DOKUZ EYLÜL ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

DÖRT EKSENLİ RC SERVO MOTOR TAHRİKLİ BİR ROBOT MANİPÜLATÖRÜ TASARIMI VE UYGULAMASI

Çağlar TOKEL

Kasım, 2009
İZMİR

DÖRT EKSENLİ RC SERVO MOTOR TAHRİKLİ BİR ROBOT MANİPÜLATÖRÜ TASARIMI VE UYGULAMASI

Dokuz Eylül Üniversitesi Fen Bilimleri Enstitüsü

Yüksek Lisans Tezi

Mekatronik Mühendisliği Bölümü, Mekatronik Mühendisliği Programı

Çağlar TOKEL

Kasım, 2009

İZMİR

YÜKSEK LİSANS TEZİ SINAV SONUÇ FORMU

ÇAĞLAR TOKEL, tarafından **YRD. DOÇ. DR. ZEKİ KIRAL** yönetiminde hazırlanan "**DÖRT EKSENLİ RC SERVO MOTOR TAHRİKLİ BİR ROBOT MANİPÜLATÖRÜ TASARIMI VE UYGULAMASI**" başlıklı tez tarafımızdan okunmuş, kapsamı ve niteliği açısından bir Yüksek Lisans tezi olarak kabul edilmiştir.

.....

Yrd. Doç. Dr. Zeki KIRAL

Danışman

.....

Jüri Üyesi

.....

Jüri Üyesi

Prof.Dr. Cahit HELVACI

Müdür

Fen Bilimleri Enstitüsü

TEŞEKKÜR

Bu bölümde ilk olarak, beni bugünlere getiren, her zaman yanımda olup bana güvenip destekleyen her şeyi borçlu olduğum canımdan çok sevdiğim benim için hayattaki en kıymetli insanlar olan sevgili annem ve babama en büyük teşekkürü borçlu olduğumu söylemeliyim.

Uzun süren çalışmalarım neticesinde böyle bir çalışmanın ortaya çıkmasını sağlayan, bana takıldığım her nokta da üstesinden geleceğimi söyleyen ve bana inanan, beş ay kadar bir süre kendi odasını çalışma alanım olarak kullanmama olanak veren, verdiği manevi desteğin yanında maddi olarak da yanımda olan değerli hocam ve tez danışmanım Yrd. Doc. Dr. Zeki KIRAL'a, tezin mekanik kısmının üretiminde tüm olanaklarını kullandığım ve bir çalışanı olduğum Deflab Robotics firmasına ve bana çok farklı bir perspektif kazandıran firmanın kurucusu ve sahibi sevgili büyüğüm Çağlar Emiroğlu'na, sekiz ay boyunca birlikte tasarım yaptığımız oda arkadaşım ve sevgili ağabeyim Övünç Akil'e tezim konusunda bana verdikleri manevi destek için teşekkür ederim.

Son paragrafı, benim için değerli zamanını ayırıp tezimdeki imla hatalarının düzeltilmesinde yardımcı olan ayrıca tezimi aşama aşama inceleyip bana farklı bakış açıları sunan benim için çok kıymetli sevgili ağabeyim Onur Tokel'e, yine imla hatalarının düzelmesinde zaman ayırıp yardımcı olan yakın arkadaşım Kadir Bozdemir'e, tezin ortaya çıkma sürecinde maddi manevi desteği ile her zaman yanımda olan sevgili arkadaşım Egemen Turan'a ayrı ayrı çok teşekkür ederim.

Çağlar TOKEL

DÖRT EKSENLİ RC SERVO MOTOR TAHRİKLİ BİR ROBOT MANİPÜLATÖRÜ TASARIMI VE UYGULAMASI

ÖZ

Bu çalışmada dört eksen ve bir tutucu'ya sahip olan bir robot manipülatör ve robotun kontrolü için mikroişlemci ailesinden PIC kullanılarak bir servo motor sürücü kartı tasarlanmıştır. Bu tez için kullanılan altı servo motoru sürebilen sürücü kartı ve seri haberleşme için kullandığımız kart, baskı devre tekniği kullanılarak üretilmiştir. Visual Basic programı kullanılarak bir arayüz oluşturulmuş ve robotun bilgisayardan bu arayüzle kontrolü sağlanmıştır. Gerekli görüldüğü takdirde bu amaç için yazılmış olan kodlar PIC içinde gömülü olan kodlarla değiştirilerek robot manipülör kendi kendini kontrol etme modunda da kullanılabilir. Tez için gerekli olan donanım göz önüne alınarak PIC ailesinin orta sınıf bir mikroişlemcisi tercih edilmiştir. Motor sürücü kartı ve seri haberleşme için kullanılan kart bilgisayar ortamında tasarlanmıştır. İlk önce breadboard'a kurulan devreler ile çalışmalar yapılmış, bu çalışmaların neticesinde robotun motorlarını sürmek için uygun bir sürücü kartı tasarlanmıştır. Robot kolun yapımında alüminyum kompozit seçilmiş, ana gövde, uzuvlar ve gripper'ın bağları, lazer ve cnc router'da imal edilmiştir.

Tasarlanan rc servo sürücü kartında bulunan PIC mikroişlemcisi yardımıyla, robot manipülatörün öğretme yoluyla istenen pozisyona gitmesi sağlanabildiği gibi aynı zamanda oluşturulan arayüzden seri iletişim protokolleri kullanılıp yollanan pozisyon bilgileriyle istenen hareketler yaptırılabilmektedir.

Anahtar Sözcükler: Robot, manipülatör, Rc, servo, motor, tutucu, mikroişlemci, mikrodenetleyici, pic, uygulama, seri, iletişim, baskı, devre, elektronik, Proteus, Solidworks, Visual Basic

DESIGN AND IMPLEMENTATION OF A 4-AXIS RC SERVO DRIVEN ROBOT MANIPULATOR

ABSTRACT

In this study, a robot manipulator, having four axes with a gripper, controllable with a PIC family microcontroller servo motor driver card is designed. The driver card used in this thesis is capable of driving six rc servo motors and the card we used for the serial communication was built with using printed circuit assembly technique. An interface is developed with Visual Basic and the control of the robot is accomplished through this interface. If necessary the robot manipulator can be used in self control mode by replacing the code embedded in the PIC with the one written for this purpose. Considering the hardware requirements, the PIC family's middle class microcontroller is chosen. The motor driver card and the card for the serial communications are designed in the computer environment. As a result of the initial studies on the breadboard, a suitable driver card is designed to drive the motors of the robot manipulator. Aluminium composit was chosen in the makings of the robot arm whereas main body, linkage and gripper links are manufactured with laser cutting and cnc router.

With the assistance of the PIC microprocessor embedded in the servo driver card, the robot arm can be directed to go to a desired position by teaching or it is possible to get the desired motions by the position data sent from the interface using serial communication protocols.

Keywords: Robot, manipulator, Rc, servo, motor, gripper, microprocessor, microcontroller, pic, application, serial, communication, printed, circuit, electronic, Proteus, Solidworks, Visual Basic

İÇİNDEKİLER

Sayfa

BÖLÜM BİR
GİRİŞ

1.1 Motor Seçimi

1.1.1 Step Motorlar (Adım Motorları)

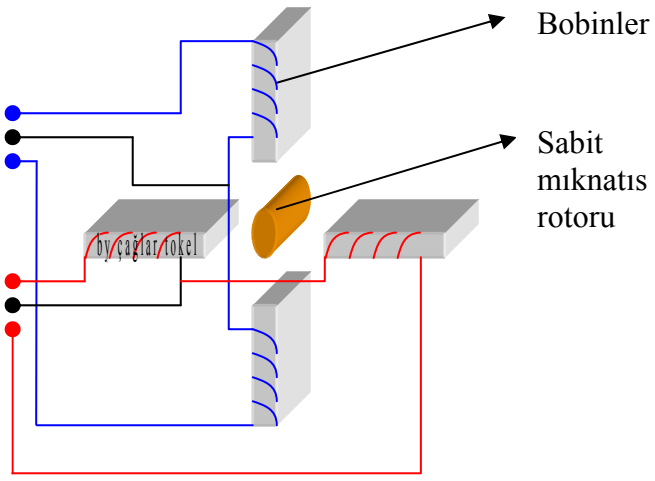
”Step motorlar uçlarına uygulanan lojik “0” ve lojik “1” bilgileri ile çalıştırıldıklarından dijital motor olarak adlandırılır….” ‘H.Şahin, A.Dayanık ve C.Altınbaşak 2006.’

Bugün bir çok uygulama alanı olan step motorlar, robotlarda, floppy disklerde yazıcılarda ve cnc gibi ileri teknoloji ürünü aletlerde kullanım alanına sahiptir. Step motorların içindeki sargıların sayısı artırıldıkça adım atabilme becerisi artar yani birim açısal dönüşteki açı azalır. 8 ayrı sargısı olan 50 parçalı bir adım motoru bir devirde 50*8 = 400 adım atabilir bu durumda açısal olarak 360/400 = 0,9 derece atabilme kabiliyetine sahiptir. Bu hassasiyet daha da arttırılabilmekte olup tork kayıpları sebebiyle tercih edilmemektedirler. Yüksek seviyeli bir dil olan Jal (just another language) ile yazdığım ve Ayyıldız (2006) ‘ın çalışmalarında da benzeri bulunabilecek programda step motoru 2 adet buton ile ileri ve geri döndürdüm. Elimizdeki step motorlar üstte anlattığım örnekteki kadar becerikli motorlar değillerdi. 7,2 derece dönüş kabiliyetli ve haliyle hassasiyeti pekte yüksek olmayan bir adım motoru olup genelde piyasada sıklıkla ve kolaylıkla bulunabilen M11 serisi bir step motordu.Adım kontrolü gerçekleştirdiğimiz bu step motorun adım kaçırması yani sürekli aynı açıyı yakalayamaması büyük bir eksisi.Ama gözlemler sonucunda yapmış olduğu bir hatayı bir sonraki adıma taşımaması artı bir özellik.Elimizdeki step motoru sürmek için bir adet 9V pil, pic voltajını 5V seviyesine çekmek için voltaj regülatörü devresi ve bir adet pic16f84 kullanıldı.Bunun yanında sürücü devresi için çoklukla önerilen piyasa da kolay bulunan ULN2803 , L293D veya bizim kullandığımız ULN2003 sürücü entegresi iş görmektedir. ULN 2803 ile ULN 2003’ün mimarisi ve yapısı aynı olup İzmir de ULN2003 edinmek daha kolaydır. Elimdeki versiyonu içerisinde ULN2003 olmadığından aynı mimarideki ULN2803 tasarıma konulmuştur. ULN2003 entegresi sahip olduğu entegre paketinin içinde (7) yedi adet darlington

transistör dizisi barındıran yüksek voltaj ve yüksek akım çeken içindeki darlington bağlantılı transistor dizisi sayesinde giriş pinine lojik 1 geldiğinde çıkış pinini lojik 0(toprak) yapan bir entegredir. Her bir kanal 500mA akım çekip 600mA e kadar akıma dayanabilmektedir. Uygulamada bu akımlar denenmiş ve bu sonuç tecrübeyle sabitlenmiştir.

Proje kapsamında step motor zayıf pozisyon kabiliyeti ve yarattığı pozisyon kayması sebebiyle tercih edilmemiş olup devresi kurulup küçük bir program yazılıp çalıştırılmıştır. Kullanılan adım motoru 6 kablolu unipolar bir motordur bu demektir ki 2 adet kablo ortak olup geri kalan 4 kablo sürücü devre kurulduktan sonra birkaç deneme yapılıp doğru kablo bağlantı sırası bulunabilmiştir.
İlk başlarda yapılan yanlış bağlantı sebebiyle adım motoru titreme ya da bir ileri bir geri şeklinde istenmeyen hareketler yapmıştır.6 adet kablo sürücü devresine doğru bir şekilde bağlandığında ise butonlara basıldığında istenen hareket gözlemlenmiş ama maalesef pozisyon almadaki becerisi adım motoru tatmin edici olmaktan uzaklaştırmıştır.

Her yönüyle servo motorun gerisinde kalan step motor daha basit ve hassasiyet gerektirmeyen uygulamalarda kullanılabilir. Step motorun ortak bağlantı kablosu kendi takımındaki diğer 2 kablo ile birlikte avometrenin direnç kademesine getirildikten sonra ölçüm yapılarak bulunmuştur.



Şekil 1.1 Unipolar bir step motorun şeması

Bu ortak kabloyu şu yolu izleyerek bulabiliriz. Üstte şemada gösterdiğim gibi unipolar 6 kablolu bir adım motoru vardır. Görüldüğü üzere 4 adet bobin vardır.
Ortak uçlar ise bu bobinlerin ortasından çıkmaktadır (siyah renkli kablolar) bu da demektir ki direnç ölçümü yaptığımda ortak uç olan kablo ile aynı parça üstündeki diğer iki kablo arasındaki direnci ölçüldüğümde ortak uç ile diğer kablonun arasındaki direnç diğer iki kablonun kendi arasındaki ölçülen direncin yarısı kadar olmalıdır. Bu şekilde diğer parçadaki ortak uçta bulunup bu iki ortak uç birlikte 12v ile beslenmelidir. Besleme için ilk önce elimdeki 9V'luk pilleri denedim. Bu şekilde bir besleme ile sonuç alamayınca sorunun besleme voltajından olduğu sonucuna vardım ve motoru 12V ile beslemeye karar verdim. Bunun üzerine 12V'luk gerilim PC'nin güç kablosundan sağlanmıştır.

Bir PC içindeki güç kablosundaki siyah renkli kabloda 0V kırmızı renkli kabloda 5V ve sarı renkli kablo da ise 12V vardır. Biz de gerilimimizi birer krokodil yardımıyla buradan sağladım. Yani güç kaynağı olarak sıradan 300W'lık bir güç kaynağı alınıp (robotta da kullanılmıştır) kullanılabilir. Tabi ki ULN2003 sürücü entegresi ve adım motorun gücü PC den sağlanırken Pic için gerekli gerilim 9V'luk bir adet pil ve voltaj regülatörü kullanılarak sağlanmıştır

JAL ile yazdığımız kod ile adım motoru A0'daki butona basınca geri A1 deki butona basınca ileriye adım atmasını sağlamak istemekteyiz. Jal kaynak kütüphanelerinin yeterliliği ücretsiz yazılımı açık kaynak kodu ile geliştirilebilirliği ile tercih edilen derleyicilerden biri olup pascal ve c temel alınan bir programlama dili olması ve c ya da pascal bilen biri kolaylıkla jal'a alışabilir.

Ben yüksek lisansım boyunca servo motor tahrikli bir robot manipulatör yapabilmek için önce pic ile başka projeler yaptım, çizgi izleyen robot vb. Bunların bir kısmında pic basic pro, bir kısmında c ve assembly dilini kullandım. Her dilin kendi içinde avantajları ve dezavantajları mevcut olduğundan, içinde hassas operasyonlar yapılabilabilen ve pic'e hakim olunabilmesi açısından assembly (makine dili) dilini program içinde kullanmaya izin veren bir dilin yeterli olduğunu düşünüyorum.

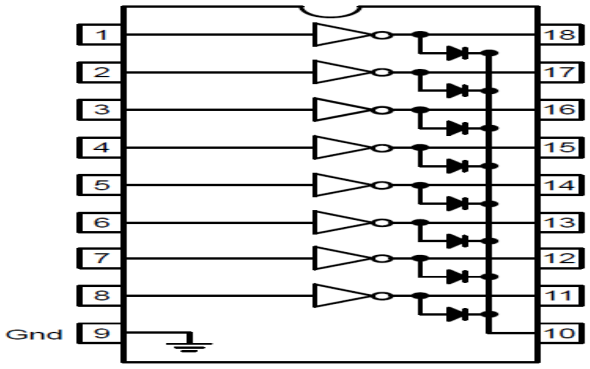
Jal öğrenmesi ve kullanması keyifli olması kod yazma ile uğraşmış bir kişi için getirdiği kolaylıklar sebebiyle ve bahsettiğim kod içinde assembly diline de izin vermesi sebebiyle robotumuzun kontrolünde son olarak karar kıldığım dil olmuştur. Aynı zamanda prosedür ve fonksiyon tanımlamamıza imkan veren bu dil tanımladığımız bu prosedür ve fonksiyonları program içinde kullanmamıza imkan vermektedir.

Adım motor kodumuzda geri için ‘backwards’ ileri için ise ‘forwards’ adında iki prosedür tanımlanıp bu prosedür çağırıldığında yapması gereken işlem ‘prosedür end prosedür’ aralığında yazılan kod ile sağlanmıştır. Gerektiği yerlerde komutların açıklaması yapılacak olup mikroişlemci konusunda piyasada ve internette sayısız doküman bulunabilecek olması ve konunun dağılmaması sebebiyle ayrıntıya yeri geldiği sürece girilecektir. Jal’ın yaratıcısı aynı zamanda faydalandığım kaynaklardan biri de olmuş olup Wouter Van Ooijendir.

1.1.1.1 Adım Motorun İleri Geri Adım Sağlayan Kod

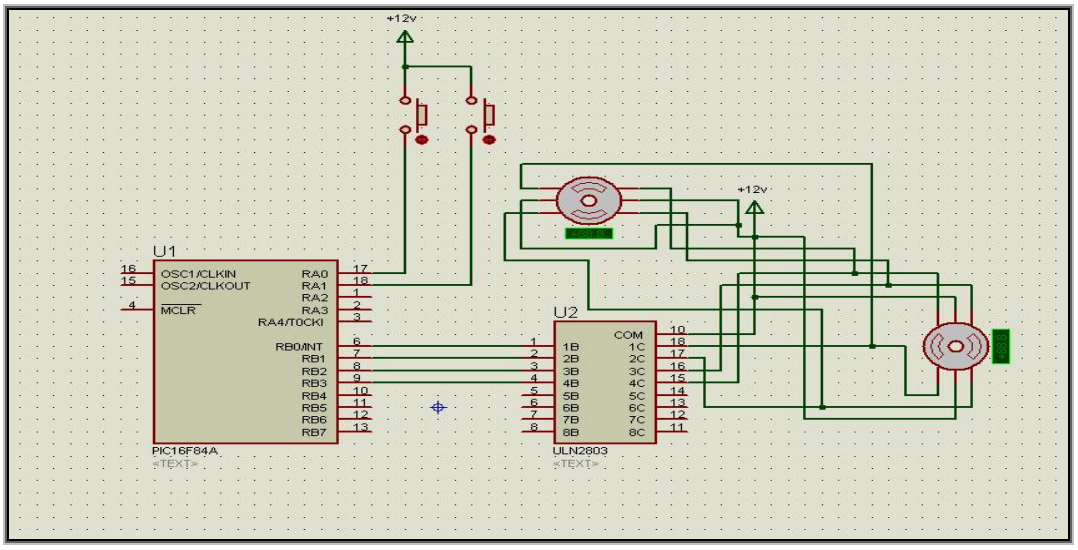
```
include 16f84_4              -- kütüphane dosyalarını
include jlib                 -- çağır

port_a_direction = all_input          -- a portunun tüm pinlerini giriş yap
port_b_direction = all_output         -- b portunun tüm pinlerini çıkış yap

var byte x = 0x01                      -- byte tipinde x değişkeni tanımla
                                       -- ve içine 0000 0001 değerini ata
procedure step_backwards ( byte in out x )is
   var bit c at x : 0    -- “c”  x tutucusunun 0. biti olsun
  if c then              -- eger c biti yani x'in 4.biti yani ***c ****  1 olursa
   x = x >> 1 | 0b_1000       -- bitsel olarak x’in içeriğini 1 bit sağa kaydır ve
                              --    0b_1000 ile yani 0000 1000 ile or’la (veya)

else  x = x >> 1               -- değilse sadece bitsel olarak 1 bit sağa kaydır.
```

```
 end if
 x = x & 0b_1111                -- bu satır ile c 1 olana dek 0000 1111 ile ve'liyoruz
end procedure

procedure step_forward ( byte in out x )is
var bit b at x : 4    ---- "b"  x tutucusunun 4. biti olsun
x = x << 1          -- bitsel olarak x in içeriğini 1 bir sola kaydır.
if b then           -- eger b biti yani x'in 4.biti  ***b ****  1 olursa
  x = x | 0b_0001      ( x i or'layalım..ve 0001 olsun )
end if
  x = x & 0b_1111
end procedure

forever loop          -- forever loop end loop içini sürekli olarak yap

  if pin_a0  ==  high then step_backwards ( x ) end if   -- a0 pini highsa bu işi yap
  if pin_a1  ==  high then step_forward ( x )  end if      -- a1 pini highsa bu işi yap
  port_b = x         -- x değişkeni içeriğini portb'ye yaz.
  delay_100ms ( 5 )          -- 500ms bekle

end loop
```



Şekil 1.2 ULN2803 entegresi

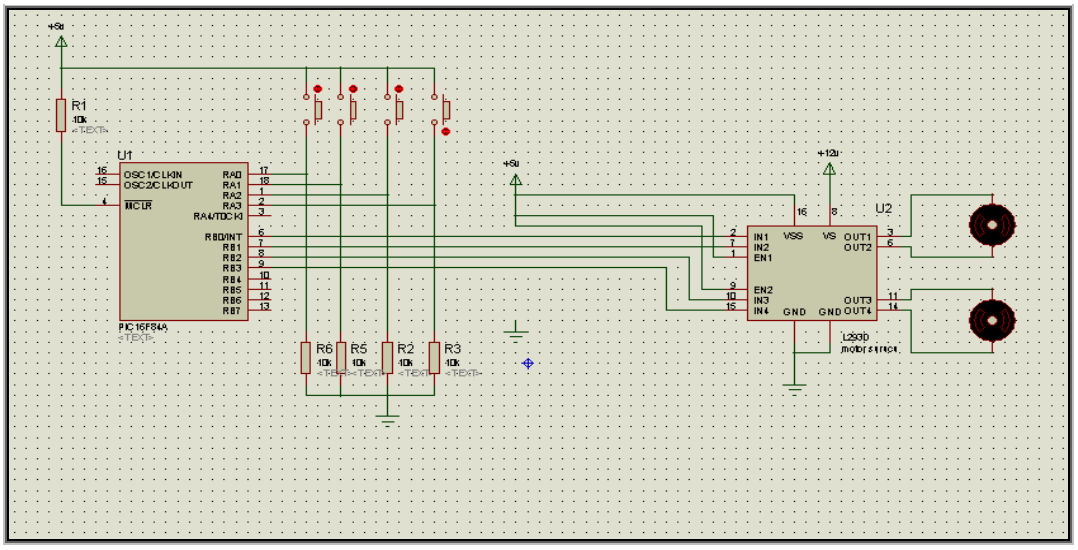
ULN2803 8 adet darlington transistore sahip ve tersleme (NOT) işlemi yapması için özelleşmiş bir entegredir.



Şekil 1.3 Adım motoru kontrol etmek için kurulan devre şeması

1.1.2.DC Motorlar (Doğru Akım Motorları)

Bu bölümde butonlarla DC motora ileri geri kontrol yapılmıştır. Bu devrede DC motora PWM verilmemiştir sadece basit bir kod ile belli bir süre butona basınca iki motoru bulunan bir çizgi izleyen robotun tekerleklerine yön veren bir benzeşme yapılmıştır.A0 pinindeki butona basınca 10 saniye ileri ve A1 pinindeki butona basınca da 10 saniye geri A2 Pinindeki butona basınca 10 saniye sağa A3 pinindeki butona basınca 10 saniye sola gitmesini sağlayan bir kod yazılmıştır ve bu devre de önce Proteus Isiste simule edilmiş sonra protoboard da denenmiştir. 4 adet buton 10K' lık dirençlerle giriş pinleri yapılmış olan B portu pinlerine bağlanmış butonlar aracılığıyla, L293D sürücü entegresi ile dc motor kontrolü yapılmıştır.



