P/X/4/BPX43-3.shtml; 2010

[18] TL072 Datasheet pdf - Dual Low-Noise JFET-Input General-Purpose Operational Amplifier - Texas Instruments; http://www.datasheetcatalog.com/datasheets_pdf/T/L/0/7/TL072.shtml; 2010

[19] AVR Studio 4; http://www.atmel.com/dyn/products/tools_card.asp?tool_id=2725&source=redirect; 2010

[20] ATmega16 dokumentációja; http://www.atmel.com/dyn/resources/prod_documents/doc2466.pdf; 2010

[21] Getting started; http://extremeelectronics.co.in/avr-tutorials/part-ii-getting-started/; 2010

[22] Introduction to pwm pulse width modulation; http://extremeelectronics.co.in/avr-tutorials/introduction-to-pwm-pulse-width-modulation/; 2010

[23] PWM signal generation by using avr timers; http://extremeelectronics.co.in/avr-tutorials/pwm-signal-generation-by-using-avr-timers/; 2010
[24] Programming in C - tips for embedded development ; http://extremeelectronics.co.in/avr-tutorials/programming-in-c-tips-for-embedded-development/; 2010
[25] Using the analog to digital converter ; http://extremeelectronics.co.in/avr-tutorials/using-the-analog-to-digital-converter/; 2010
[26] Servomotor control by using AVR ATmega32 microcontroller; http://extremeelectronics.co.in/avr-tutorials/servo-motor-control-by-using-avr-atmega32-microcontroller/; 2010
[27] AVR stdint könyvtár; http://www.nongnu.org/avr-libc/user-manual/group__avr__stdint.html; 2010
[28] Analóg elektronika letöltések; http://elektro.get.bme.hu/index.php?mid=aedown&lang=hu; 2010
[29] O-Ring Kft.; http://www.oring.hu/; 2010
[30] Iramko-fogaskerekek; http://www.iramko.com/Fogaskerek; 2010
[31] Lynxmotion robot ízület; http://www.lynxmotion.com/p-707-micro-pan-and-tilt-kit-black.aspx; 2010
[32] Corona analóg szervómotorok; http://www.graysonhobby.com/catalog/corona-analog-p-892.html; 2010
[33] Corona digitális szervómotorok; http://www.graysonhobby.com/catalog/corona-digital-p-695.html; 2010
[34] Hitec szervó kép; http://www.pyroelectro.com/tutorials/servo_motor/parts/hitec_servo_b.jpg; 2010
[35] Wikipédia - Fototranzisztor; http://hu.wikipedia.org/wiki/Fototranzisztor; 2010
[36] Loctite epoxi ragasztók; http://www.loctite.hu/huu/content_data/91938_Epoxy.pdf; 2010
[37] ;Wikipédia – Alpakka; http://hu.wikipedia.org/wiki/Alpakka; 2010
[38] PWM; http://digitus.itk.ppke.hu/~tihanyia/Segedlet/Modulacio/PWM.pdf; 2010
[39] WILL-I építése avagy nulláról a robotokig - AVR mikrovezérlők; http://www.hobbielektronika.hu/kapcsolasok/will-i_epitese_avagy_nullarol_a_robotokig_-_avr_mikrovezerlok.html?pg=8; 2010
[40] BME Műszaki Mechanika tanszék; http://www.mm.bme.hu/mm_hu/; 2010
[41] Wikipédia - Analóg-digitális átalakító; http://hu.wikipedia.org/wiki/Anal%C3%B3g-digit%C3%A1lis_%C3%A1talak%C3%ADt%C3%B3; 2010
[42] Biomimikri az élővilág ihlette találmányok; http://www.origo.hu/tudomany/20100120-biomimikri-az-elovilag-ihlette-talalmanyok.html; 2010
[43] Wikipedia – Biomimicry; http://en.wikipedia.org/wiki/Biomimicry; 2010
[44] Wikipedia – Solar tracker; http://en.wikipedia.org/wiki/Solar_tracker; 2010
[45] Wikipedia – Solar cell; http://en.wikipedia.org/wiki/Solar_cell; 2010

6.5. Programforrás

6.5.1.1. napra.h

```
#include <avr/io.h>
#include <util/delay.h>

void wait_ms(uint16_t ms)
{
     while( ms )
     {
          _delay_ms(1);
          ms--;
     }
}

int8_t adc(uint8_t ch)
{
          ADMUX&=0xF0;//elozmenyek torlese
          ch&=0x0F;// csak a mux bitek
          ADMUX|=ch;

          ADCSRA|=(1<<ADSC);//inditas

          ADCSRA & (1<<ADIF)));//varakozas

          ADCSRA|=(1<<ADIF); //interrupt flag torlese

          return(ADCH);
}
void init()
{
//PWM beállítás
     TCCR1A|=(1<<COM1A1)|(1<<COM1B1)|(1<<WGM11);
     TCCR1B|=(1<<WGM13)|(1<<WGM12)|(1<<CS11)|(1<<CS10);
     ICR1=2499;

//kimenetek
     DDRD|=(1<<PD4)|(1<<PD5);

// ADC
      ADMUX=(1<<REFS0)|(1<<ADLAR);
      ADCSRA=(1<<ADEN |(1<<ADPS2)|(1<<ADPS1)|(1<<ADPS0);

}
```

6.5.1.2. program.c

```
#include <napra.h>

void main()
{
     init();
     uint8_t i, pwm_a=97, tart=233, tarolo, max, maxhely, oszto;
     oszto=10;

     while(1)
     {
          i=0; max=0; maxhely=0; tarolo=0;

          while(i<(oszto+1))
          {
               OCR1A=pwm_a+i*tart/oszto;
               wait_ms(1000);
               adc();
                   tarolo=ADCH;
               if(tarolo>max)
               {
                   maxhely=i;
                   max=tarolo;
               }
               i++;
          }

          OCR1A=pwm_a+maxhely*tart/oszto;
          wait_ms(1000);

          i=0; max=0; maxhely=0; tarolo=0;

          while(i<(oszto+1))
          {
               OCR1B=pwm_a+i*tart/oszto;
               wait_ms(1000);
               adc();
                   tarolo=ADCH;
               if(tarolo>max)
               {
                   maxhely=i;
                   max=tarolo;
               }
               i++;
          }
          OCR1B=pwm_a+maxhely*tart/oszto;
          wait_ms(100000);
     }
}
```