Şekil 1.4 İki adet Dc motoru pic16f84A kullanarak sürmeye yarayan devre şeması

Simulasyon ekranında görülen iki adet motor ile bir hareketli araç kontrolü yapılabilmektedir. Kodlar çok açık olduğundan burada açıklama satırları konmasına gerek görülmemiştir. Pic mikroişlemcisi olarak 16f84 yeterli görülmüş ve kullanılmıştır.

```
include 16f84_4
include jlib
port_b_direction = all_output
port_a_direction = all_input
port_b = 0

procedure ileri is
pin_b0 = high   -- 1. motor ileri
pin_b1 = low
pin_b2 = low    -- 2. motor ileri
pin_b3 = high
end procedure

procedure geri is
pin_b0 = low    -- 1. motor geri
pin_b1 = high
pin_b2 = high   -- 2. motor geri
pin_b3 = low
end procedure

procedure sol is
pin_b0 = high   -- 1. motor ileri
pin_b1 = low
pin_b2 = high   -- 2. motor geri
pin_b3 = low
end procedure
```

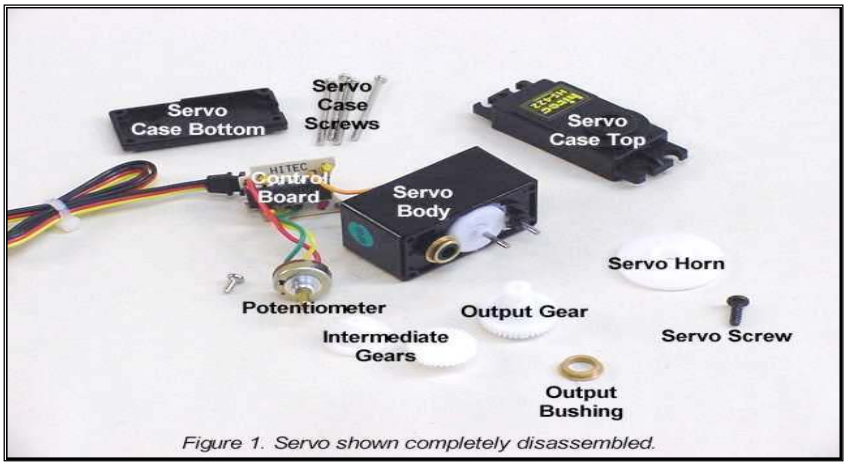
```
procedure sag is
pin_b0 = low   -- 1. motor geri
pin_b1 = high
pin_b2 = low   -- 2. motor ileri
pin_b3 = high
end procedure

procedure dur is
pin_b0 = low   -- 1. motor dur
pin_b1 = low
pin_b2 = low   -- 2. motor dur
pin_b3 = low
end procedure

forever loop
if pin_a0 == high then  -- 10 saniye ileri
ileri delay_1s(10) dur
end if

if pin_a1 == high then  -- 10 saniye geri
geri delay_1s(10)  dur
end if

if pin_a2 == high then  -- 10 saniye sağ
  sag delay_1s(10)  dur
end if

if pin_a3 == high then  -- 10 saniye sol
  sol delay_1s(10)  dur
end if

end loop
```

1.1.2 Servo Motorlar (RC servo motorlar)

Çoğu DC motor, step motor ve AC motor açık döngü prensibiyle çalışırlar yani bir geri beslemeleri yoktur. Servo motorlar ise kapalı döngü prensibine göre çalışırlar ve geri beslemeye sahiptirler. Bunun için sahip olduğu entegre kontrol sinyalini alıp harekete geçtikten sonra içindeki potansiyometre sayesinde gerçek pozisyonu öğrenmesi ile motorun gittiği konumu gitmesi gereken konuma doğru yönlendirir. Servo motorlar kompakt bir bütünün içinde birkaç adet parça içerirler.

Şekil 1.5 Futaba S3003 Rc servo motor

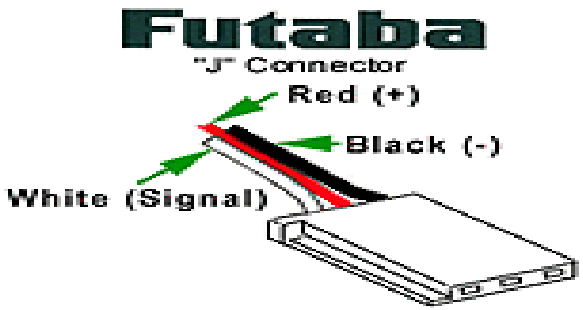
- Küçük bir DC motor
- Torku arttırmak için dişli takımı
- Elektronik şaft kontrol algılama (potansiyometre) kontrol devresi



Şekil 1.6 Robotta da kullanılan ebatlarda standart bir servo motorun içi ve bileşenler gözükmektedir

1.1.4 Rc Servo Motorun Kablo Bağlantısı

Bütün servo motorların 3 adet kablosu vardır:



Şekil1 1.7 Bir servo motor bağlantı kablo şeması

- **siyah veya kahverengi olan toprak;**.
- **Kırmızı** olan güç (~4.8-6V).
- **sarı**, **turuncu**, veya **beyaz olan ise sinyal kablosudur** (3-5V).

1.1.5 Servo Motor Voltajı (Kırmızı ve Siyah Kablolar)

Rc Servo motorlar belli bir aralıktaki gerilimde çalışabilmektedirler. Bilinen piyasada bulunan tipik bir servo **4.8V ila 6V** arasında çalışabilmektedir. Bazı daha küçük servolar daha küçük voltajlarda (mikro servolar) ve bazı servolar ise daha büyük voltajlarda da çalışabilmektedir.

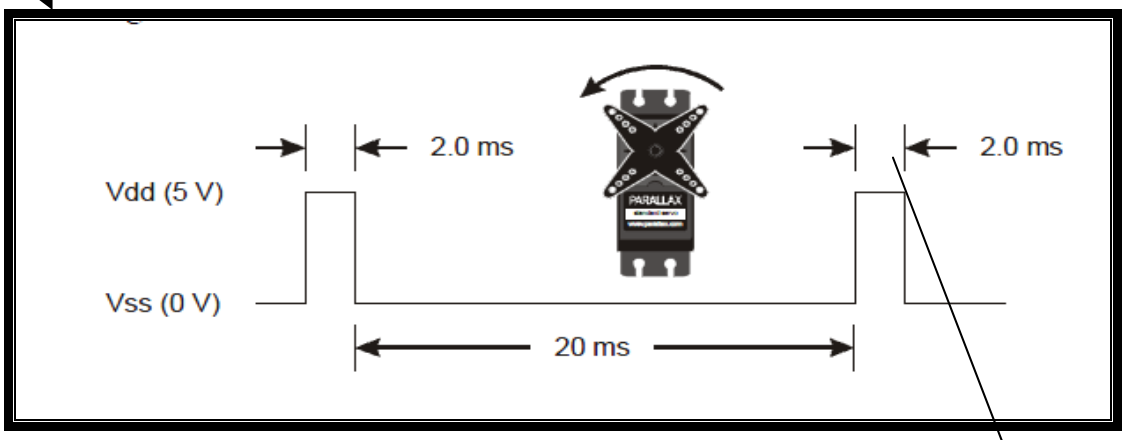
Bunun sebebi ise çoğu microkontrolörün ve RC(radio controlled) alıcılar bu voltaj aralığında çalışabilmesidir. Bir servo motorda azami torku elde edebilmek için 6V ile çalıştırmakta fayda vardır çünkü servo motorlar içinde bir adet dc motor vardır ve bu motorlar yüksek torklara yüksek voltaj değerlerinde ulaşır. Bu sebepledir ki uygulayabileceğimiz maksimum voltajı sağlamamızda fayda vardır. Proje de 300W lık 5V da 19A verebilen bir güç kaynağı kullanılmıştır. Bir PC güç kaynağında 6V olmadığından proje kapsamında aslında servo motorlardan alınabilecek en yüksek torku almamaktayız. Ancak istendiği taktirde 6V'da yeterli akım verebilen bir güç kaynağı kullanarak bu değerlere ulaşabiliriz.

Robot kolun tasarımında daha sonra ayrıntıyla açıklanacak olan 6 adet servo motor kullanılmıştır.

Bunlar;	TORK DEĞERLERİ
1 adet towerpro mg995 servo motordur.	12,9 kg/cm
2 adet futaba S3305 servo motor	8,9 kg/cm
3 adet futaba standard S3003 servo motor	3,9 kg/cm

Şekil 1.8 Robotun aktüatorleri (hareket organları) olarak kullanılan motorlar

* Parallax Rc servo motor üretici firmasından alınmıştır.



Şekil 1.9 PWM ile Rc Servo Motor Kontrolü

2ms'lik darbe (pulse)

Bu gösterim de görülen PWM(pulse with modulation) ile bir servo motorun nasıl sürüldüğü çok net görülmektedir. 20ms'lik peryot ile sürekli olarak servo motor için gerekli darbe (pulse) olan 2ms servo motora gönderilir.

Yalnız burada bir mikroişlemciyi programlarken altta gösterdiğim gibi bir kod yazdığımızı düşünelim.

Pin_ao =high (servo motoru bağladığımız pini high yapalım 5v)

Delay_10us (200) (2 ms süre boyunca bekle)

Pin_ao = low (servo motoru bağladığımız pini low yapalım 0v)

Delay_1ms (20) (20 ms süre boyunca bekle)

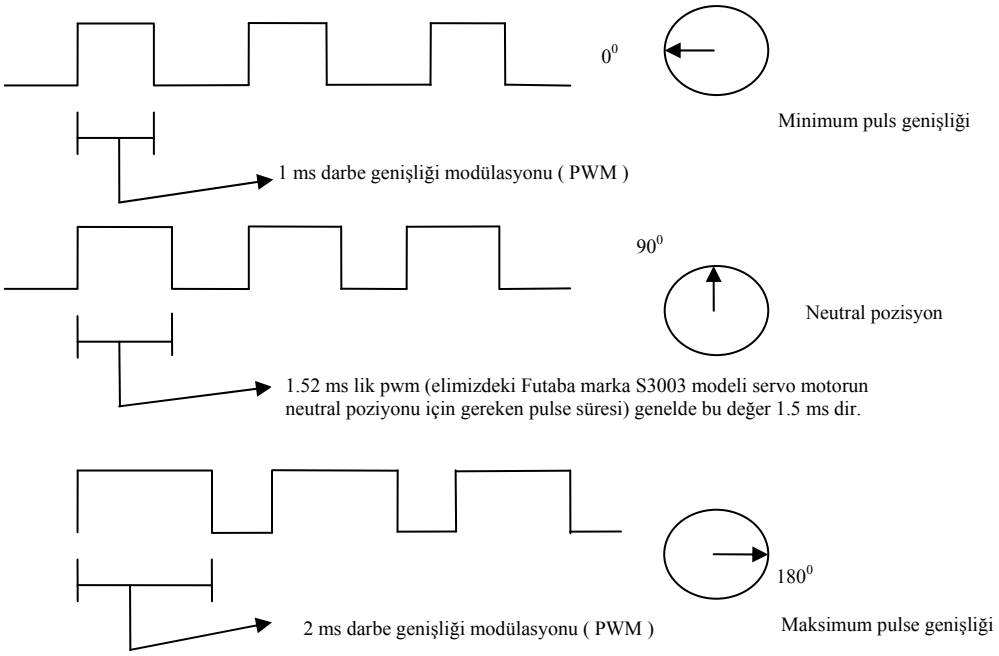
Bu kodda robot kol için yazılan programda da ilerleyen bölümlerde görüleceği gibi değişiklik yapmak gerebilir. Çünkü küçükte olsa servo motorlar birbirinden farklı pwm sürelerine ihtiyaç duymaktadırlar. Aynı zamanda aynı prosedür içinde birden fazla servo motor sürülmek istenirse her satırda da bir gecikme olacağından dolayı her zaman 20ms lik bekleme yapılırsa bu durumda servo motor hareket etmez tecrübeyle bu değerler bulunabilir.

Ben kullandığım 3 ayrı servo motoru üç ayrı skala'da sürmek durumunda kaldım çünkü her biri farklı pwm'e ihtiyaç duymaktaydılar.

Not: Tabi ki bu kodun çalışması için TRIS kaydedicisini (register) başa eklemek gerekir. Bu picin portundaki pinlerin giriş mi çıkış mı olacağını seçen register'dır. Aynı zamanda ADCON1 registerı da "0" yapılarak giriş olarak seçtiğimiz A portunun analog dijital dönüştürme özelliğini de kapamak gerekir.Eğer bu işlemler yapılmazsa bu kod çalışmaz.

Siyah ve kırmızı kablolar servo için gerekli gücü sağlarken sinyal kablosu servo motoru kumanda etmek için kullanılan kablodur.

Genel olarak bir servo motoru kontrol etmek için servo motorumuza basitçe belli bir dalga boyunda bir "***kare dalga***" göndeririz. Bu şekilde servomuz belli bir pozisyona gider.



Şekil 1.10 PWM yöntemi ile Rc servo motorun son, orta ve en baş pozisyona yönlendirilmesi

1.1.6 PWM

Gönderilen sinyalin periyodunun süresi sabit kalmak koşulu ile iş (duty) süresinin arttırılıp azaltılmasıyla high ve low süresinin değişmesi ve sonuç itibariyle bu işleme PWM (pulse with modulation) denilmektedir. PWM uygulanışında seri iletişim için yazdığım arayüzde de servo motorumuza 0 ile 255 arasında belirlediğimiz değerler gönderebiliriz. Pic'lerin çoğu 8 bitlik olduğundan zaten 255'den daha büyük bir değer istesekte gönderemeyiz. Bu süre decimal 255, %100 iş çevrimi(cycle) yaratmak için decimal 0, %0 iş çevrimi(cycle) için ve decimal 127, %50 iş çevrimi(cycle) yaratmak içindir.

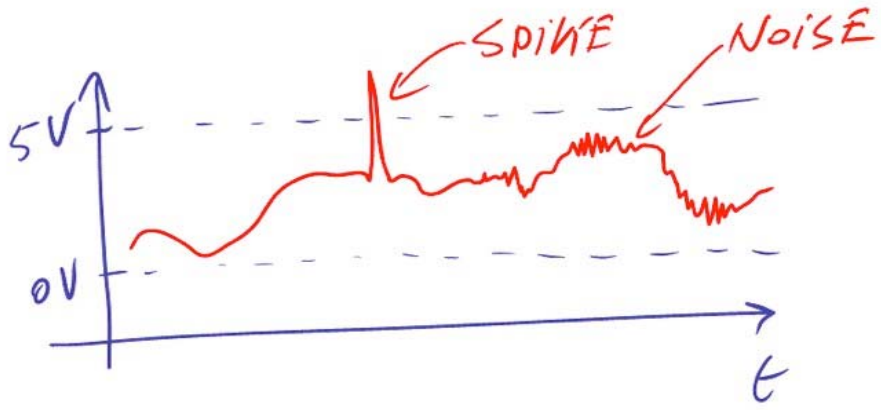
- 255 = 1111 1111 maksimum 180 derece
- 127 = 0111 1111 (center) 90 derece
- 0 = 0000 0000 minimum 0 derece

Bizim projede kullanacağımız servo motorlar 180 derece dönme yeteneğine sahiptirler. Bu açı tutma ve bırakma (pick and place) işlemi için yeterli olmaktadır. Robot senkron servo motorlar aracılığıyla diğer istikamete döndüğünde de 180 derece tarayacağından 360 dereceye de ulaşabilme imkanımız vardır.

Bir servo motor istek doğrultusunda üstünde yapılacak birkaç küçük işlem ile devamlı dönen bir servo haline getirilebilir. Bunun için yapılması gereken işlem çok basit gibi görünür yalnız bu işlemdeki handikap geri dönüşü olmamasıdır. Bizim bir robot kol yaptığımız düşünülürse devamlı dönen bir servo motora ihtiyaç duymayacağımız ortadadır.

Üstte verilen 255, 127 ve 0 girdileri visual basic programımızda "baş" "merkez" ve "son" olarak isimlendirilmiş olup farklı özellikteki motorlarda küçük değişiklikler yapılarak kullanılabilir.

Bu haliyle servo motorumuzun seri porttan iletişimini sağlarken birtakım sorunlar ortaya çıkmış olup bu hataların sebebinin noise(gürültü) faktörü olduğu düşünülmüştür.



Şekil 1.11 Zamanla sürekli değişen bir analog sinyal

Not: Biz robot kol için bir sensör'den bilgi okuyup işlem yaptırmadığımız için bu tez kapsamında analog sinyal için portlarımızın seçimini kaldırdık ve giriş bacaklarımızı dijital giriş (digital input) olarak yönlendirdik.
Fakat bu kısımda Şekil.1.10'daki analog sinyal grafiğinden ve buradaki gürültü (noise) olayından da bahsetme gereği duyduk.

Analog sinyal'ler zamanın fonksiyonu olarak sürekli olarak değişen sinyallerdir. Genellikle analog sinyaller belli bir voltaj aralığında (0 – 5V) azalmak üzerine

uygun duruma getirilmişlerdir. Bu voltaj aralığının dışında kalan işlemler gürültü ve parazitin olmadığından emin olmak için uygulanırlar.

Bir sinyal ölçtüğünüzde ve bu sinyal sabit olması beklenirken bu sinyalin zamanla değiştiğini gözlemliyorsanız bu muhtemelen bir **gürültüdür**. (noise) Bunun yanında **parazit** ise başka bir sinyalin varlığına işaret eder.

Bir analog sinyalle bir iş yapabilmek için onu mikrodenetleyici için uygun bir forma getirmemiz gerekir. Yani onu dijital bir sinyale dönüştürmeliyiz.

Bu bahsedilen gürültüyü önlemek ve sinyali bozması olası bu çarpışmayı önlemek için servonun sinyal hattına 1 adet kapasitör bağlanarak bir çözüme gidildiğinde bu gürültünün engellendiği görülmüştür.

PIC 16F877A mikro denetleyicisinin bacaklarını (pin) seri portun Rx ve Tx girişlerine doğrudan girmek yerine arada RS232 seri iletişim protokolünü kullanmak verilerin hatasız iletimi açısından zorunludur. Bunun için MAX232 entegresi kullanılabilir.

MAX232 entegresi -12, +12V seviyesindeki seri port sinyallerini TTL (+5, 0V) seviyesine veya TTL seviyesini -12 V, +12 V sinyallerine çevirir.
Aşağıda tezimizde kullanılan bir MAX232 entegresi görülmektedir.

Şekil 1.12 Max232
entegresi pin tablosu

1.1.7 Seri Port

Yine isminden de anlaşılabileceği gibi bilgilerin seri olarak aktarıldığı bir porttur. Veriler bir hat üzerinden belli zaman aralıklarında belli bir sıra ile yollanarak karşıya bit bit ulaştırılır.

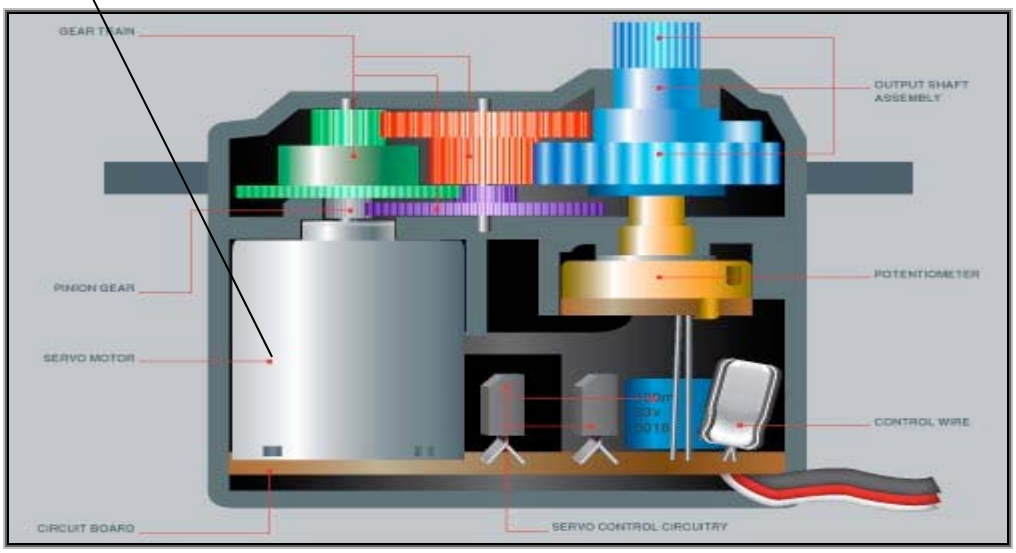
Örneğin Pc'den pic'e "a" karakteri yollayacağımız zaman bu seri porttan onluk olarak "97" o da 0110 0001 olarak gönderilir.

Burada zamanlama yani frekans uyumu çok önemli olduğundan dolayı protokol kavramı oluşmuştur.

Aynı zamanda çift yönlü iletişim kurabilmesi ile seri iletişim, paralel iletişimin önüne geçmiştir. Seri port kullanılarak veri kaybı olmadan çok daha uzun mesafelere iletim yapılabilir. Bunun daha az kablo kullanarak yapılması seri portun önemli bir avantajıdır.

Daha sonraki bölümlerde senkron ve asenkron iletişim ve USART (universal senkron asenkron reciever transmitter) örneklerle açıklanacaktır.(bknz 3.1.3, 3.1.4, 4.1.15)

* Bildiğimiz anlamda bir Dc motordur. Bütün olan yapıya servo motor denir



Şekil 1.13 Tipik bir Rc servo motorun kesiti alınmış haldeki iç yapısı

1.1.8 Basitçe PWM Yaratmak

```
Start:
        High PORTA.0    ‘PORTA nin 0 no’lu pinini high ( lojik 1) yapalım
        PAUSE 10        ’10 ms bekleyelim
        Low PORTA.0     ‘PORTA nin 0 no’lu pinini low  ( lojik 0 ) yapalım
        PAUSE 10        ’10 ms bekleyelim
Goto start
```

Bu komut dizisiyle 20 ms periyotlu ve (1 / 20) * 1000 = 50 Hz frekansına sahip bir pwm elde etmiş oluruz. Şimdi iş (duty) süresini değiştirelim. Periyot’umuz yine aynı kalacak, yani 20 ms olacak şimdi 18ms high 2 ms low yapalım.

```
Pic basic                       Jal
Start:                  forever loop
        High PORTA.0    pin_a0 = high    ‘PORTA nin 0 no’lu pin’ini high yap
        PAUSE 18        delay_1ms(18)    ‘18 ms bekle
        Low PORTA.0     pin_a0 = low     ‘PORTA nin 0 no’lu pin’ini low yap
        PAUSE 2         delay_1ms(18)    ‘2 ms bekle
Goto start              end loop
```

Bu durumda duty değeri %50 den 18/20 ye yani %90 e çıkartmış olduk.

1.1.9 Servo Motorun Akımı

Servo akımı DC motorla eşdeğer bir akım çeker, ancak buna ilave olarak servoların içinde normalde enkoder kullanılmazsa çok zor olarak tespit edilebilen bir geri besleme kontrol sistemine sahibizdir. Eğer Dc motorumuz istenilen açıda değilse aniden istenen açıya ulaşmak için büyük miktarlarda akım çekecektir. Servo motorun belli bir açıda o pozisyonda sabit kalması için yüksüz iken ve bir ağırlıkla yüklü iken çektiği akım aynı değildir. Ama servo motorların çektiği akım beklenenin aksine yüklenen ağırlıkla doğru orantılı olarak (lineer) artmaz. Yani servonun çektiği akım önceden kestirilemez.

1.1.10 Servo Motorun Hızı

Servo motorların dönme açıları ve bunun için gereken zaman gibi özellikleri her servo için yakın olup datasheet'lerinden bu bilgiler kolaylıkla edinilebilir. Bir servo motorun hızı o servo'nun belli bir miktar hareket etmesi için gereken zamandır ve bu dönel hareket elimizdeki Futaba S3003 model servo motorda 45 derece / 0.17 sn' dir. (Tabi ki yüksüz iken). Bu servo motor 180^0'lik bir açısal dönme kabiliyetine sahiptir. Haliyle 180^0 tarama süresi 0,68 olduğu hesaplanabilir.

1.1.11 Verim ve Gürültü

Gürültü (noise) ve kontrol devresinin gereksinimleri sebebiyle servoların verimleri kontrol elemanı olmayan sıradan DC motorlara göre daha azdır. Boşta iken servoların kontrol devreleri 5-8mA civarında bir akım çeker. Gürültü (noise) esnasında servolar bir sabit pozisyonlama anında çektiği akımın 3 misli akım çekerken, yine aynı gürültü, servo dönme hareketi yaparken çektiği akımın 2 katına yakın akım çeker. Gürültü servoların verimsizliğindeki en büyük etkendir. Bundan kurtulmak gerekir. Servo motorda bir titreme oluyorsa bu servonun hızlı bir şekilde çakışma sonucu pozisyonunun iki açı arasında atladığını gösterir. Bu çakışmaya sebep olan şey,sinyal kablosunun uzun bir antenden farkı olmaması ,yabancı ve istenmeyen sinyalleri de kabul edebilme yeteneğinin olması ve aldığı bu sinyalleri doğal olarak servomuza bir komut olarak göndermesidir.Üzerinde çalıştığımız servo olan futaba S3003 model servonun pc ile seri iletişiminde bizde ilk başta buna benzer bir durum yaşadık.Genel olarak bu sinyal çakışmaları yakındaki diğer servolara giden sinyallerin karışması sonucu olmakta ya da bu servoların kablolaması sonucu hatların birbirine çok yakın olmasından olmaktadır.Bu sorunu çözmek için sinyal kablosunu kısa tutmak ve servomuza ek uzatma kablosu takmama yoluna gidilebilir.Ben robotta bütün motorlar için ek uzatma kablosu kullanmama rağmen bu sorunları düzgün ve temiz bağlantılar sayesinde yaşamadım.Bir diğer çözüm ise sinyal kablolarını birbirine sarmaktır.

Sonuç: 1.kısmın sonunda görülmüştür ki bu boyutlarda bir robot için kullanılabilecek en iyi motor bir servo motordur. Bunda geri beslemesinin olması, düşük voltajlarda yüksek tork değerleri vermesi gibi avantajları eklendiğinde robotun tasarımı için bu motorların kullanılması uygun bulunmuştur.

BÖLÜM İKİ
MEKANİK TASARIM

2.1 Robotun Mekanik Tasarımı

Piyasa da farklı amaçlar ve uzmanlıklar için kullanılan çeşitli tasarım programları mevcuttur. Ben 2002 yılından beri Solidworks kullandığımdan dolayı tasarım için bu programı seçmiş bulunuyorum.

2.1.1 Tasarım programının seçimi

Farklı kullanım alanlarıyla birlikte en çok tercih edilen tasarım programları AutoCAD, Rhinceros, Ideas, Catia, Pro engineer, Solidworks ve SpaceClaim programlarıdır. Solidworks dost canlısı arayüzü tasarım sırasında sağladığı yüksek hakimiyet, piyasada makine tasarımı yapılan firmalarda çoğunluk olarak tercih edilebilir olması gerçeği, üstüne çok sayıda kaynak ve doküman temin edilebilirliği sebebiyle tez kapsamında tercih edilen program olmuştur. Yüksek lisansım kapsamında 8 aylık bir süre profosyonel olarak makine tasarımı yapma fırsatım olduğundan ve bu süre içerisinde Solidworks 2009 kullanmam sebebiyle tasarım SW2007 ile başlamış bir takım revizyonlar da SW2009'da yapılmıştır. Solidworks ileriki versiyonlarda kaydedilen dosyaları daha önceki versiyonlarda açmaya izin vermediğinden dolayı 2 ayrı program içinde ayrı klasörlerde tasarımlar mevcuttur.

İlk olarak robotun çalışma uzayı belirlenmiş 400 mm lik yarı çapta bir çalışma uzayı uygun görülmüştür. Bu noktadan sonra tasarıma geçilmiştir. Bu noktada dünyada satılabilirliği olan bu boyuttaki robotlar incelenmiş ve robotun tasarımında şık tasarımı sebebiyle Lynxmotion 5 robotu örnek teşkil etmiştir. Projede ilk olarak kullanılacak olan Rc servo motorların seçimi üstünde duruldu. Bu doğrultuda en çok kullanılan ve piyasada saygın bir yeri olan futaba marka servo motorlar tercih edildi. Dünya da bir numaralı Rc servo motor üreticisi olmaları sebebiyle bu marka motorlar (bir diğeri de Hi-tech firmasıdır) aktüatör (hareket organı) olarak seçildi. Robot manipülatörün bir objeyi taşıdığı andaki en büyük yükü taşıyacak olan bölgesi olan robotun omuz bölgesi için iki adet eşzamanlı

çalıştırmak üzere 9kg/cm tork verebilen daha üst seviye motorlar tercih edilirken bir diğer büyük yük taşıyan dirsek bölgesine ise 13kg/cm tork veren towerpro mg995 motorları alındı. Bu motor Standart servo motorlarla (3,2kg/cm) aynı fiyatta olup buna rağmen içinde plastik yerine metal dişlilere sahip olması sebebiyle seçilmiştir. Towerpro marka motorların hassasiyetinin 'Çin malı' olması sebebiyle futaba ya da Hi-tech kadar yüksek olmadığı muhakkaktır. Ama çalışmalarım boyunca en çok korktuğum şey olan ve bir servo motorun eksi hanesinde olabilecek "overshoot" problemine rastlamadım.

2.1.2 Overshoot

Basitçe motorun gitmesi istenen pozisyona gidip geçmesi ve tekrar geri dönmesidir. Bu işlem bir geri beslemeye sahip olan bir motordan istenmez. P, I ve D kontrol projenin konusunun dağılmaması amacıyla çok ayrıntılı olarak tezde gösterilmeyecek olup gerekli incelemeler bu konu üstüne yazılmış kitaplardan yapılabilir. Ancak kısaca PI kontrolden bahsedilmesi bu bölümün kavranılması için uygun görülmüştür.

Servo bir sistemde olan PI (oransal integral) kontrol ile P hızlı kontrol imkanı verirken I ise hatayı tamamen yok edici özelliğe sahiptir.

Yani bir nevi P ile kaba ayar yapar I ile hassas ayar yapılmaktadır.

Motorun referans torku $T^* = Kp\,(t)\,e\,(t) + K_i(t) \int e\,(t)\,dt$ formülünden bulunur.

K_p oransal kazanç olup maksimium ve minimum değerleri arasında değişir.

K_i ise integral kazançtır.

T: birim zamandır.

e (t) : hız hatasıdır.

Kp maks: Hızı artırmak için kullanılan kazançtır.

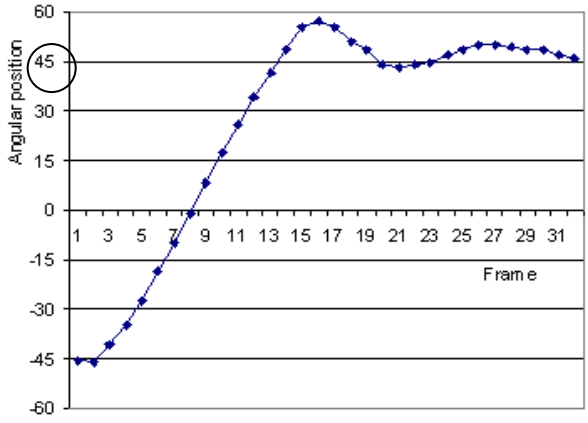
Kp min: Salınımları ve overshoot'u azaltan kazançtır.

$e(t) = [1-\alpha(t)] K_{maks} / Ki(t)$ formülünden bulunan hatadır.

Kararlı durumda e (t) *küçük olduğunda* kararlı durum hatasının üstesinden gelmek için *büyük* integral kazanç kullanılır.

Kararlı durumda e (t) *büyük olduğunda* kararlı durum hatasının üstesinden gelmek için *küçük* bir integral kazanç kullanılır.

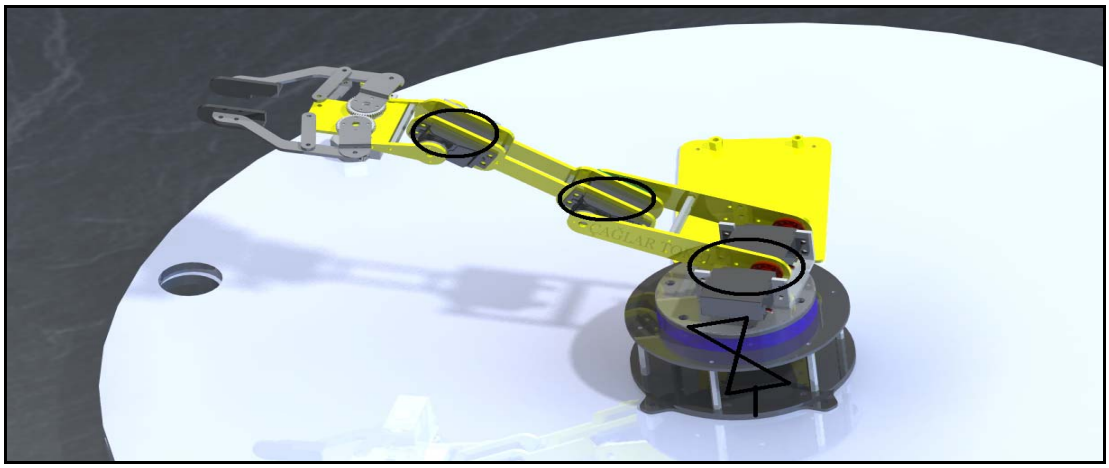
Ki (maks) = 0 olması durumu ise tahmin edilemeyen salınımların ve overshoot'un üstesinden gelmek için kullanılır.



Şekil 2.1 Overshoot oluşumu ile gitmesi istenen pozisyonu önce kaçıran bir motorun adım adım pozisyonunu nasıl bulduğunu gösteren grafik
45^0 motorun gitmesini istediğimiz pozisyon

2.1.3 Serbestlik Derecesi

Denavit-Hartenberg metodu robot kolların serbest cisim diyagramını (SCD) göstermek için kabul edilmiş bir metoddur. Bu metod gripper'ın (tutucu) hareketini bir serbestlik derecesi olarak kabul etmez. Gripper'a "end effector" denir. Bu durumda robotumuz 4 serbestlik derecelidir. Eksen sayısı: 4 + gripper olarak değerlendirilir.



Şekil 2.2 Robotumuzun SCD'sinin (serbest cisim diyagramı) gösterimi

Tez için yapılan robotun resimde bu metoda göre olan 3 adet elips ve ikişer adet ters üçgen sembolü ile gösterilmiş tam 4 ekseni olduğu görülmektedir.

Bir uzvun yapabileceği sadece iki hareket ve bu hareketlerin olabileceği de x,y,z olmak üzere 3 eksen vardır. Bu iki hareket dönme ve öteleme hareketidir. Bu öteleme ve dönme hareketlerinden ortaya çıkan serbestlik derecelerinin de limitleri vardır. Bunun sebebi uzuvların 360^0 dönme olanaklarının olmayışıdır.
Bizim robotumuzun limiti ise servo motorlarımızın 180^0 lik dönüş kabiliyetleridir.

2.1.4 Çalışma Uzayı: Robotun çalışma uzayı, bir diğer deyişe göre robotun ulaşabileceği uzay robotun tutucusunun (gripper) ulaşabileceği en uzak yerlerdir. Çalışma uzayı dönme ve öteleme serbestlik derecelerinin limitlerine, robot kol uzunluklarına, tasarıma ve alınacak objenin alınacağı açıya bağlıdır.

2.1.5 İleri Kinematik ve Ters Kinematik

”Robot kinematik denklemleriyle robotun hız tork ve ivme analizi yapılır. Her bir robot ekseninin konumu, bir öncekine veya bir sonrakine göre ifade edilir. Arka arkaya oluşturulan bu ilişkiye kinematik zincir denilir. Bu ilişkiyi oluşturan ifadeler, robotun konum ve yönelim bilgisini içeren 4 x 4 homojen dönüşüm matrislerinden (transformasyon matrisi) oluşturulur. Her bir eklem için bir homojen dönüşüm matrisi oluşturulur. Oluşturulan bu matrislerin sayısını, robotun serbestlik derecesi belirler. Üç boyutlu uzayda herhangi bir noktaya ulaşmak için 6 serbestlik derecesi yeterlidir. Buna karşın, serbestlik derecesi altıdan fazla olan robotlarda artıklık (redundancy) durumu oluşur.” (Bingöl ve Küçük, 2005)

2.1.6 Robotun Tasarımı

Robotun tasarımı yapılmadan önce ilk etapta daha önce yapılmış benzer boyuttaki 4, 5, 6 eksenli robotlar incelenmiş olup özellikle www.lynxmotion.com tarafından satışı da gerçekleştirilen ürün üstünde durulmuştur. Robotun dış görünüşü büyük ölçüde bu modele benzemektedir. Üretim teknikleri dikkate alınarak tasarım yapılmıştır. Robotun parçalarının tasarımı yapılırken dayanımı arttırma yoluna gidilmiş, robotun hareketi esnasındaki dinamik davranış sebebiyle ortaya çıkan salınımları azaltmak amacıyla yumuşak bir malzeme olan derlinden imal edilmiş sönümleme amaçlı bir parça tasarlanmıştır.

Bütün parçalar Solidworks ortamında ve üretim teknikleri dikkate alınarak tasarlanmıştır. Tez kapsamında yapılacak robot için 1050 (piyasa çeliği), T200, 6013 benzeri alüminyum parçaların üretimi yapılmayacağından freze ve torna da üretimin yapılmayacağı ortadadır. Bu yüzden cnc router ve lazerde üretim yapılacağı göz önünde bulundurularak tasarım yapılmıştır. Bir tasarım yapılacağı zaman üretimin yapılacağı makineler dikkate alınarak tasarım yapılmalıdır. Burada önemli olan birtakım ayrıntılar şunlardır. Lazerde kesilecek bir parçanın et kalınlığı 5mm ise bu parçaya açılacak delik çapı 5mm’ den küçük olmamalıdır aksi takdirde lazer yapısı itibariyle bu deliği delmeyecektir. Profosyonel açıdan ise düşünülecek olursa lazerde

bir parça için en pahalı olan şey delik delme işlemidir çünkü kesime göre uzun zaman alan bir işlemdir.

Lazerin kesim kabiliyeti çok hassastır. Cnc router'a göre sahip olduğu **backlash** sıfıra yakın olduğundan (yazılımla giderilebilir) çok büyük bir hassasiyeti vardır. Lazerde bir parça kesilecek ise tasarımı yaparken Solidworks'te sac şekillendirme parametrelerinde varsayılan olan ayarlarda değişiklik yapmak gerekmektedir. Ayrıca yatak genişliği de dikkat edilmesi gereken bir unsurdur. Aşağıdaki grafikteki kullanarak doğru bir tasarım yapılabilir aksi takdirde istenmeyen sonuçlarla karşılaşılacaktır.

Tablo 2.1 Solidworks için sac şekillendirme parametreleri

Solidworks için saç şekillendirme parametreleri

Saç Kalınlığı (mm)	Yatak Genişliği (mm)	Büküm Yarıçapı (mm)	K-Faktor
0.8 - 2	10	1.64	0.39
2.5-3	16	2.26	0.36
4	30	5.5	0.35
5	40	5.5	
6 ve üstü	50	Kalınlık+Uzama Miktarı	

K-Faktor değeri 0.35 ile 0.40 arasında değişmelidir.
6 mm için 0.35
1 mm için 0.4
Ara değerler için ortalama bir değer alınabilinir.

Bu değerler paslanmaz ve siyah saç için ortak kullanılabilinir.

Backlash: Basitçe dişli ya da vidalı bir sürücü sistemindeki boşluktur. Eğer bu boşluk giderilmez ise bu boşluktan doğan hatalar ortaya çıkacaktır. Bu boşlukların giderilmesi işlemine **backlash kompanzasyonu** denir.

Robotun parçaları tasarlandıktan sonra alüminyum kompozit olan parçalar cnc router'da, alüminyum olan parçalar lazerde kesilmiştir. Bir router lazer kadar hassas olmamakla birlikte bu seviye bir iş için yeterli görülmüş ayrıca piyasada lazer kesim yapan firmaların bu malzemeyi bulundurmaması sebebiyle CAD ardından CAM yapılarak bu parçalar üretilmiştir.

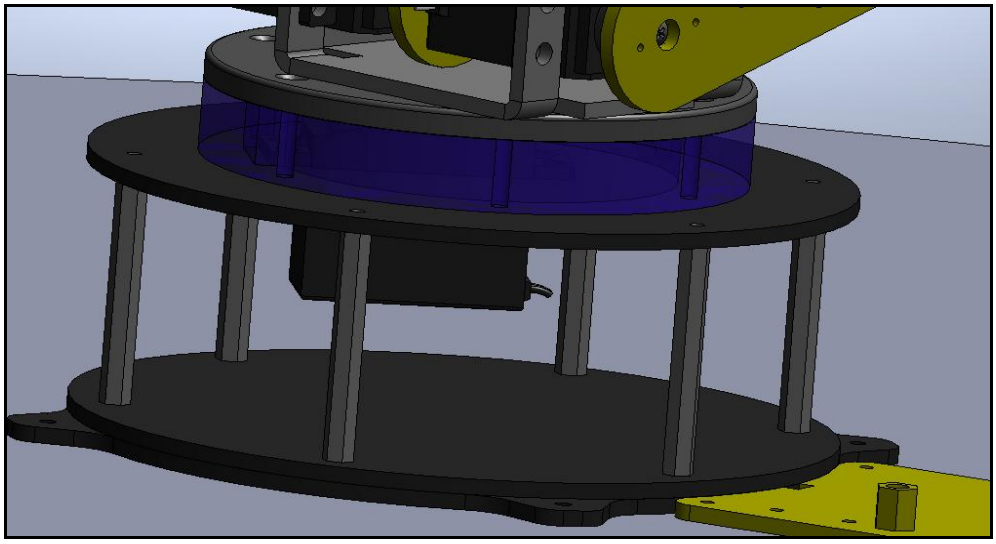
Robotun ana gövdesi ve geri kalan uzuvlarının üretiminde hafif ve aynı zamanda dayanımı yüksek olan bir malzeme tercih edilmelidir. Bu boyutta ve bu amaçta robotlar üretilirken ‘lexan’ tercih edilen profosyonel bir ürün olup ilk düşünülen malzeme olmuştur. İzmir de bu malzemeye ulaşılamamıştır. Bu yüzden çeşitli alternatifler olan Plexiglass, Alüminyum, Derlin, Dakota gibi malzemeler üzerinde durulmuş ama hem dayanımının yüksekliği hem hafif olması hem de alüminyum plakanın içindeki polietilen tabaka sebebiyle sönümleyici özelliği olması açısından robotun imalatı için doğru malzeme olduğuna karar verilmiştir.

Cnc Router daha önce belirttiğimiz olan backlash’e sahip olması ve bu denli küçük parçalar üretilmesi sebebiyle küçük ölçü hataları bu boyutta küçük problemler ortaya çıkarmış bu yüzden bu hatalar eğeleme yöntemiyle giderilme yoluna gidilmiştir. Parçaların kesimi sırasında ortaya çıkan çapaklar zımpara yapılmış ve pürüzsüz bir görüntü elde edilmeye çalışılmıştır. 2.bölümde robotun Solidworks ortamındaki tasarım hali, parçalarının teknik resimleri ve montaj sırasındaki fotoğrafları gösterilecek olup adım adım tasarım aşamaları bu grafiklerle açıklanacaktır.

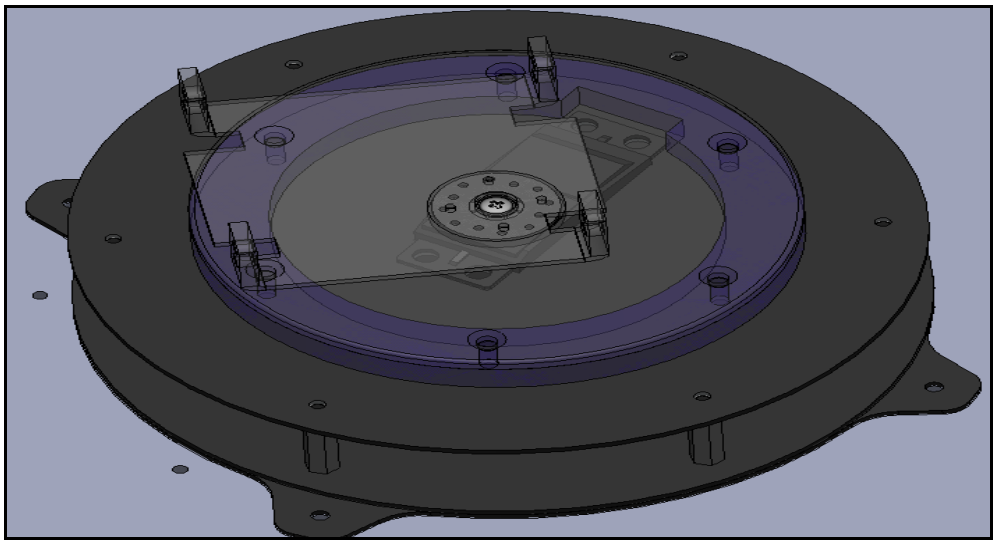
Montaj elemanı olarak M2,M3,M4 cıvata kullanılmış olup M2, M3 boyutlarındaki cıvata, somun, rondela vb. piyasada standart olarak kolay bulunmayan bağlantı elemanlarıdır. İzmir’de bir iki yerde satılmakta olup pahalı olmaları önemli bir eksidir. Bunun için tasarım yapılırken hafiflik önemli bir unsur değilse en küçük M4 bir cıvata - somun bağlantısı yapmakta fayda vardır. Gerekli görülen yerlerde cıvatalar kesilmiş uçları somunları ağızlayabilmesi için tekrar yuvarlatılmıştır.

2.1.7 Robot manipulatör'ün Solidworks ile tasarımının oluşturulması

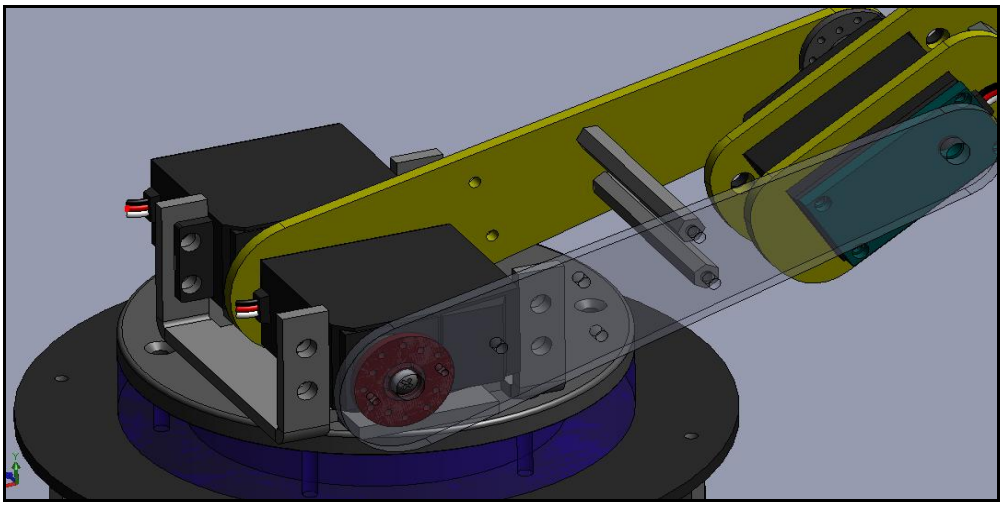
Bu aşamada mekanik tasarımın Solidworks ortamında hazırlanışı çizim ekranından alınan görüntüler üzerinden açıklanacaktır. Robotun tasarımı oluşturulurken Solidworks'ün 'optimizasyon' özelliği kullanılarak tasarım da düzenlemelere gidilmiştir. Bu bölümde tasarım programımın nasıl kullanıldığı ve menü'ler anlatılmayacak olup tezin amacından sapılmaması üzerine ayrıntıya girilmeyecektir.



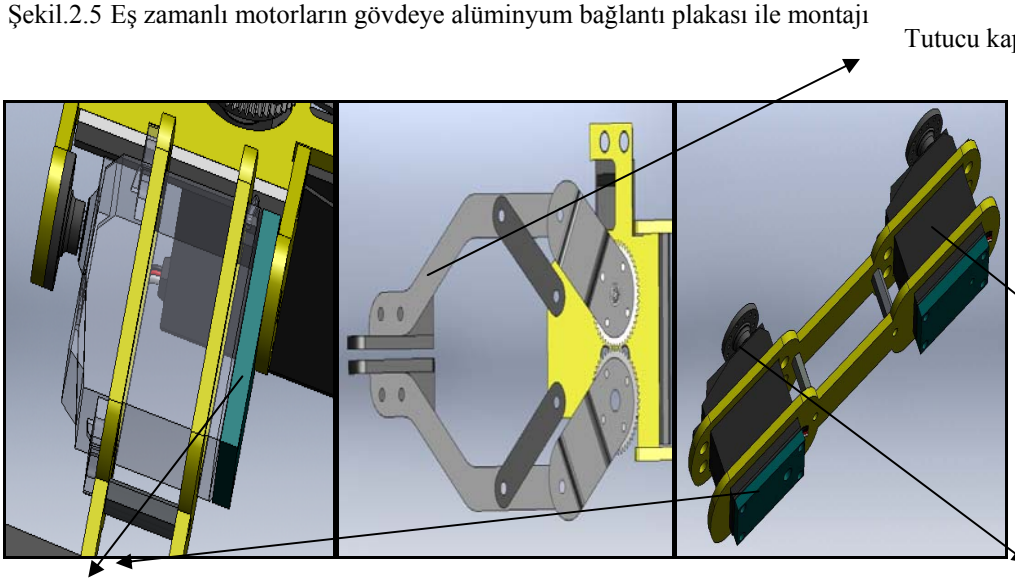
Şekil 2.3 Ana gövdenin oluşturulması ve motor bağlantısının yapılması



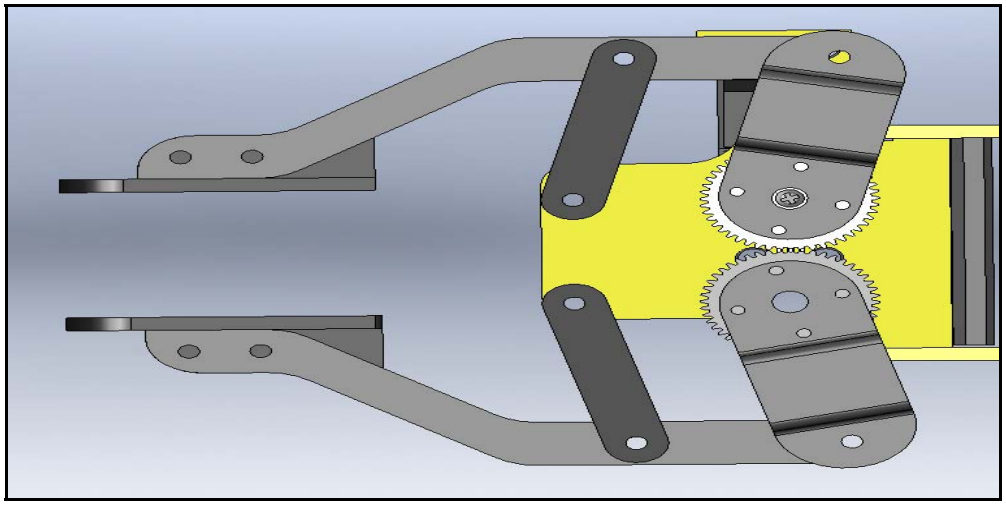
Şekil 2.4 Motorun gövdeye ve alüminyum bağlantı plakasına montajı



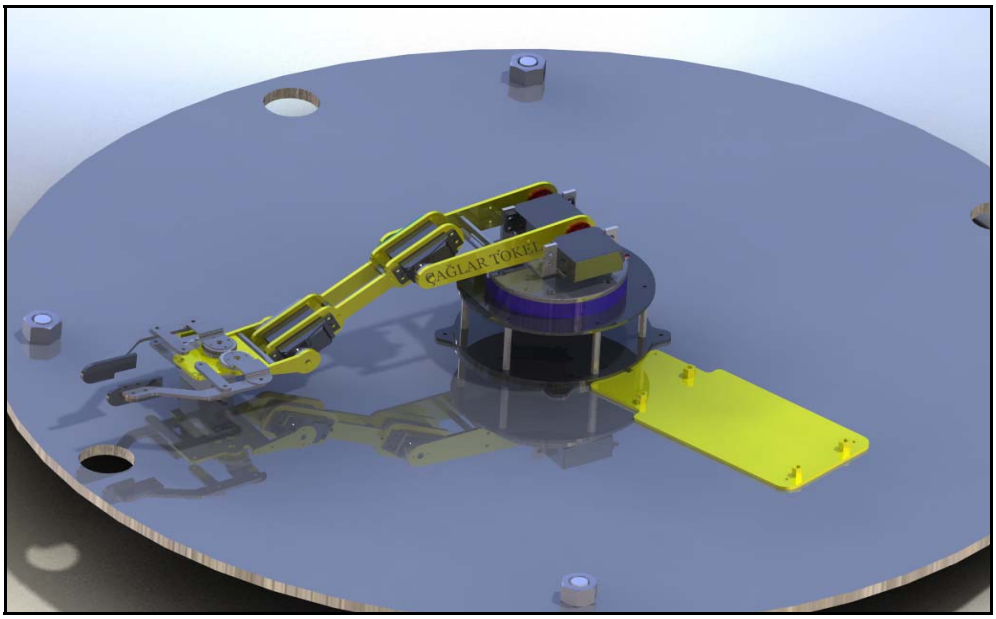
Şekil.2.5 Eş zamanlı motorların gövdeye alüminyum bağlantı plakası ile montajı



Şekil 2.6 Yeşil ek montaj parçaları tasarlanarak aradaki boşluğun giderilmesi



Şekil 2.7 Tutucu (gripper) tasarlanması. Tutucu açık



Şekil 2.8 Robotun tasarımının sona ermesi ve çalışma uzayında iş yaparken ki görüntüsü

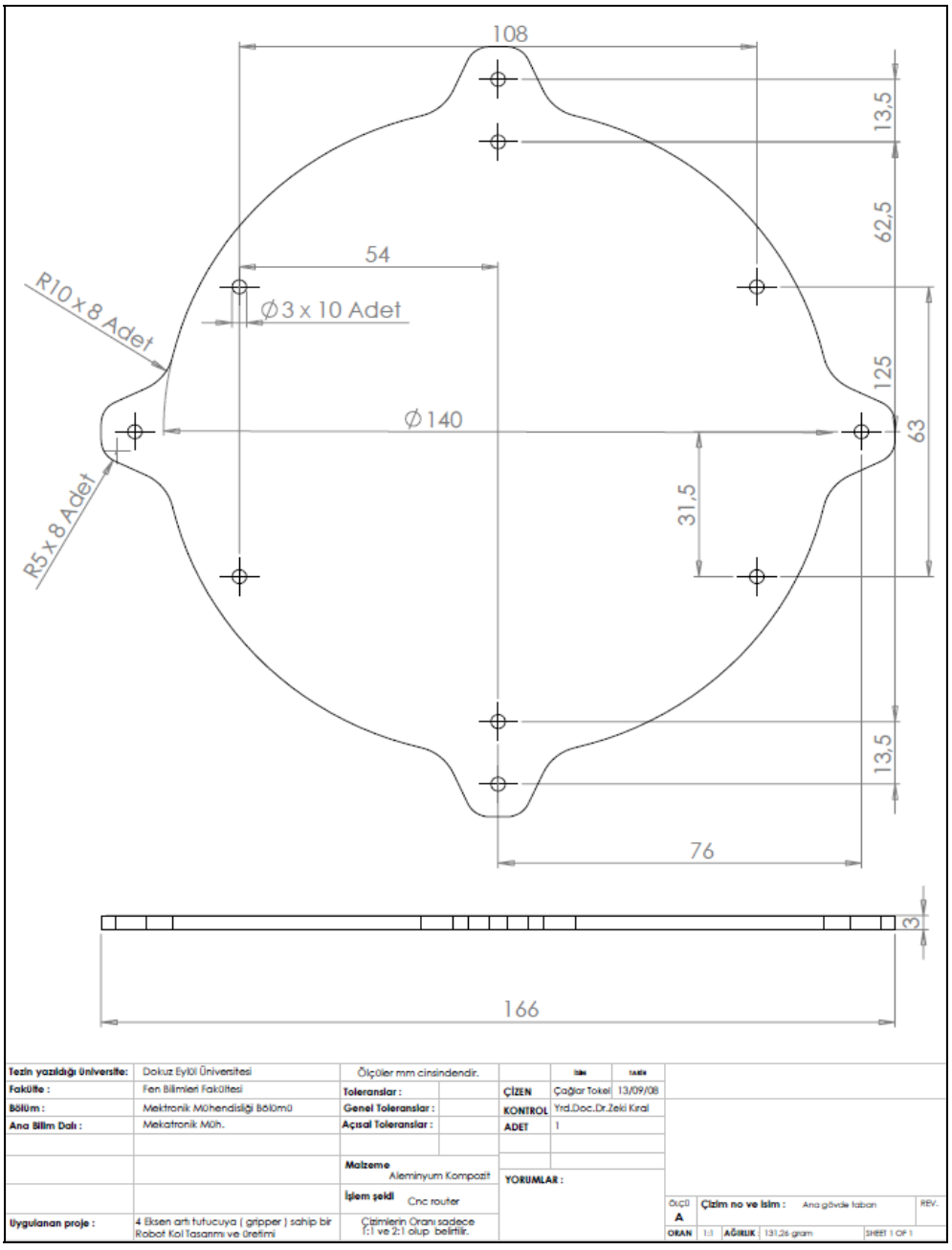
Bu bölümle birlikte Solidworks ortamında yapılan tasarım tamamlanmış robotun teknik resimleri çıkartılıp parçaların imalatına geçilmiştir. Bir sonraki bölümde robotun uzuvlarının ve gövdesinin teknik resimleri verilmiş olup detaylı olarak montaj ve üretim esnasında yapılması gerekenler anlatılmaya çalışılmıştır.

2.2 Robotun Mekanik Aksamının Teknik Resimleri

2.2.1 Ana gövde taban

Köşelerden 4 adet M3 civatayla 10mm et kalınlığındaki ana taşıyıcıgövdeye monte edilen parçadır. Bu parça içteki 6 adet delikten montajı yapılan aralayıcılarla yükseltilecektir.

Parça Ağırlığı: 131,26gr

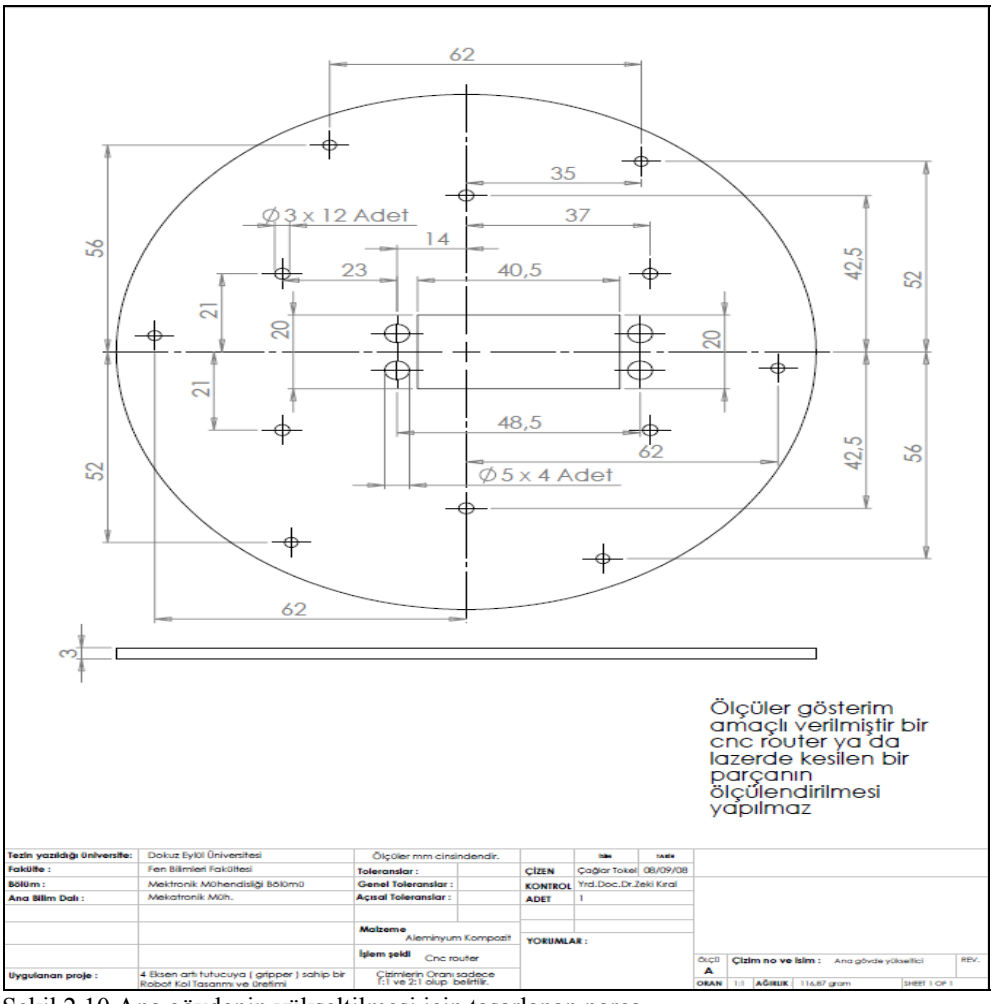


Şekil 2.9 Ana gövde tabanın teknik resmi

2.2.2 Ana gövde yükseltici

Z ekseni etrafında tarama yapmamızı sağlayan motorun monte edildiği parçadır. Bu parçanın tasarımı yapılırken servo motorun ölçüleri ve kafa (horn) yüksekliği hareketin düzgün bir şekilde iletilmesi açısından önem taşımaktadır.

Parça ağırlığı:116.87 gramdır.

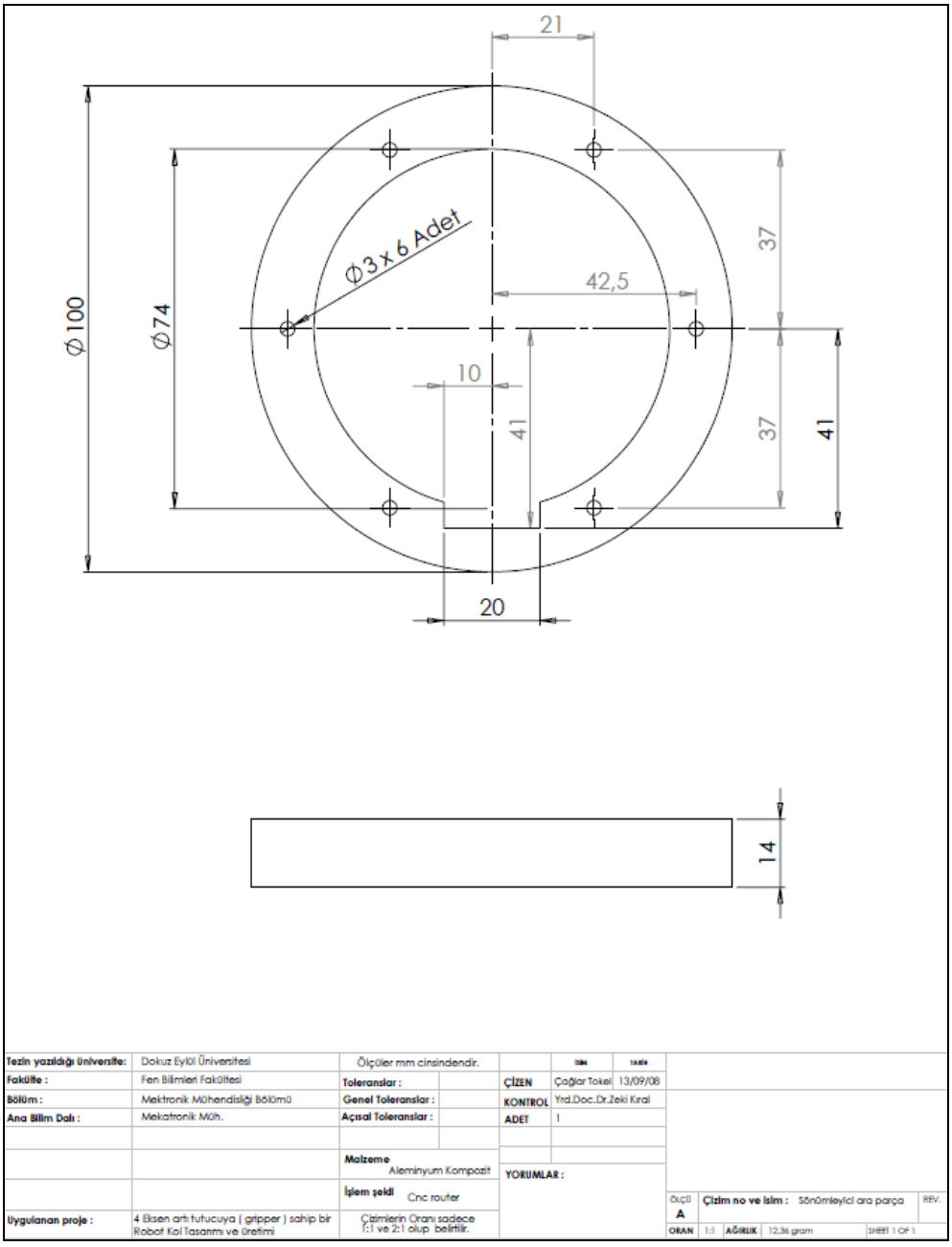


Şekil 2.10 Ana gövdenin yükseltilmesi için tasarlanan parça

2.2.3 Sönümleyici Ara Parça

Robotun hareketi sırasında doğan titreşimleri sönümlemek için düşünülmüş ve tasarlanmış bir parçadır. Malzeme olarak Forex seçilmiştir. PVC köpük olarak bilinen bu malzeme piyasa adıyla Dakota, sönümleyici özelliği, titreşimi azaltıcı yapısı sebebiyle tercih edilmiştir. Bu malzeme yapısı sebebiyle lazerde şekillendirilemez aksi halde üstünde yanıklar oluşur. 3 mm'lik delikler **2.2.2** ve **2.2.4**'e bağlantı için açılmıştır.

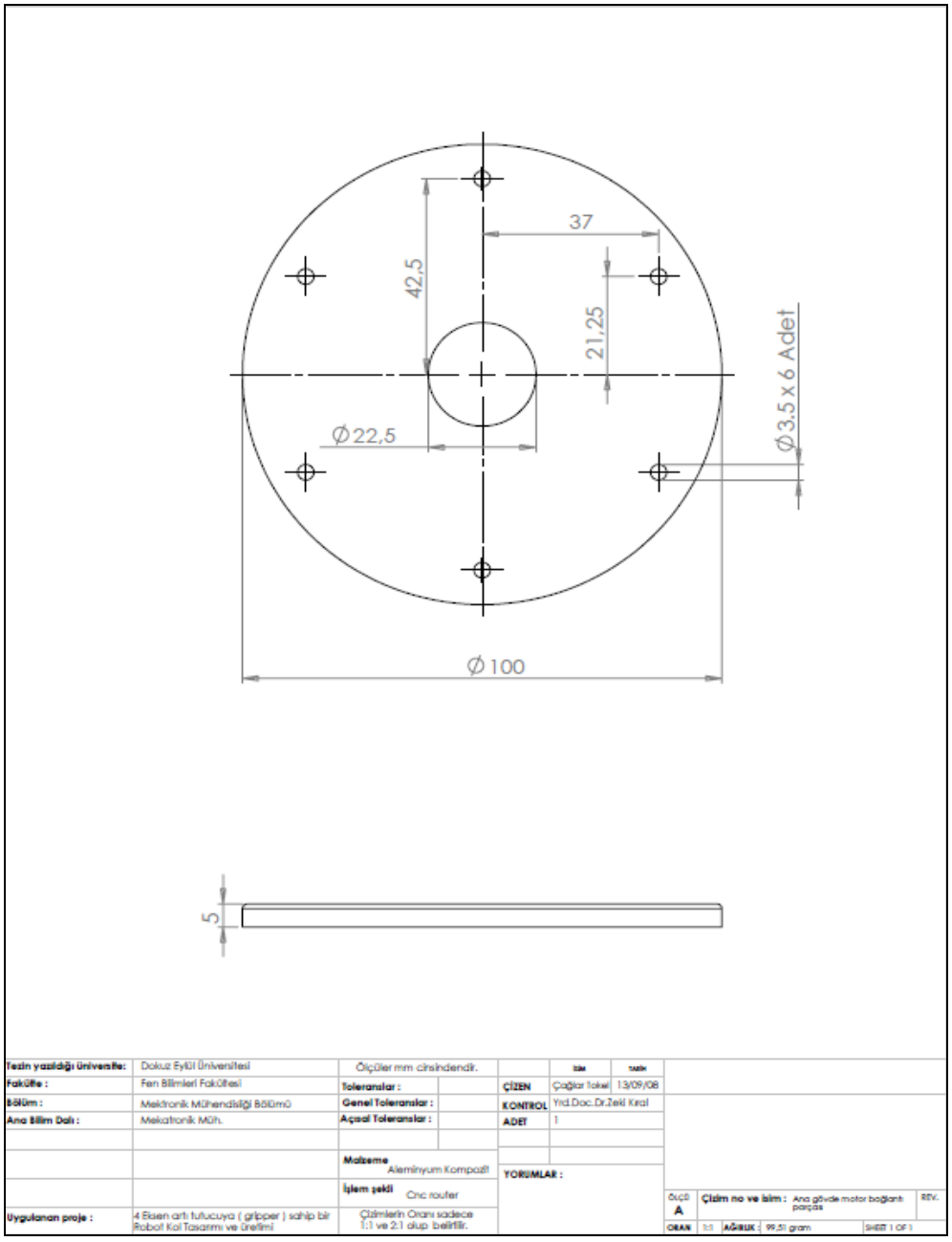
Parça ağırlığı: 61,79 gramdır



Şekil 2.11 Titreşimleri sönümlemek için Dakota'dan tasarlanan parça

2.2.4 Ana gövde motor bağlantı parçası

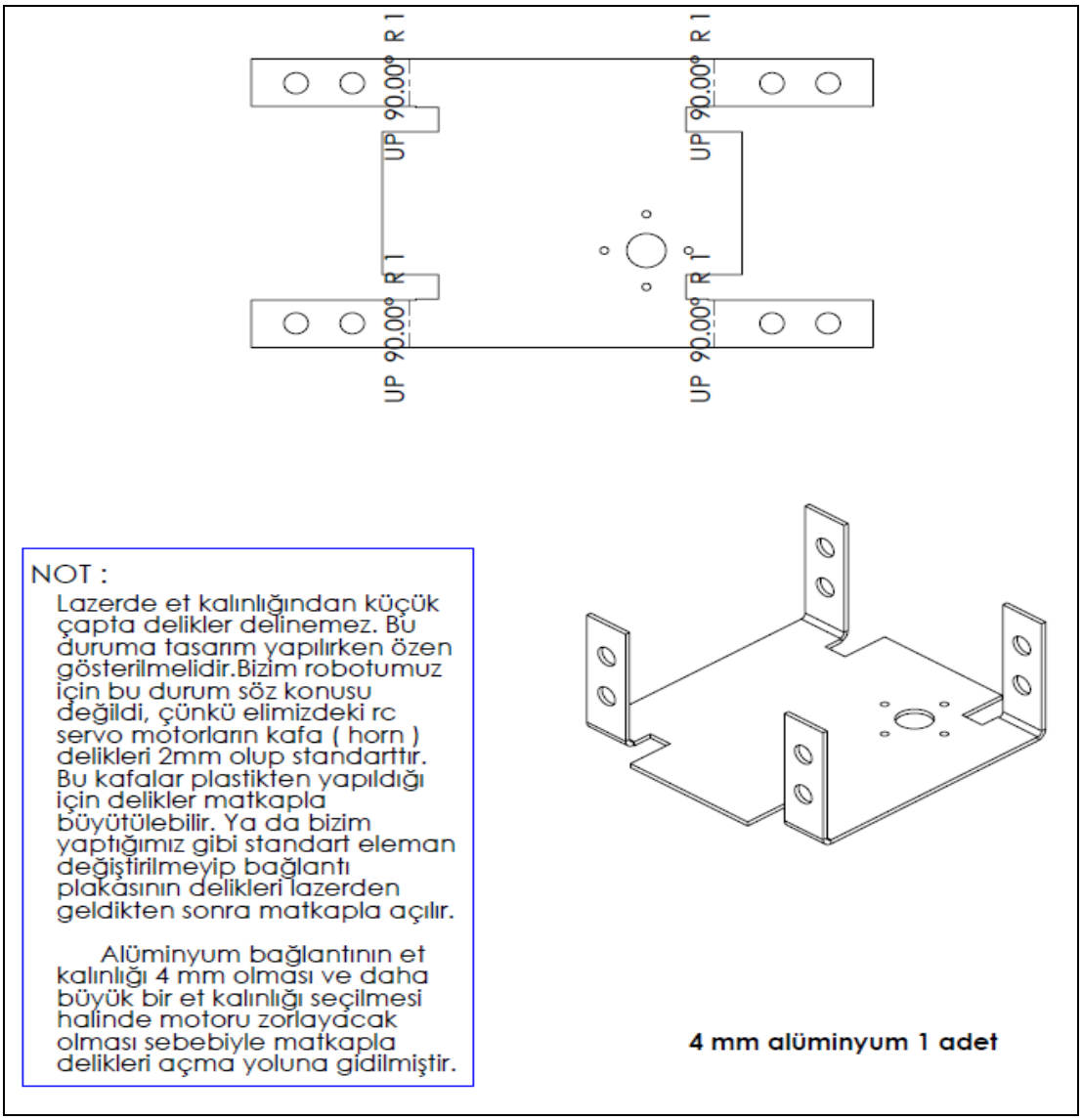
Motorun kafası (horn) 'nın bir sonraki sayfada görülen alüminyum bağlantı elemanına monte edilmesi için 22,5 çapında delik açılmıştır. Kenardaki 3mm'lik delikler ise sönümleyici elemana **2.2.3'e** bağlantı için açılmıştır. Parça ağırlığı: 99,51gr.



Şekil 2.12 Ana gövdedeki motorun montajının yapılması için tasarlanan parça

2.2.5 Eş zamanlı motorların bağlantı plakası

Eşzamanlı olarak çalışan 2 motorun montajının yapıldığı hassasiyeti yüksek bir parçadır. Dikkat edilmesi gereken ne çok ağır ne de çok hafif olmaması gerekmesidir. Ağır olmamalı ki motoru yük esnasında fazla akım çekmeye ve bozulmaya zorlamasın hafif olmamalı ki 9kg/cm tork verebilen 2 motor'un montaj plakası olarak zorlanıp zarar görmesin. Parça ağırlığı: 35,97 gr.

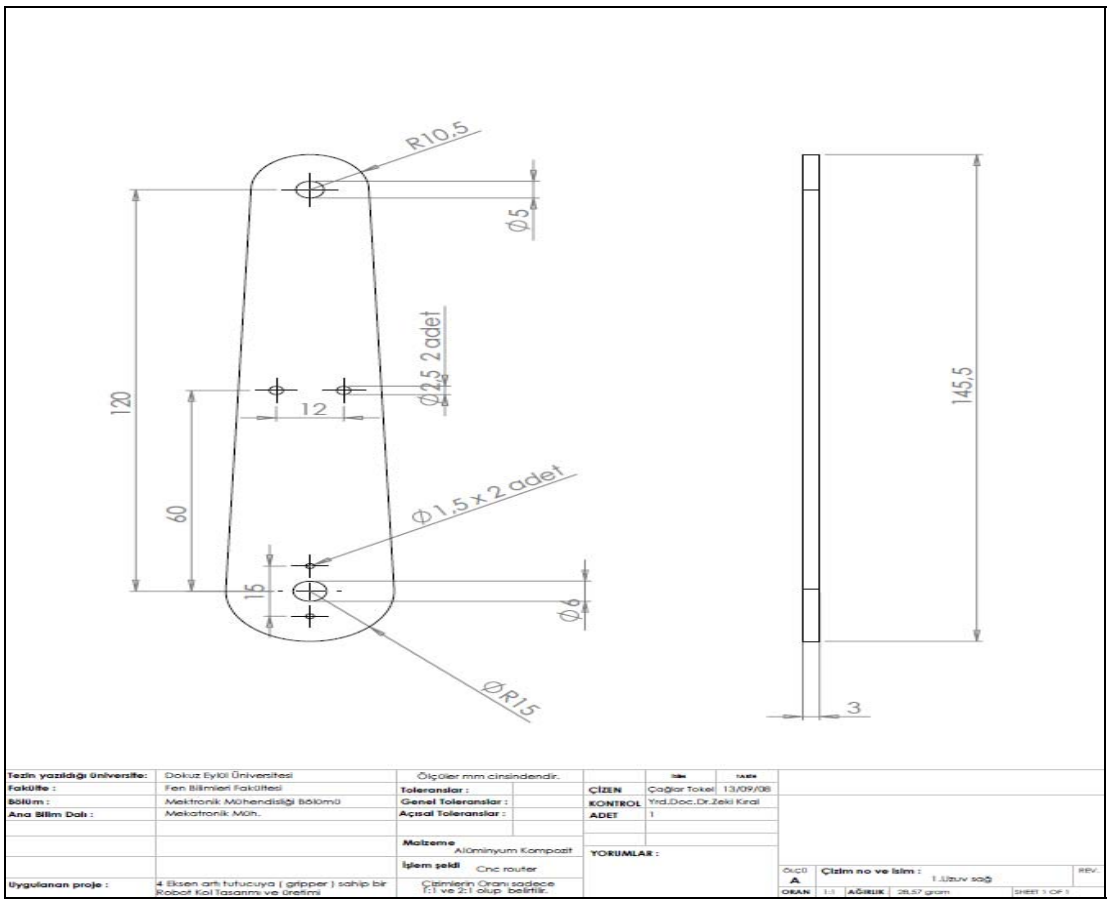


Şekil 2.13 Bu bağlantı plakası 4mm alüminyum'dan imal edilmiştir.

2.2.6 Eş Zamanlı Çalışan Motorlar ile İletimin Yapıldığı Uzuvlar

Eş zamanlı çalışacak 2 servo motorun hareketi ilettiği bu uzuvların motorlara bağlantı şekli çok önemlidir. Rc servo motorlar aynı marka aynı ürün olsa dahi eş zamanlı iki motoru çalıştırmak için motorları çalıştırmak ve hareketi aktaracağı parçalara monte edip durumu gözlemlemek ve gerektiği kadar deneme yapmak gerekir. Atlanacak bu ayrıntı ve iki motoru da en başa veya en sona getirip hareket organlarını bağlama ve uzuvların hareketini gözlemlemeden geçildiği takdirde motorlar farklı pozisyonlara gitmek isteyeceklerdir. Böyle bir durum ikisinin de çok büyük akım çekmesine sebebiyet verecektir. Bu da motorların aşırı ısınmasına uzun süre bu şekilde çalışılması halindeyse arızalara ve motorun çöp olmasına sebep olacaktır.

Parça ağırlığı: 28,57 gr

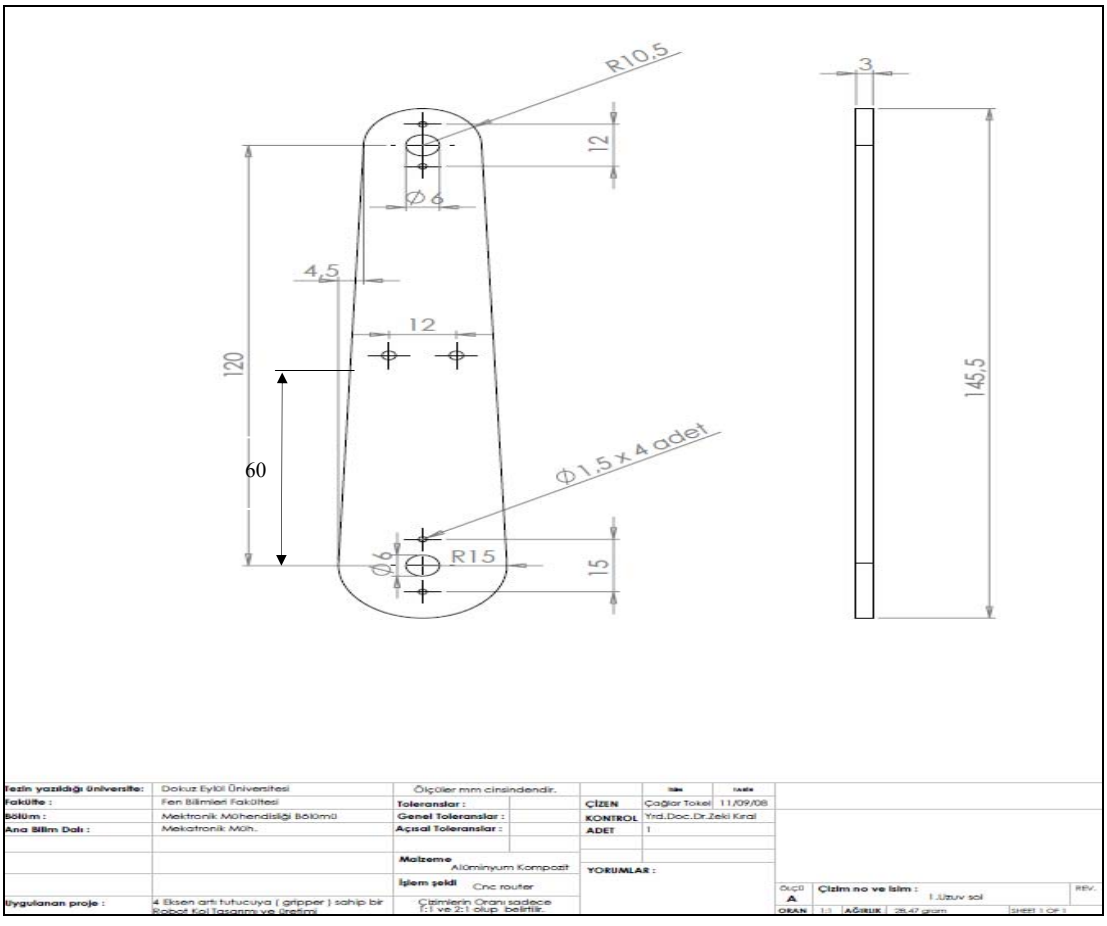


Şekil 2.14 Omuz ve dirsek motorlarının montajının yapıldığı 1 numaralı uzuvlar

Bunun için yapılması gereken önce tabi ki yazılımla iki motora da aynı sinyali yollamak olmalıdır. Elektronik ile ilgili bölümde bu kısım ayrıntıyla incelenmiştir. Ardından motorlar aynı pozisyona gitmek için sinyal aldıklarından dolayı beklentimiz iki kolun da aynı pozisyona gidecek olmasıdır. Fakat bu durum böyle olmayacaktır. Bu fark gözlemlenip motorlardan birinin kafası hareket iletim organından çıkarılmalı ve göz kararı olarak, gözlemlenen açı kadar döndürülmelidir. Bu işlem gerektiği kadar yapılmalı ve sinyal neticesinde tam olarak aynı pozisyona geldikleri görüldükten sonra bir sonraki adıma geçmeli ve aralayıcılar takılmalıdır.

Not: Bu dirsek ekseninde eşzamanlı (senkron) çalışacak olan motorlar yüksek torklu olup 1cm'e 9kg verebilen yüksek torklu rc servo motorlardır.

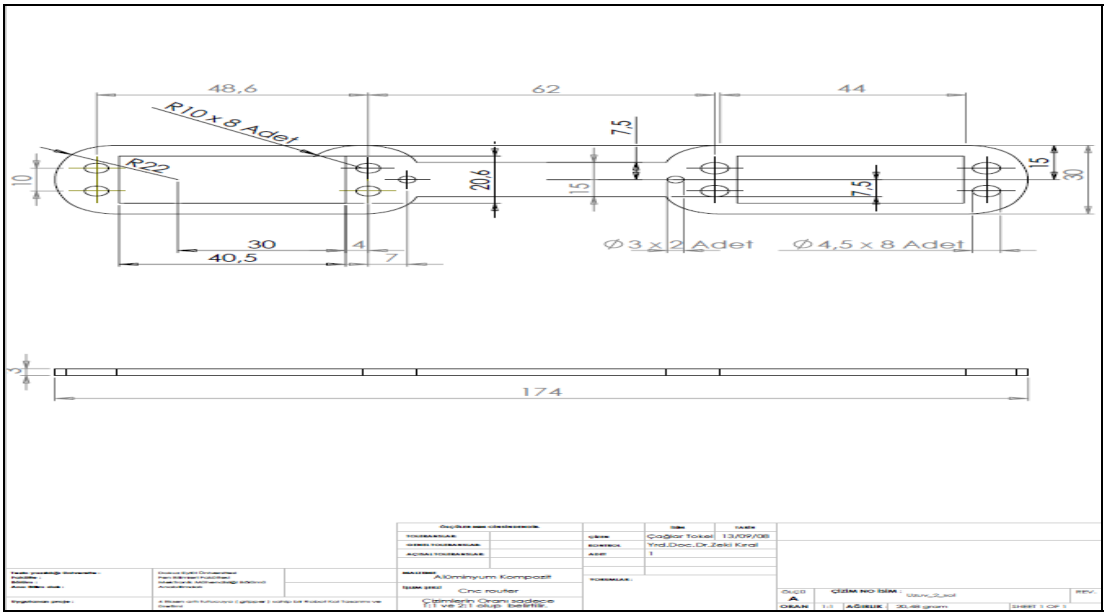
Parça ağırlığı: 28,47gr



Şekil 2.15 Omuz ve dirsek motorlarının montajının yapıldığı 1 numaralı uzuvlar

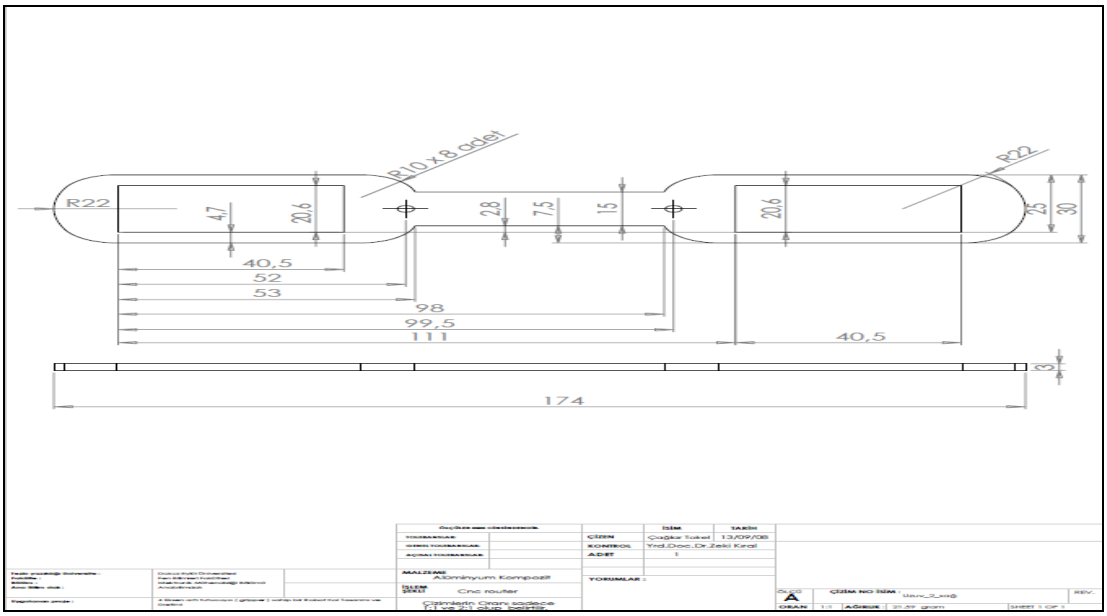
2.2.7 Dirsek ve Bilek Ekseni Motorlarını Taşıyan Eleman

Dirsek ve bilek için montaj parçasıdır. Kritik olan tek başına en fazla yüke maruz kalan motor Dirsek motorudur. M4 cıvata ve somun ile motorların montajı yapılmış ardından bilek bölgesinin montajına geçilmiştir. Parça ağırlığı: 20,48 gr



Şekil 2.16 Dirsek ve bilek motorlarının montajının yapıldığı 2 nolu uzuvlar

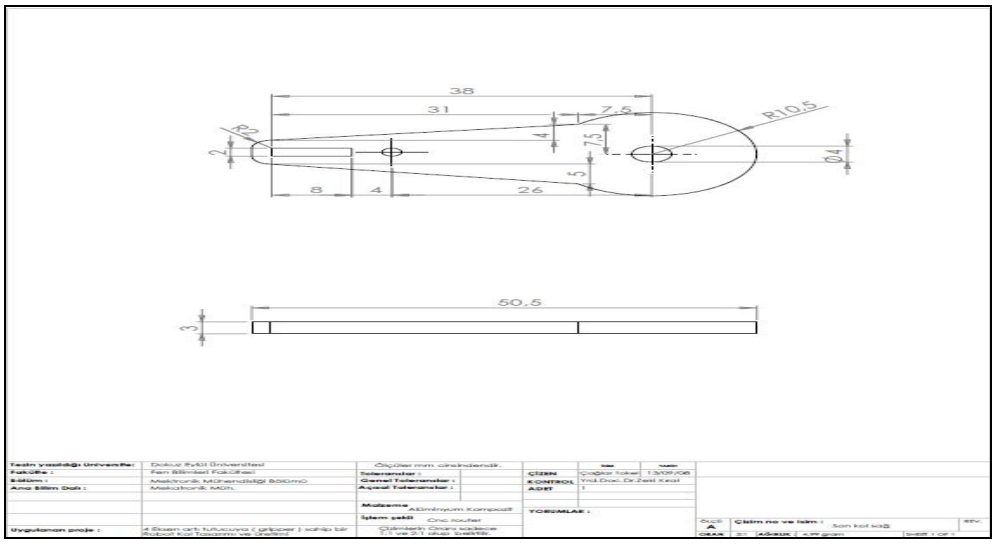
Bu ikinci parça dengeyi sağlamak dayanımı arttırmak ve simetriyi sağlamak amacıyla tasarlanmıştır. Parça ağırlığı: 21,59 gr



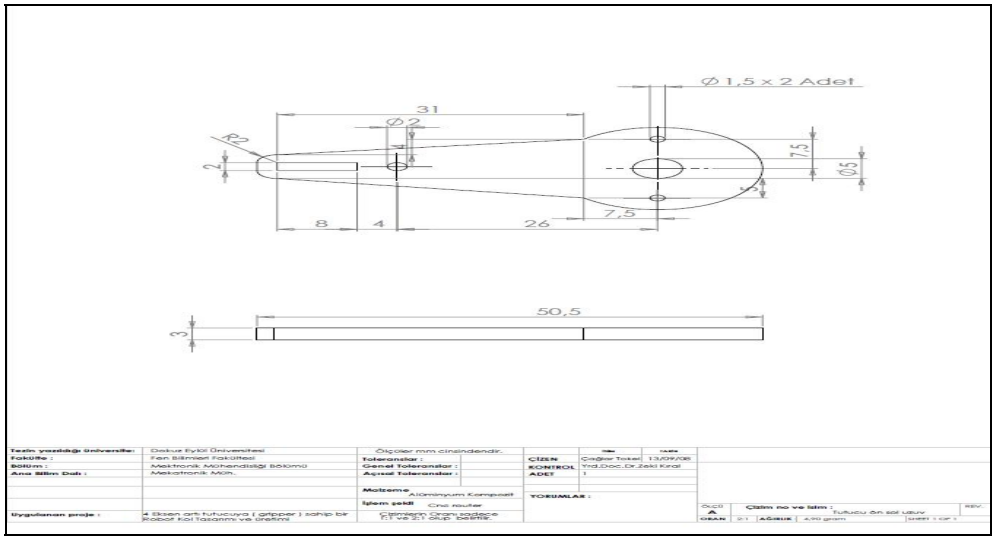
Şekil 2.17 Dirsek ve bilek motorlarının montajının yapıldığı 2 numaralı uzuvlar

2.2.8 Tutucu Eksen Hareketinin İletimini Sağlaması için Tasarlanmış Elemanlar

Bu iki eleman 2mm çapındaki delikle bağlantısı yapılacak olan standart aralayıcı eleman ile hareketin birinden diğerine aktarılması ile eş olarak çalışacaklardır. Birinin montajı motora yapılırken diğer uzuv'un daha bir yere sabitlenmesi gerektiği için motorun arka tarafına yeşil renkteki parça tasarlanmış ve bağlantı için bu parçaya delik açılarak parçanın bu delikten motora sabitlenmesi yoluna gidilmiştir. Parça ağırlıkları sırası ile 4,90 gr ve 4,99 gr' dır.



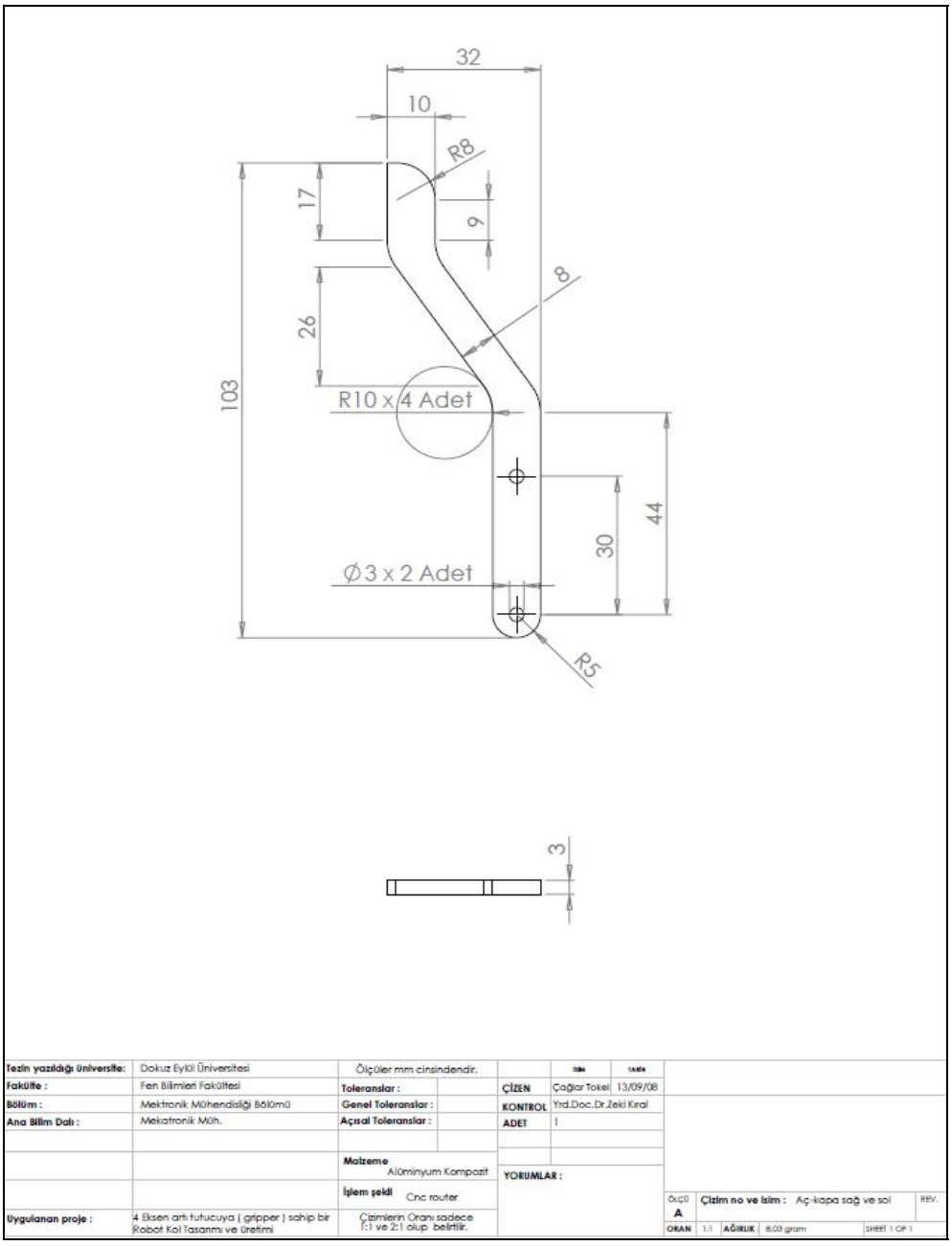
Şekil 2.18 Bilek Ekseni'nin motorunun montajının yapıldığı 3 numaralı uzuvlar



Şekil 2.19 Bilek Ekseni'nin motorunun montajının yapıldığı 3 numaralı uzuvlar

2.2.9 Tutucu Aç / Kapa mekanizması elemanları

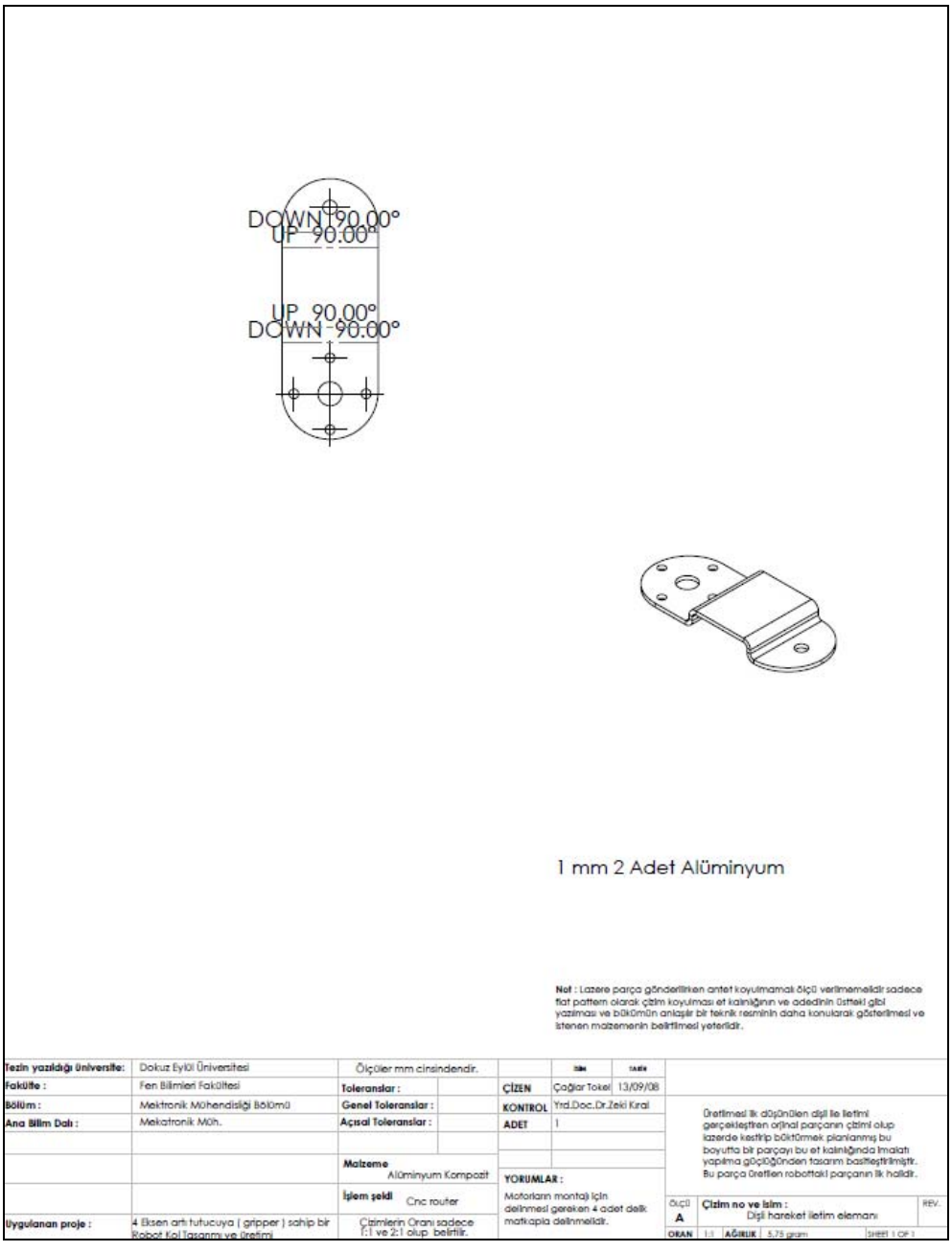
Bu parçalar dört kol mekanizmasının hareketi ilettiği parçalardır. Eleman üzerine açılmış 3mm'lik iki adet delikle monte ederek hareketi bu elemanlara iletmekteyiz. Tutucunun bütün olarak incelendiği bölümde dört kol mekanizması anlatılacaktır. Parçanın tasarımında uç bölümün ağırlığı minimuma indirmek için olabildiğince küçültülmeye gidilmiştir. Parça ağırlığı:8,03 gr'dır. Bu parçadan 2 adet olup tutucu motorunun çalışmasıyla hareketin iletildiği, tutma bırakma işlemini yapan eleman olarak tasarlanmıştır.



Şekil 2.20 Tutucu'nun açma kapama işleminin nihayetinde objeyi tutan eleman

2.2.10 Dişli Hareket İletim (orijinal) Elemanı

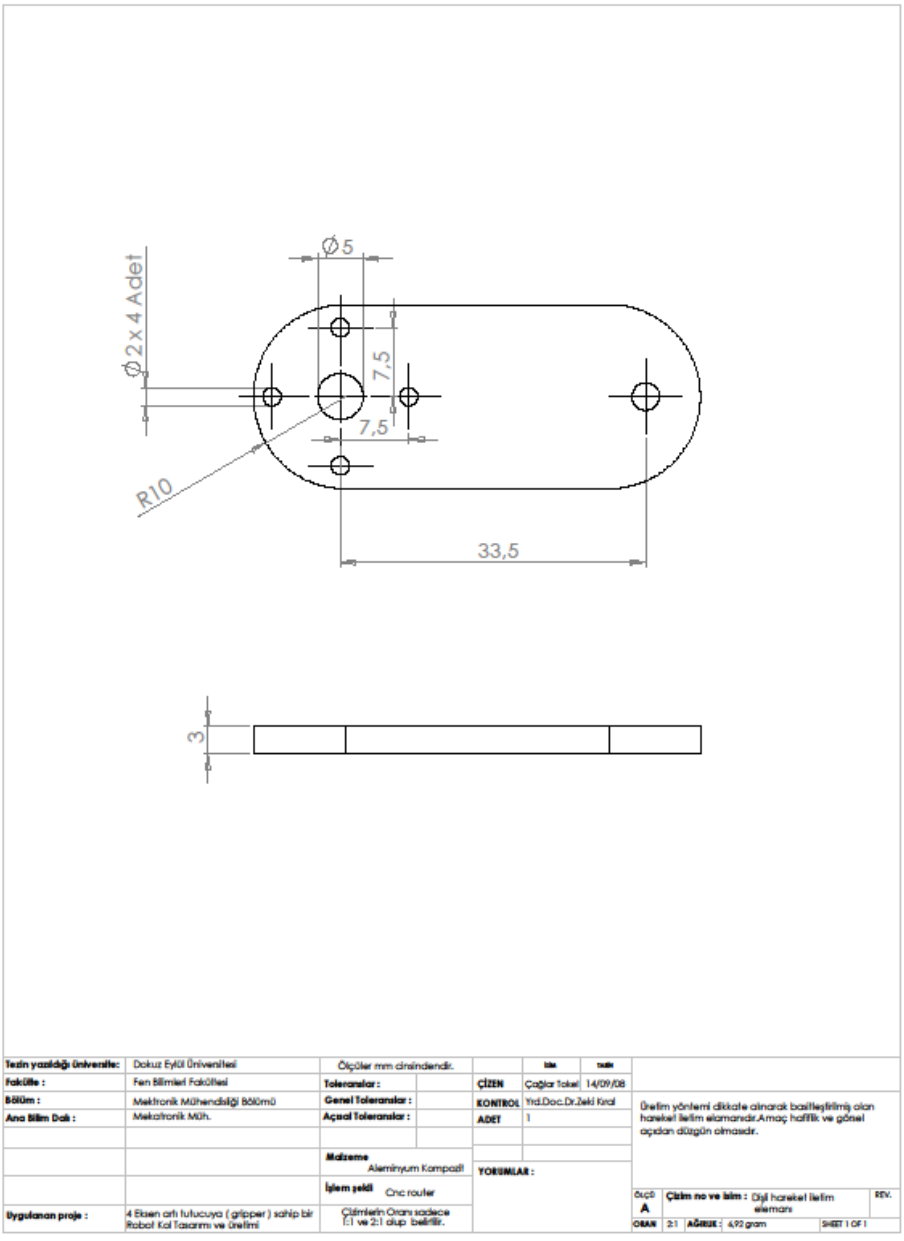
Bu eleman 2.2.11den önce tasarlanan dişlilere monte edilecek bu parçaların dişlilere temasını engellemek için tasarlanmıştır. Tasarıma başlanırken kullanılan dişliler bu yüzden üretim teknikleri göz önünde bulundurularak daha basit bir tasarıma indirgenmiş ve bu doğrultuda içe gömük dişliler tamamen düz olanlarla değiştirilmiştir. Not: Lazere gönderilecek çizimlerde antet ve ölçü konulmamalıdır.



Şekil 2.21 Dişli hareket iletim elemanı

2.2.11 Dişli Hareket İletim Elemanı

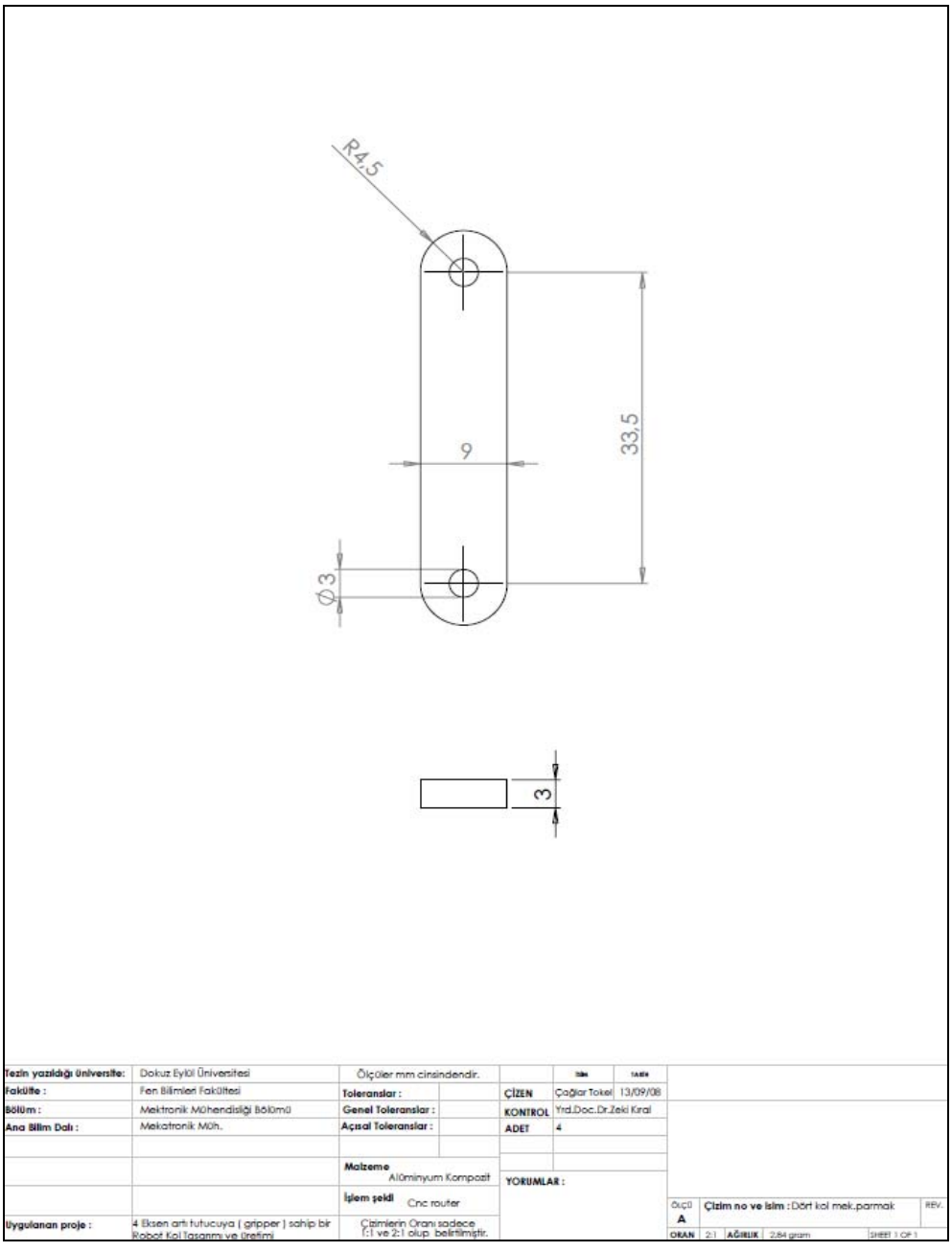
4,92 gr olan bu eleman tutucudaki (gripper) hareket iletim organıdır. Rc servo motordan dişlilere monte edilmiş bu eleman ile hareket birinden ötekine aktarılır, bu sayede tutucu bölgesinde birbirine eş ve simetrik bir hareket elde etmiş oluruz.



Şekil 2.22 Dişli hareket iletim elemanı

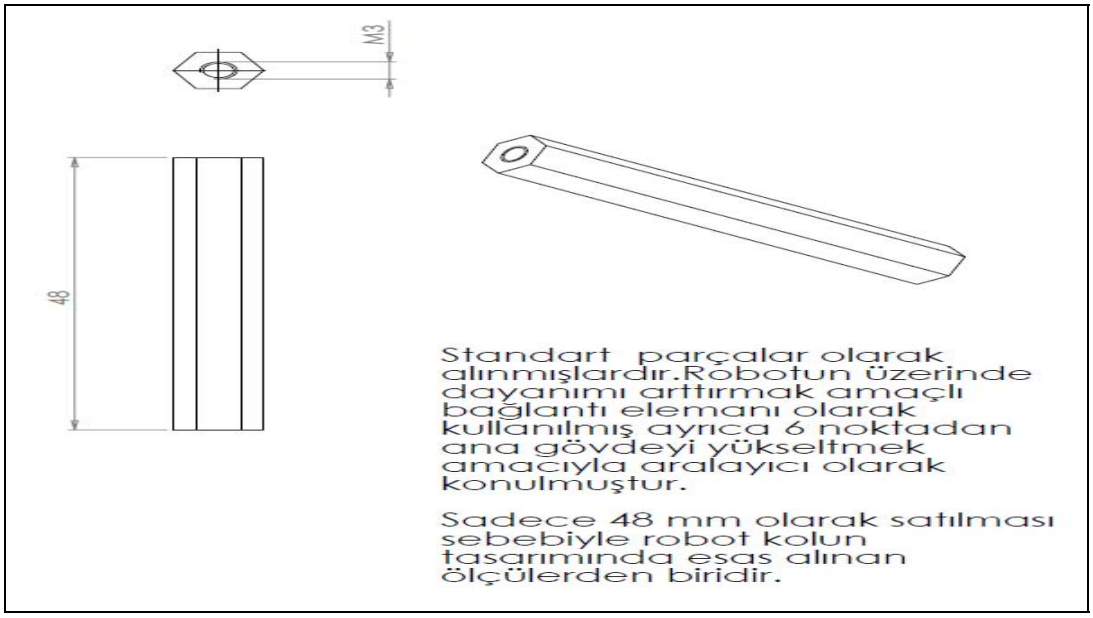
2.2.12 Tutucu Hareketinin Kararlı (Stabil) Olması için Tasarlanan Eleman

Sadece 2,84 gr olan bu eleman 4 kol mekanizmasının hareket iletimindeki hareketin kararlığını sağlayan elemandır. Robotun tutucu bölümünde hareketin iletimini kolaylaştıran hafif ve basitlik esas alınarak yapılmıştır.



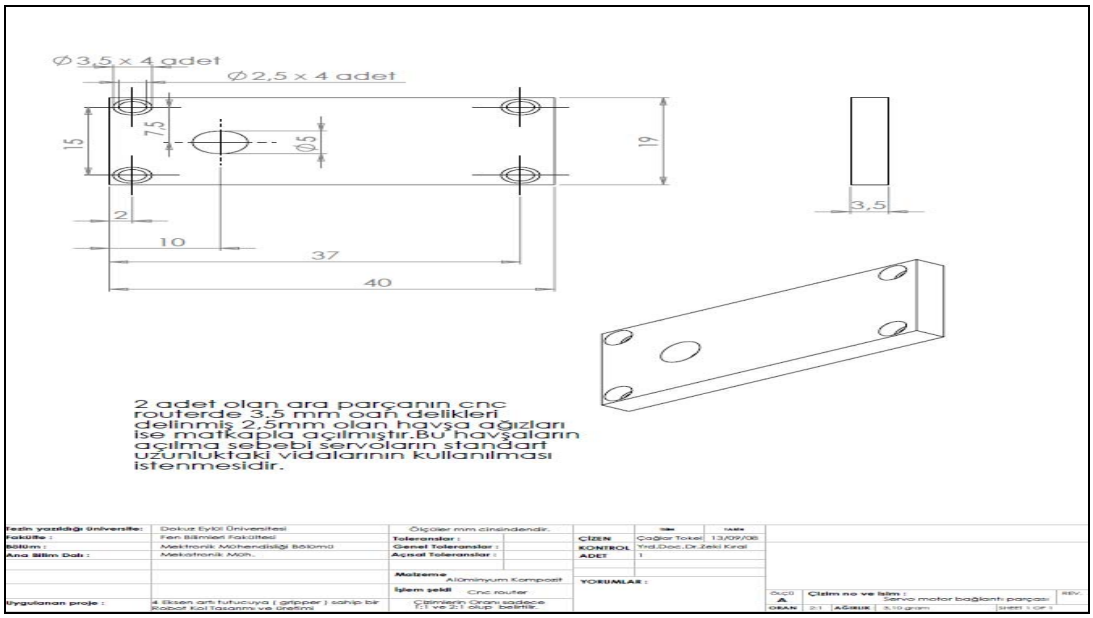
Şekil 2.23 Dört kol mekanizmasını oluşturan elemanlar

2.2.13 Alüminyum aralayıcı (standart eleman)



Şekil 2.24 Alüminyum aralayıcı standart eleman Parça Ağırlığı = 3,89gr

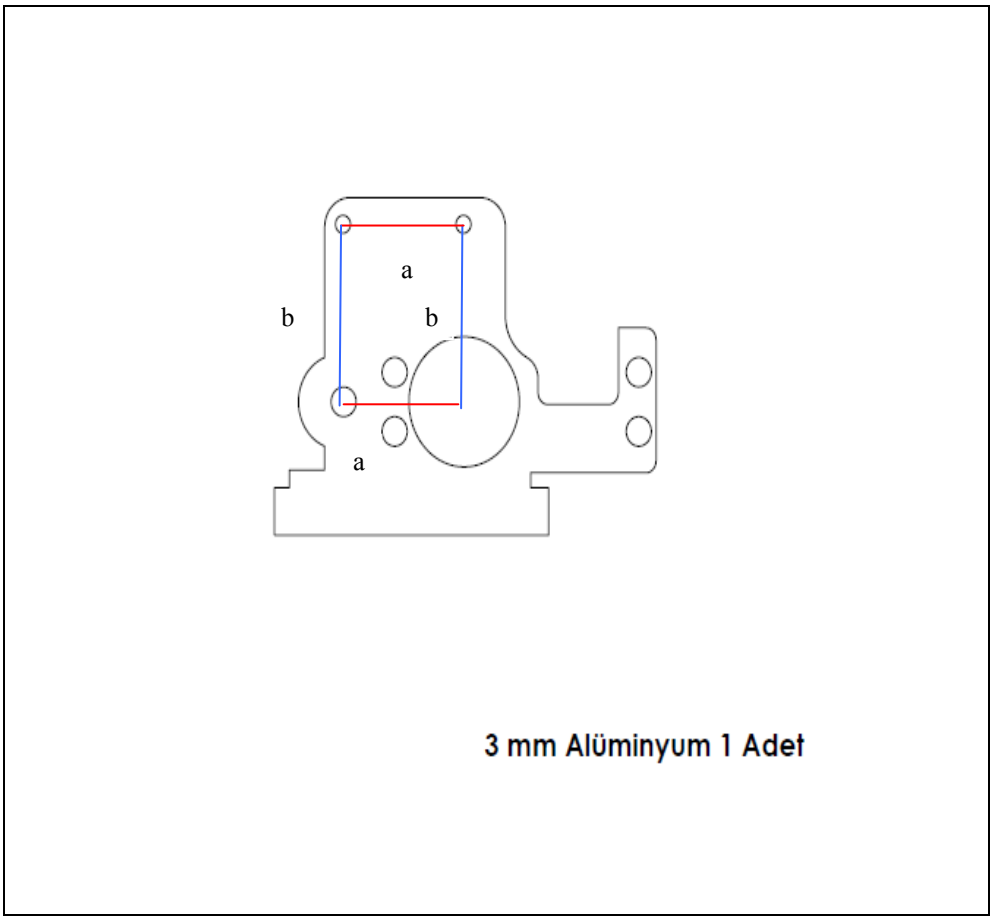
2.2.13.1 Servo Aralayıcı



Şekil 2.25 Dirsek ve bilek motorlarının montajının yapıldığı uzuv2'ye aralayıcı olarak tasarlanmıştır.

2.2.14 Tutucunun Sabit Ana Taşıyıcı Elemanı

Tutucunun tut-bırak işlemini yapması için gerekli olan motorun montajı için dizayn edilmiştir. Tasarımda hafifletme için boşaltmalar yapılmıştır. Tutucunun hareketini doğru bir şekilde yapmak tasarımı çok hassas olan bir elemandır. Bir dört kol mekanizmasından esinlenilerek karşılıklı kenarları eşit olan bir tasarım yapılmıştır ki hareket iletimi esnasında tutucunun iki tarafı da aynı hareketi eş zamanlı olarak yapabilsin.



Şekil 2.26 Gripper'ın üzerine kurulduğu ana taşıyıcı eleman

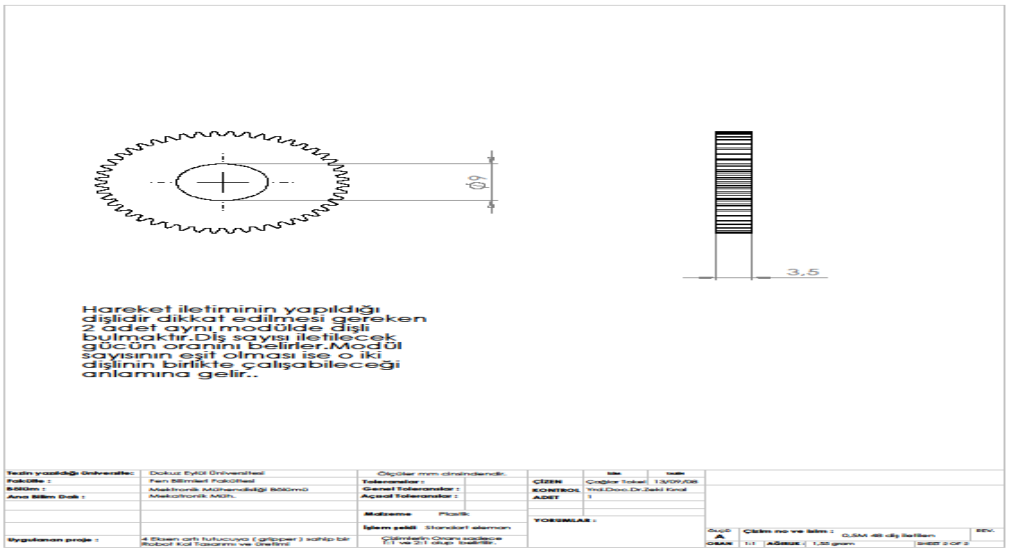
2.2.14.1 Dört Kol Mekanizması

Dört adet uzvu olan hareketli en basit mekanizmalardır. Bu mekanizmada uzuvlardan her biri diğer ikisine bağlanarak kapalı çalışan bir sistem oluşturulur.Dirseklerin serbestlik derecesi 1 ise yani eskenel hareket bir düzlemde oluyorsa bu sistemler düzlemsel dört çubuklu mekanizmalar olarak adlandırılır.Bu sistemlerin konumu iki kolun pozisyonu ile belirlenebilir.Bu mekanizmalarda bir uzuv sabit olup hiç hareket etmez.

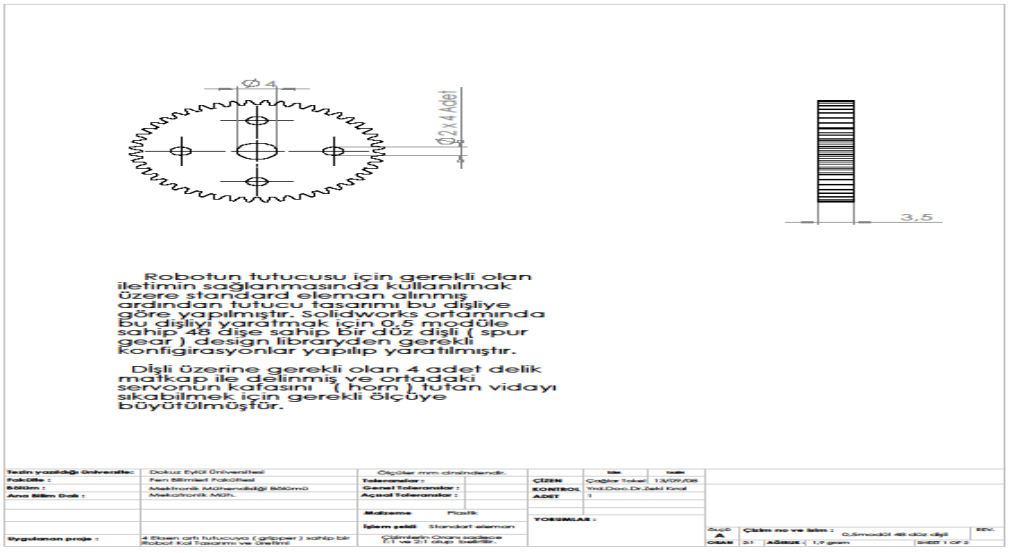
Tutucu'nun (gripper) tasarımında bu mekanizmalar incelenmiş ve hareket iletimini hem kolaylaştırmak hem de düzgün bir hareket sağlamak amacıyla 4 adet parmak tasarlanmış bunların karşılıklı olarak bağlandığı 2 elemanla tutucu tasarlanmıştır

2.2.15 0,5 Modül 48 Dişe Sahip Düz Dişli

Robotun tutma işleminin gerçekleşmesi için kullanılmak üzere alınmış standart elemanlar olup üzerlerinde tasarım ve montaj nedenleri ile teknik resimlerde belirtilen değişiklikler yapılmıştır.



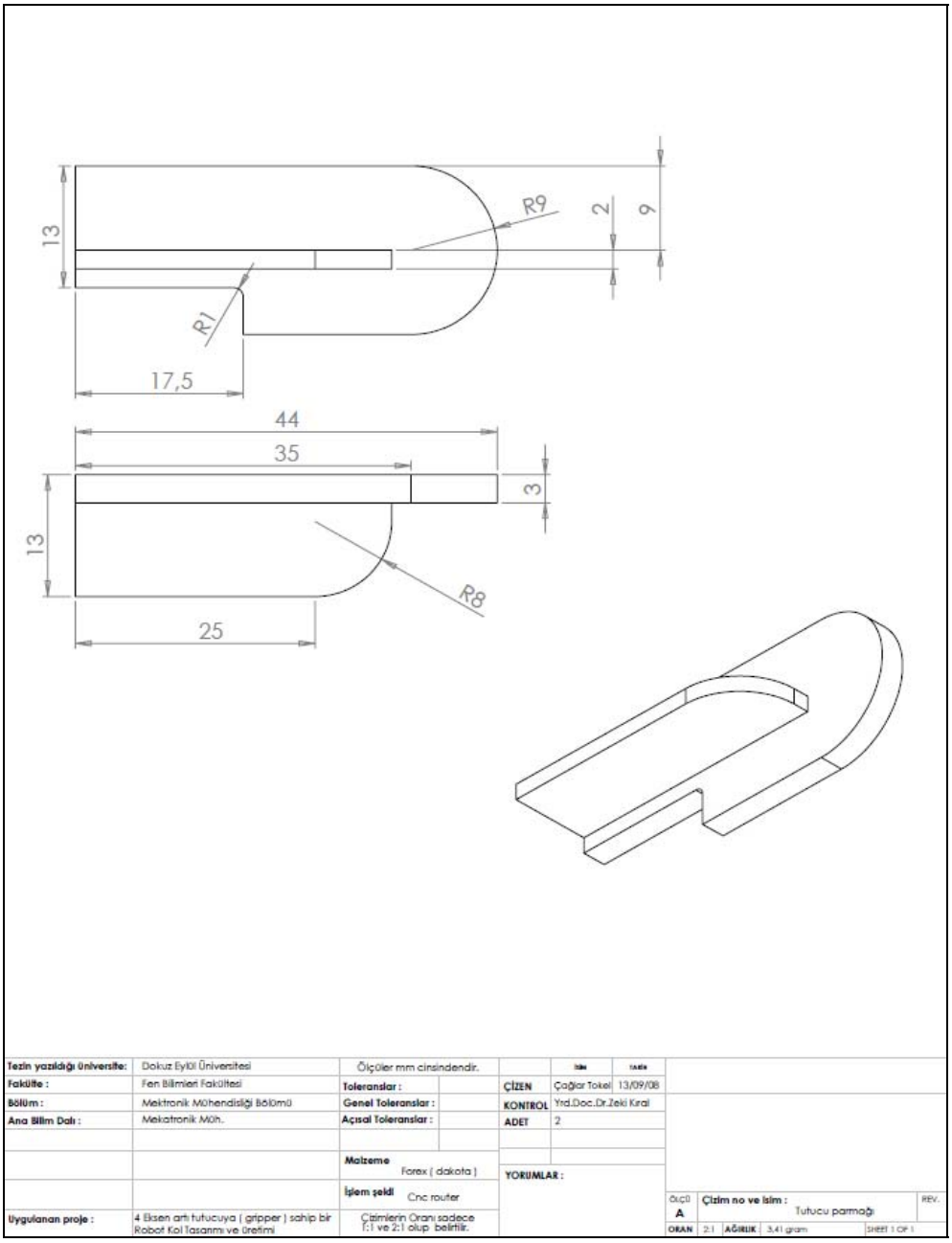
Şekil 2.27 Standart eleman olup hareket iletimin yapıldığı dişlidir



Şekil 2.28 Standart eleman olup hareket iletimini sağlayan dişlidir

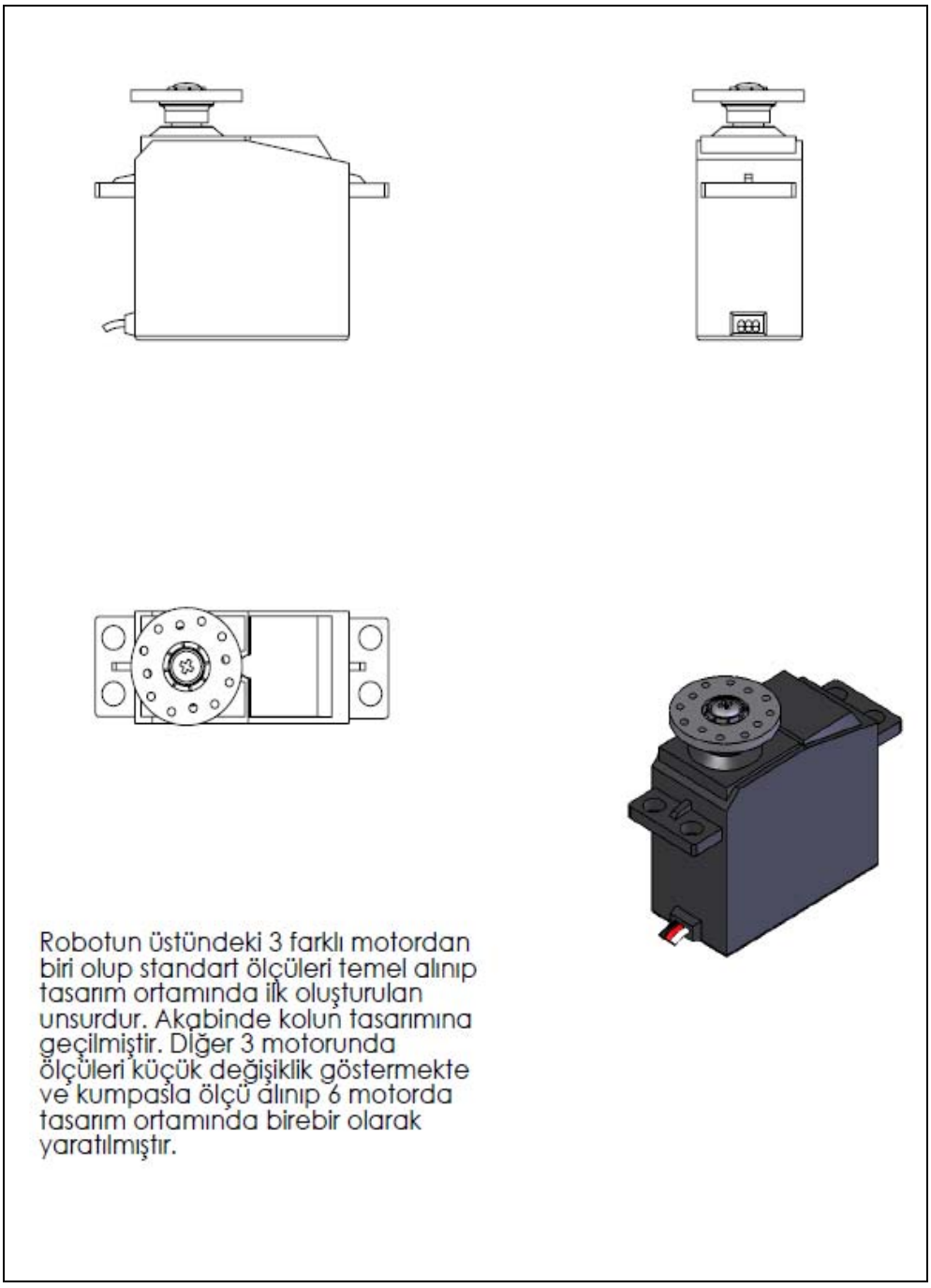
2.2.16 Tutucu Uçtaki Temas Yüzeyini Arttıran Parmak Eleman

Bu eleman istendiği takdirde robotun tutucusundan demonte edilerek (sökülerek) bu kez uç kısmındaki bir tutma operasyonuyla değil, tutucunun içindeki boş kısmının içine obje gelecek şekilde kavrayarak tutma ve bırakma işlemi yapılacaktır.



Şekil 2.29 Hassas tutuş için tasarlanan ve dakotadan imal edilen parmak eleman

2.2.17 Standart Elemanlar Olan Rc Servo Motorlar



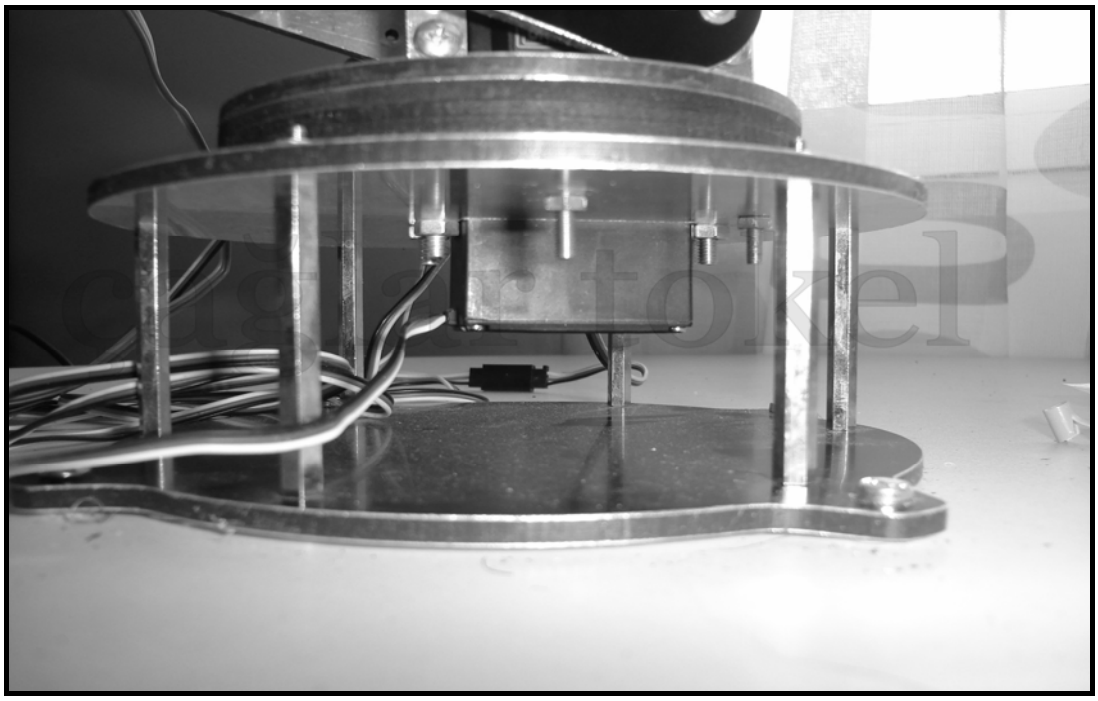
Şekil 2.30 Solidworks tasarım ortamında oluşturulan bir S3305 Rc servo motor

2.3 Robot Manipülatörün İmalat ve Montaj Aşaması

Robotun montajıyla ilgili dikkat edilmesi gereken noktalar bu bölümde resimlerin ardından yapılacak açıklamalarla gösterilerek açıklanacaktır.

2.3.1 Ana Gövdenin ve Z Ekseni Tarayıcı Motorunun Montajının Yapılması

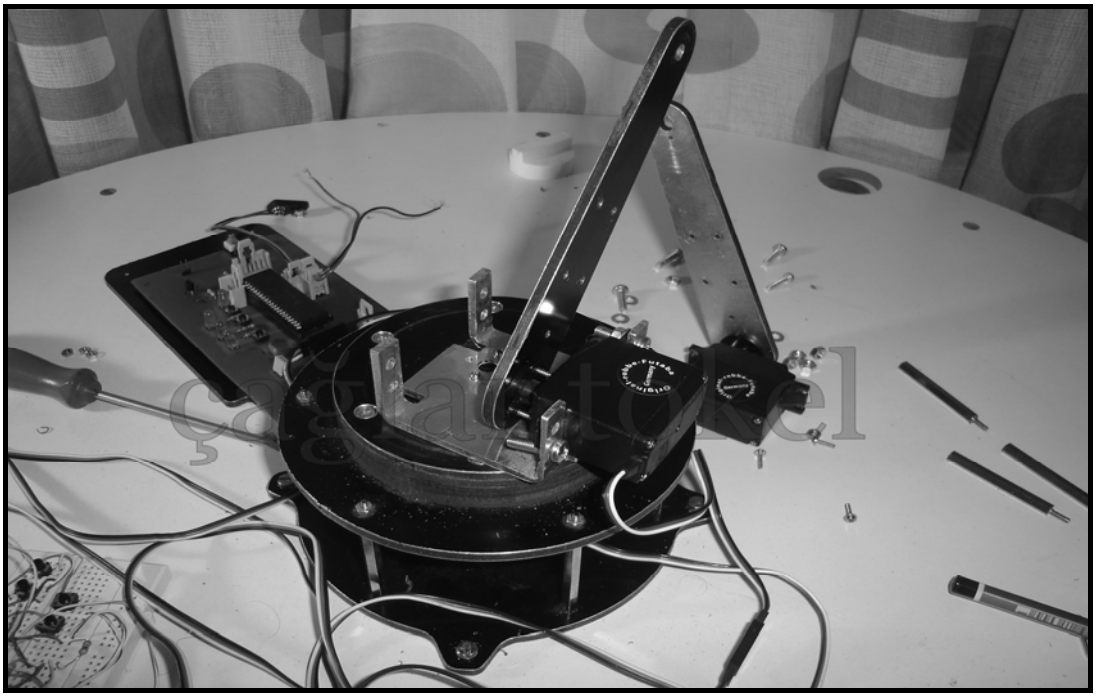
Bu aşamada gövdenin oluşturulması işlemi gerçekleştirilmiştir. Alüminyum aralayıcılarla yükseltme işlemi üst plaka ile gerçekleştirilmiş, Dakota'dan (Forex) imal ettiğimiz sönümleyici parçanın montajı ise motorun montajının ardından yapılmıştır. Üst kapağında montajı tamamlandıktan sonra oldukça rijit (sağlam) bir yapı elde edilmiştir. Bu işlemin ardından motorun kablosunun mesafesi bir standart Rc servo uzatma kablosu ile arttırılmıştır. Bu uzatma kabloları Türkiye'de tanesi 10 dolara satıldığından uluslar arası bir yoldan 20 tanesi 1 dolara satın alma yoluna gitme yoluna gidilmelidir. (e-bay). Gerekli cıvata-somun bağlantı elemanları ile montajın bu kısmı bitmiştir. Dikkat edilmesi gereken nokta servo motorun 4 noktadan düzgün bir şekilde civatalarının sıkılmasıdır. Aksi bir durumda motorun yerinden oynaması sonucu aşırı akım çekme ve yanabilme ihtimali vardır.



Şekil 2.31 Z ekseninde tarama yapan motorun ana gövdeye montajı sonrası

2.3.2 Eşzamanlı Motorların ve Alüminyum Montaj Plakasının Montajı

Montajın ikinci aşaması gövdenin ortasındaki servo motora alüminyum montaj plakasının monte edilmesidir.



Şekil 2.32 Robot eş zamanlı çalıştırılacak motorlar için ayarlamalar yapılırken

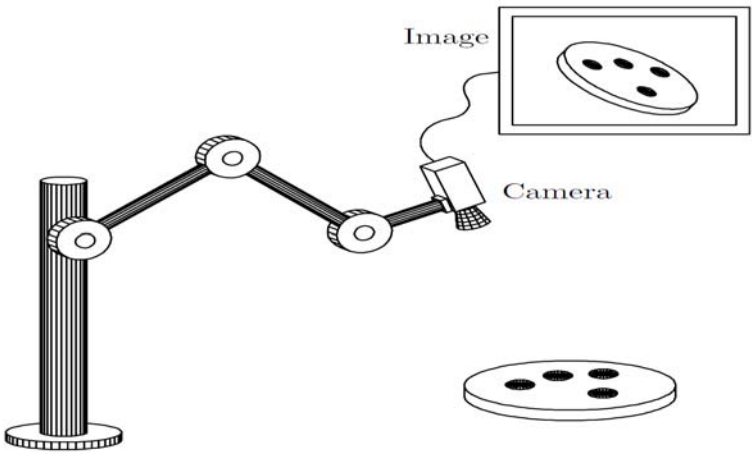
Bu işlemin ardından ***eş zamanlı*** çalışacak olan motorların montajına gelinmiştir. Robot için belki de en kritik olan bölüm burasıdır. Bunun sebebi, olurda yanlış bir pozisyonda bağlanan motorlar uzun süre çalıştırılırsa bu o motorların sonu olabileceğidir.

Yapılacak işlem: Bu kısım daha önce de açıklanmış ancak resmin altında daha faydalı olacağı düşünüldüğünden yinelenecektir. İstenirse bölüm **2.2.6**'ya dönülüp incelenebilir**.**

Eş zamanlı çalışacak 2 servo motorun hareketi ilettiği uzuvların motorlara bağlantı şekli çok önemlidir. Rc servolar aynı marka aynı ürün olsa dahi eş zamanlı olarak iki motoru çalıştırmak ve aynı pozisyona gitmelerini sağlamak için motorlara gerekli sinyalleri pic ile göndermek, hareketi aktaracağı parçalara monte edip durumu gözlemlemek ve gerektiği kadar deneme yapmak gerekir. Atlanacak bu ayrıntı ve iki motoru da en başa veya en sona getirip hareket organlarının montajını yapmak ve uzuvların hareketini gözlemlemeden montaja devam edilmesi halinde motorlar farklı

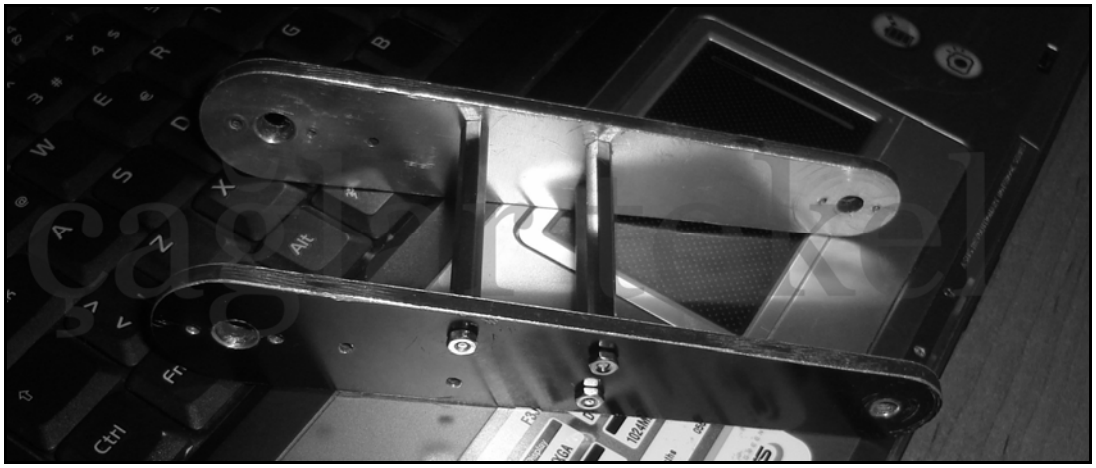
pozisyonlara gitmek isteyeceklerdir. Bu durum ikisinin de çok büyük akım çekmesine sebebiyet verecektir ve bu da motorların aşırı ısınmasına uzun süre bu şekilde çalışılması halindeyse arızalara ve motorun çöp olmasına sebep olacaktır. Bunun için yapılması gereken tabi önce iki motora da yazılımla aynı sinyali yollamak olacaktır. Elektronik ile ilgili bölümde bu kısım ayrıntıyla incelenmiştir. Ardından motorlar aynı pozisyona gitmek için sinyal aldıklarından dolayı beklentimiz iki kolun da aynı pozisyona gidecek olmasıdır.

Fakat bu durum böyle olmayacaktır. Bu fark gözlemlenip motorlardan birinin kafası hareket iletim organından çıkarılmalı ve göz kararı olarak gözlemlenen açı kadar döndürülmelidir. Bu işlem gerektiği kadar tekrarlanmalı ve gönderilen sinyal neticesinde aynı pozisyona geldikleri görüldükten sonra bir sonraki adıma geçilmeli ve aralayıcılar monte edilmelidir.

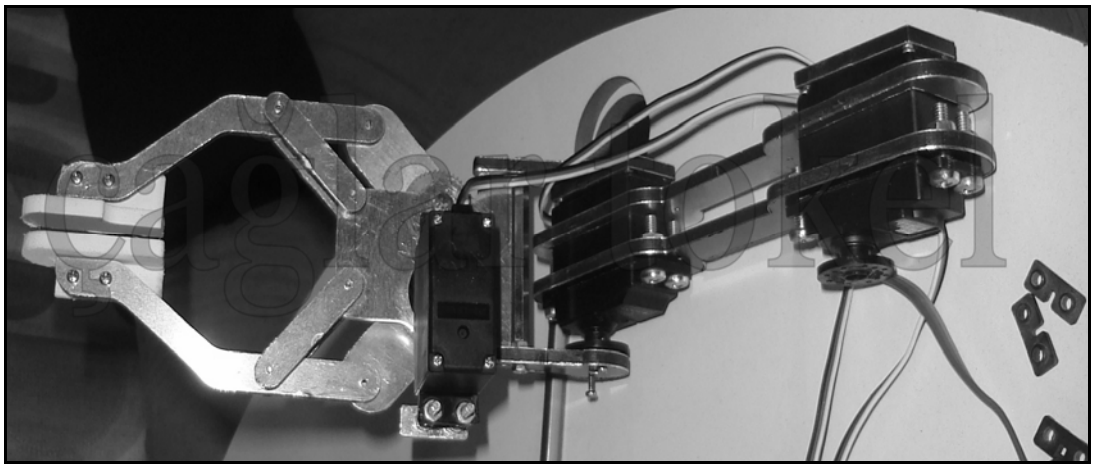


Şekil 2.33 Üç eksenli görüntü işleme yapan bir robot kol benzetimi

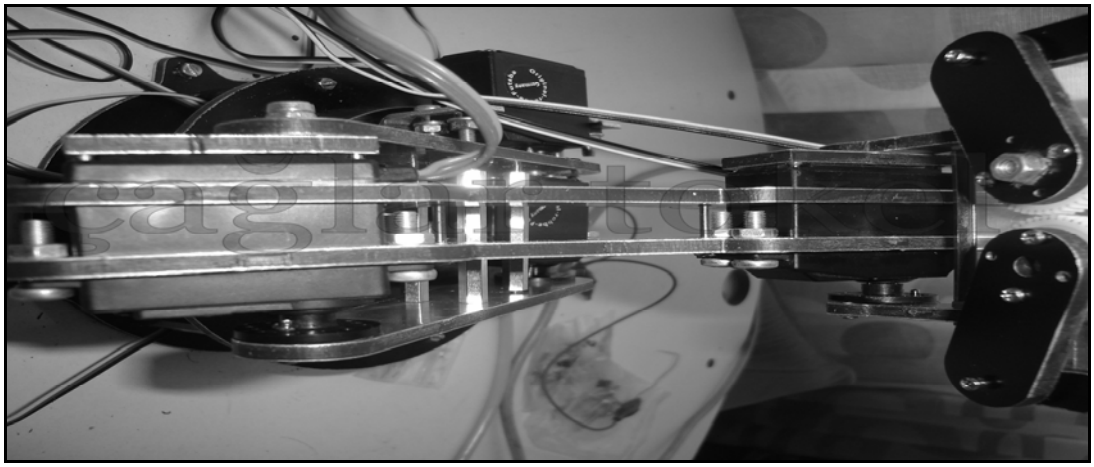
2.3.3 Dirsek ve Bilek Bölgesinin Montajı



Şekil 2.34 Eş zamanlı motorlara sinyal gönderilir,aynı pozisyona geldiklerinde aralayıcı monte edilir.



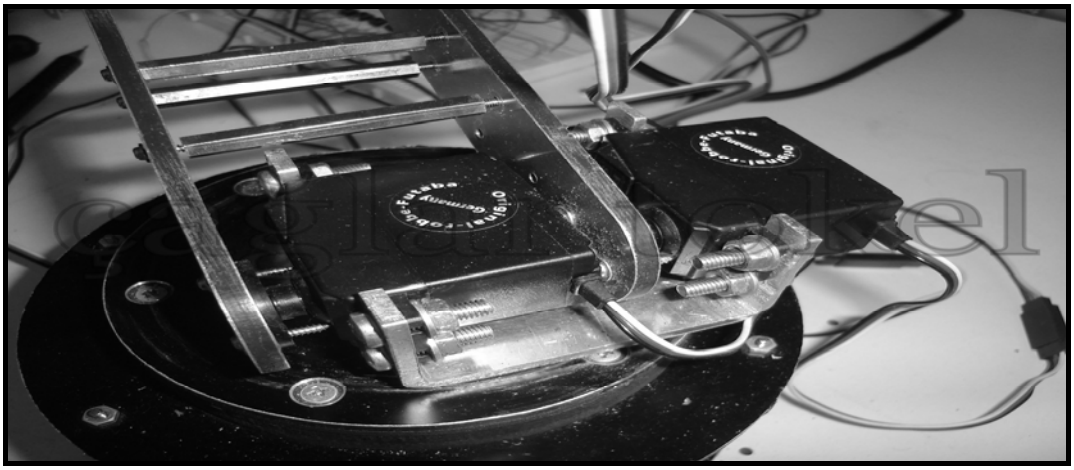
Şekil 2.35 Dirsek ve bilek motorlarının montajının yapıldığı bölüm



Şekil 2.36 Dirsek ve bilek motorlarını taşıyan eş iki parça M2 iki cıvata somun ile birbirine bağlanır

2.3.4 Eş Zamanlı motorların, Bilek ve Tutucunun (gripper) Montajı

Robotun tek tek bütün montaj aşamaları tekrar tekrar ele alınıp en iyi sonucu elde etme yoluna gidilmiştir.



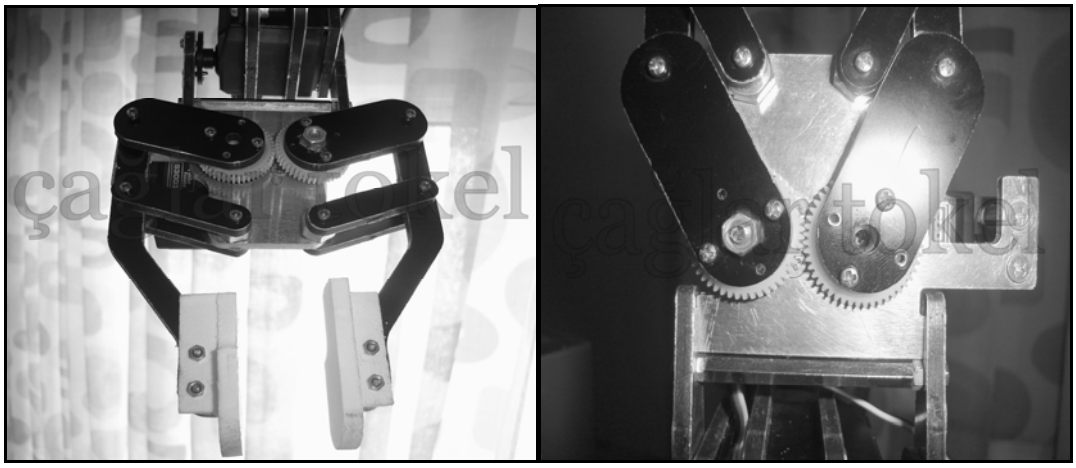
Şekil 2.37 Eş zamanlı motorların gövdeye rondela somun ve cıvata ile monte edilmesi

Şekil 2.38 Robotun bilek kısmının 2.uzuv'a montajı

Şekil 2.39 Robotun tutucusunun motor ile birlikte gövdeye montajı

2.3.5 Tutucunun Dişlilerinin Birlikte Çalışmak Üzere Montajının Yapılması

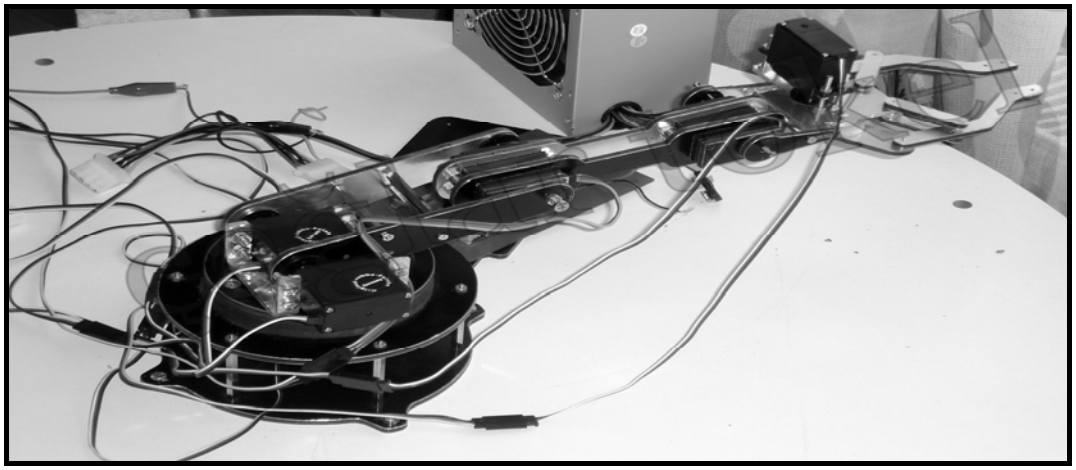
Dişlilerin montajı çok hassas bir işlem olup PWM ile gönderilen sinyaller ayarlanarak tutucuya sadece aç ve kapa işlemi yaptırmakta fayda vardır. Bu sinyaller gönderilip gereken denemeler yapılıp hareketin iletildiği dişlinin somunu gevşetilip sıkılması yoluyla doğru çalışma pozisyonları bulunmalıdır.



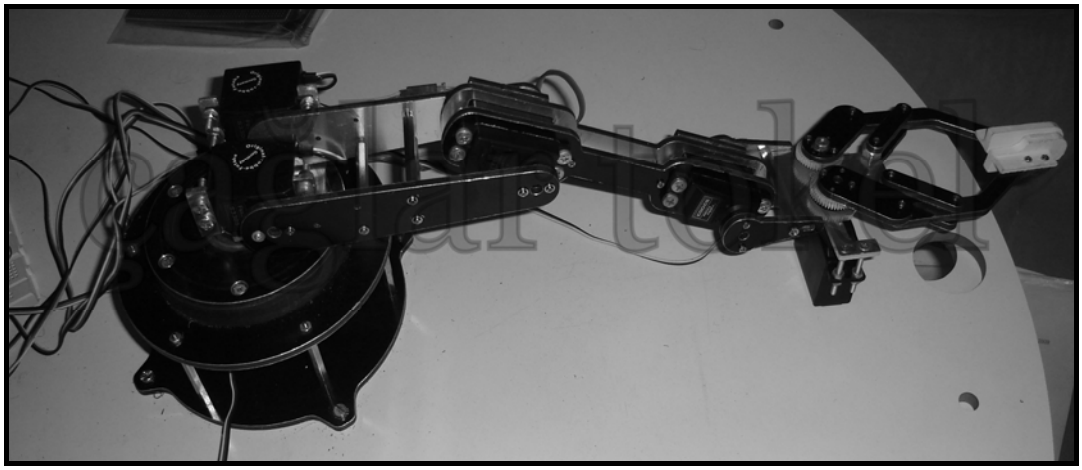
Şekil 2.40 Birlikte çalışacak dişliler sadece aynı modüllerde olmaları durumunda hareketin iletimi gerçekleşebileceğinden bu duruma dikkat edilmelidir. Bunun haricinde tutucunun tasarımı itibariyle çevrim oranının 1:1 olması gerektiğinden iki dişlide (48 diş) aynı diş sayısına sahip olmalıdır.

2.3.6 Robot Manipulatör'ün Montajının Tamamlanması

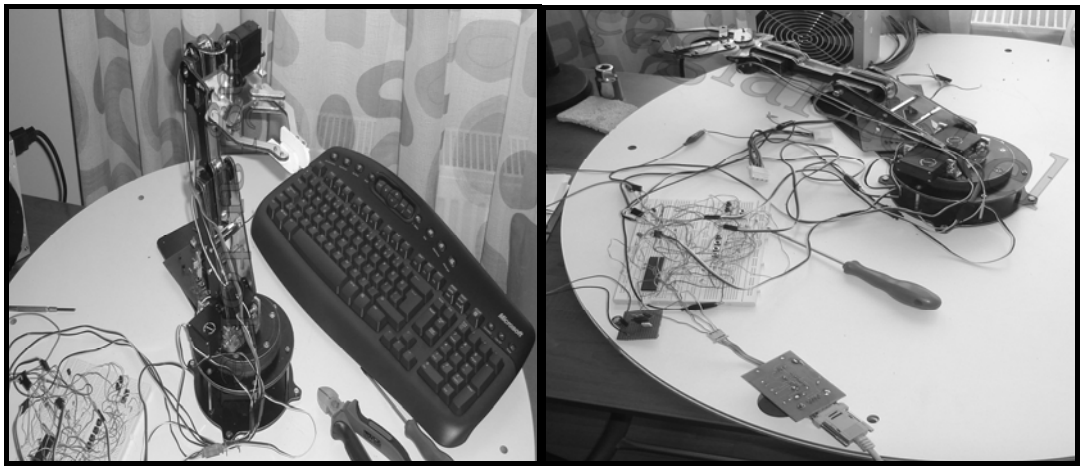
Robot manipülatör montajı tamamlandıktan sonra çalışmalar sırasında gerekli görülen noktalar çalışma esnasında demonte edilip yeniden montajı yapılmıştır. Bu çalışma da eş zamanlı motorların aşırı akım çektiği ve ısındığı saptanarak omuz Dirsek taşıyıcısının montajı düzeltilerek yeniden yapılmıştır. Ayrıca gripper'ın dişlilerinin doğru şekilde çalışmasını sağlamak için de yeniden montaj yapılmıştır.



Şekil 2.41 Robotun mekanik kısmı tamamlanıp yatay pozisyona getirilmesi



Şekil 2.42 Robotun bir objeyi tutmak için pozisyon alırken ki görüntüsü



Şekil 2.43 Robot ekranda görülen klavye ile kontrol edilirken

* Robotun kontrolleri üzerinde çalışırken üstünde değişiklik yapmak için basite indirgenmiş breadboard üzerindeki kontrol devresi

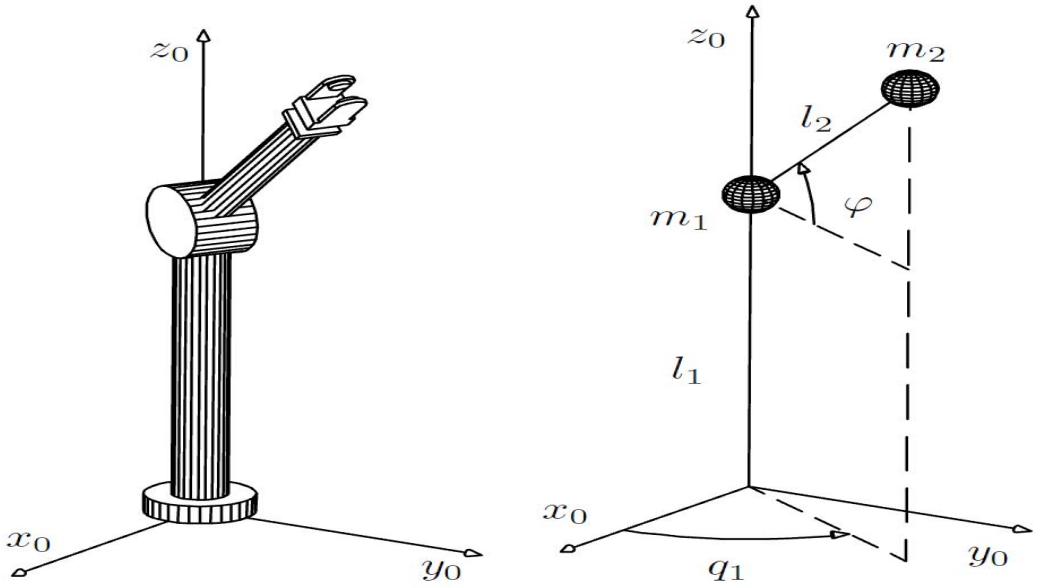
* Robotun seri porttan haberleşmesini sağlamak için yaptığımız RS232 devresi

2.4 Manipülatör ve Gripperlar

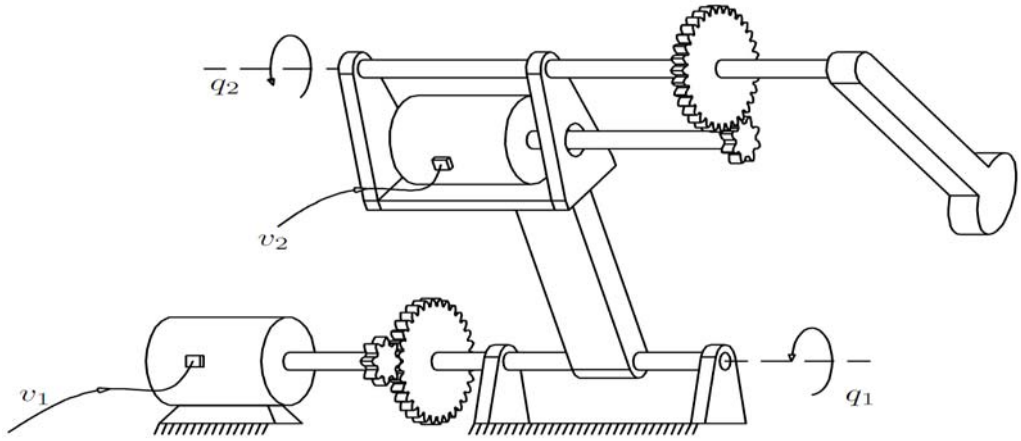
Mekanik tasarım burada sona ermiş olup bu bölümde manipülatör'ler ve gripper'ların (tutucu) yapısı hakkında öz bir bilgi verilmesi amaçlanmıştır.

2.4.1 Manipülatör'ler

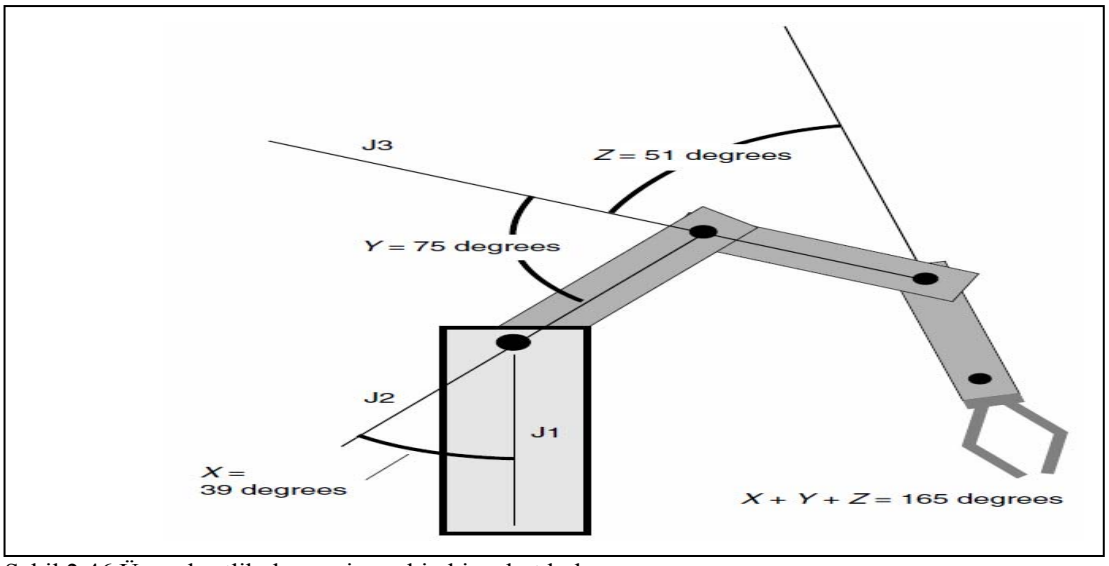
Manipülatörler'i bir tanım yapmak gerekirse, genellikle birden fazla sayıda uzuv'a sahip birbirine bağlı ya da biri diğerine göre kayan ya da diğerine göre dönel bir hareket yapan ve bunu objeleri almak, tutmak, bir yere koymak amaçlı yapan birden fazla serbestlik derecesine sahip bir mekanizma olarak tanımlayabiliriz. Bir manipulatör bir insan tarafından bir bilgisayar aracılığıyla uzaktan kontrol edilebilir.



Şekil 2.44 Tek serbestlik derecesine sahip bir robot kol

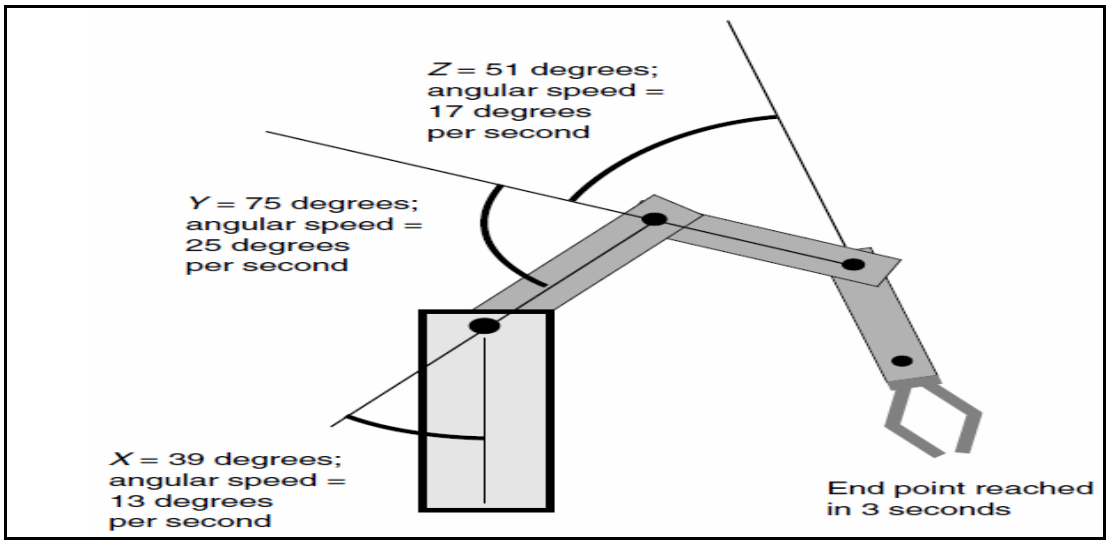


Şekil 2.45 İki serbestlik derecesine sahip bir robot kol



Şekil 2.46 Üç serbestlik derecesine sahip bir robot kol

Bu robot kolu, çalışma uzayı içerisinde belirli bir yere götürmek istendiği zaman operatör bilgisayardan veriler girer. Bu bilgiler X,Y ve Z ölçü açısal değerlerini içerir. Gösterilen bu örnekte X = 39^0, Y = 75^0 ve Z = 51^0 değerleri girilmiştir.

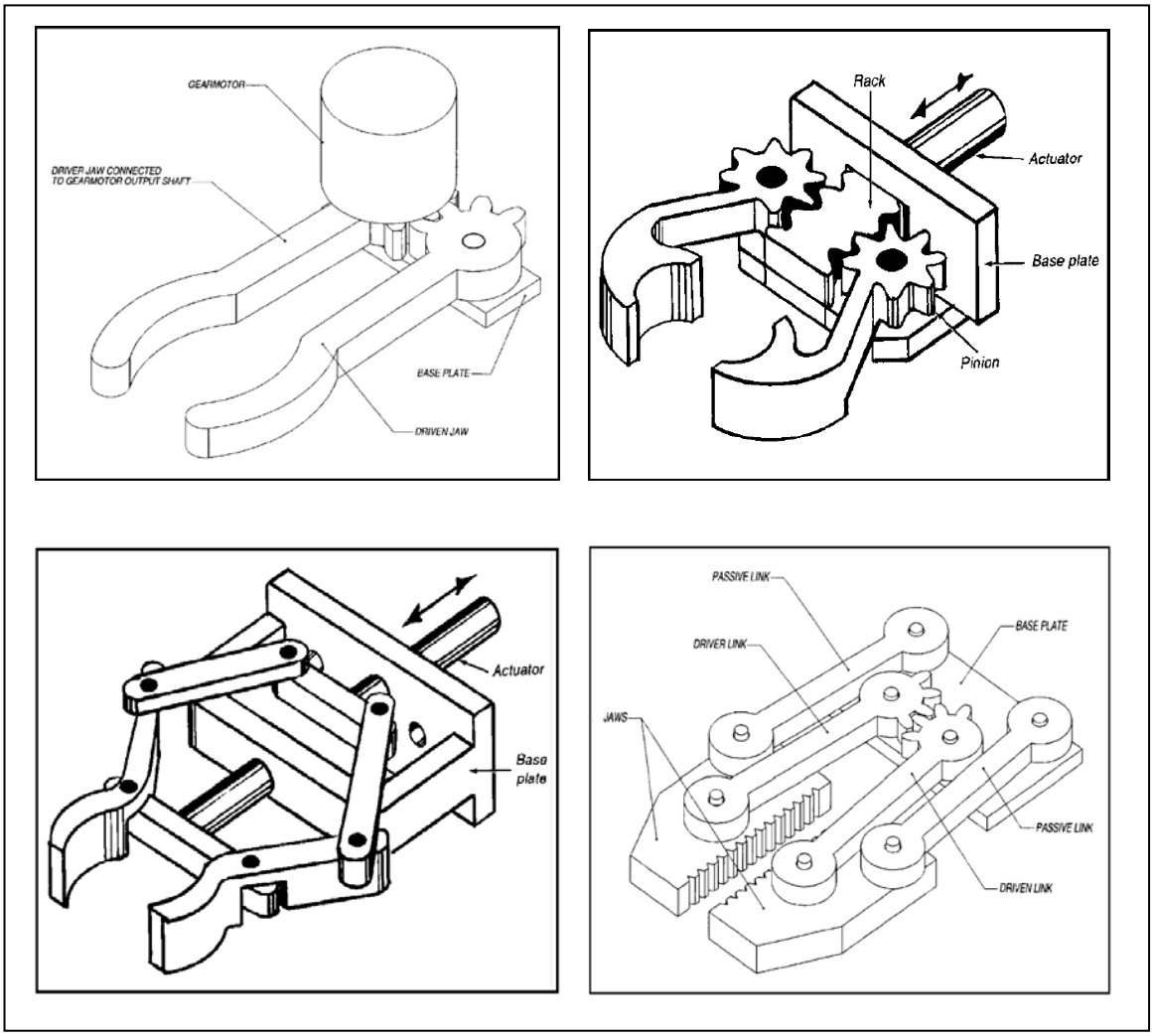


Şekil 2.47 Robotun uç noktasının istenen pozisyona gelmesi için gereken süre 3saniyedir. * Bunun için görüldüğü gibi X,Y ve Z nin üçü de 3 saniyede hareketlerini tamamlamışlardır.

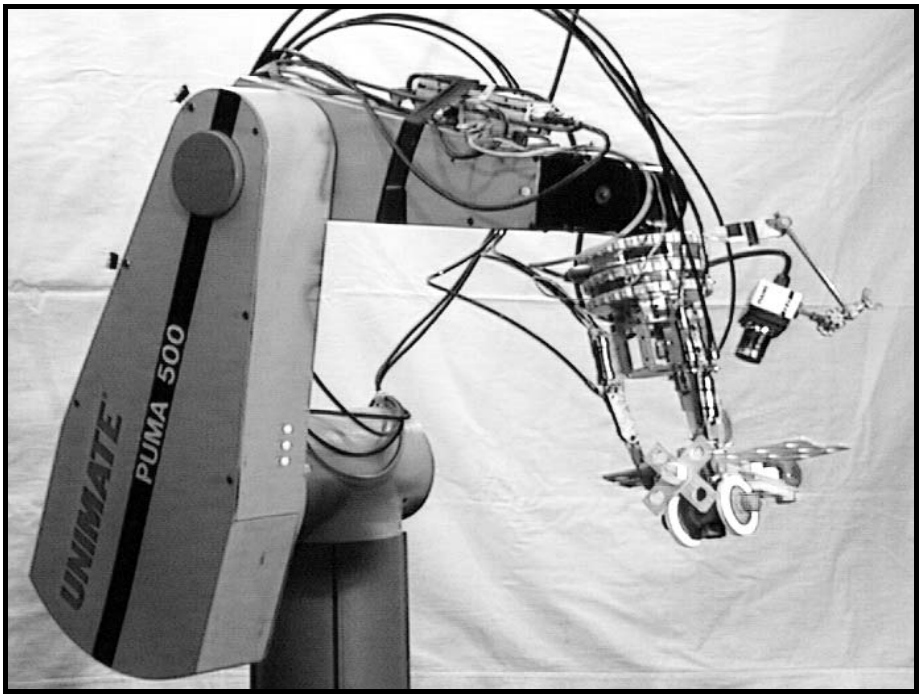
2.4.2 Tutucular (Gripperlar)

Manipülatörlerin uç kısmında robotun bir işlem yapması ve çevreyle bir etkileşiminin olması için tasarlanan parçadır. Bu yüzden genellikle 'end–effector' yani uç etkisi olarak bilinirler. Gripper'ların hareketleri Denavit Hartenberg prensibine göre eksen hareketi olarak tanımlanmazlar. Bu yüzden bir serbestlik derecesi olarak sınıflandırılmazlar. Bu elemanlara end-effector dense de gripper tanımı daha sık kullanılmaktadır. Bunun sebebi ise yapacak olduğu görevi yapması için tutma işlemi yapması gerekmesidir. (Gripper: mandal, pens, pense, tutucu)

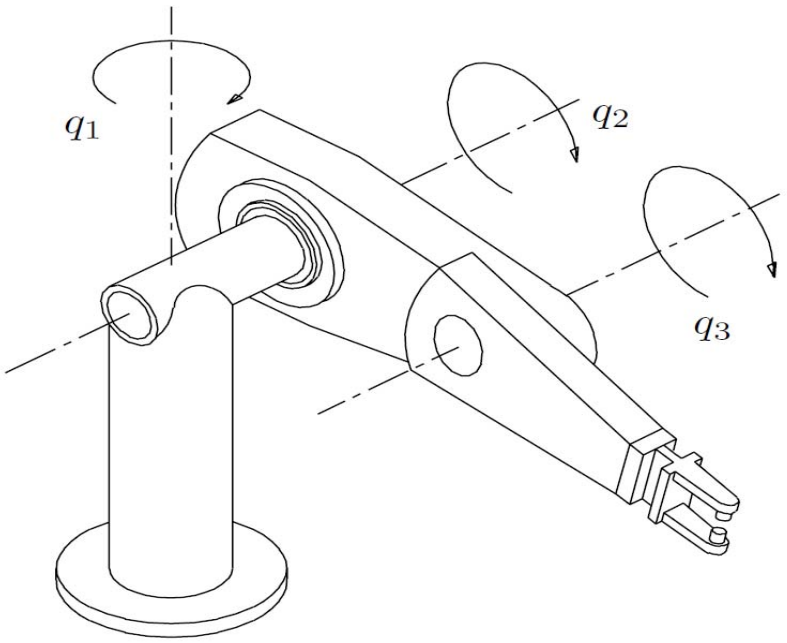
Çoğunlukla bir şeyi almak hareket ettirmek yerini değiştirmek için tasarlanırlar. Yalnız bir gripper'ın almak durumunda olduğu objeye gereğinden fazla yaklaşması tehlikeli sonuçlar doğurabilir. Bu durum robota öğretilmelidir. Bu yüzden sensör'lere başvurulmuştur. Ama buna rağmen gripper'la işlem yapmak zor bir iştir. Bu yüzden tasarım yaparken taşınacak objeler konusunda da yeterli bilgiye sahip olmak gerekmektedir. Ağırlıkları, şekilleri, ölçüleri, karşı gösterdikleri direnç vb. dikkate alınması gereken başlıca faktörlerdir.



Şekil 2.48 Üzerinde durulmuş olan ve bizim tasarımımızda da esinlendiğimiz birkaç gripper geometrisini inceleyelim.

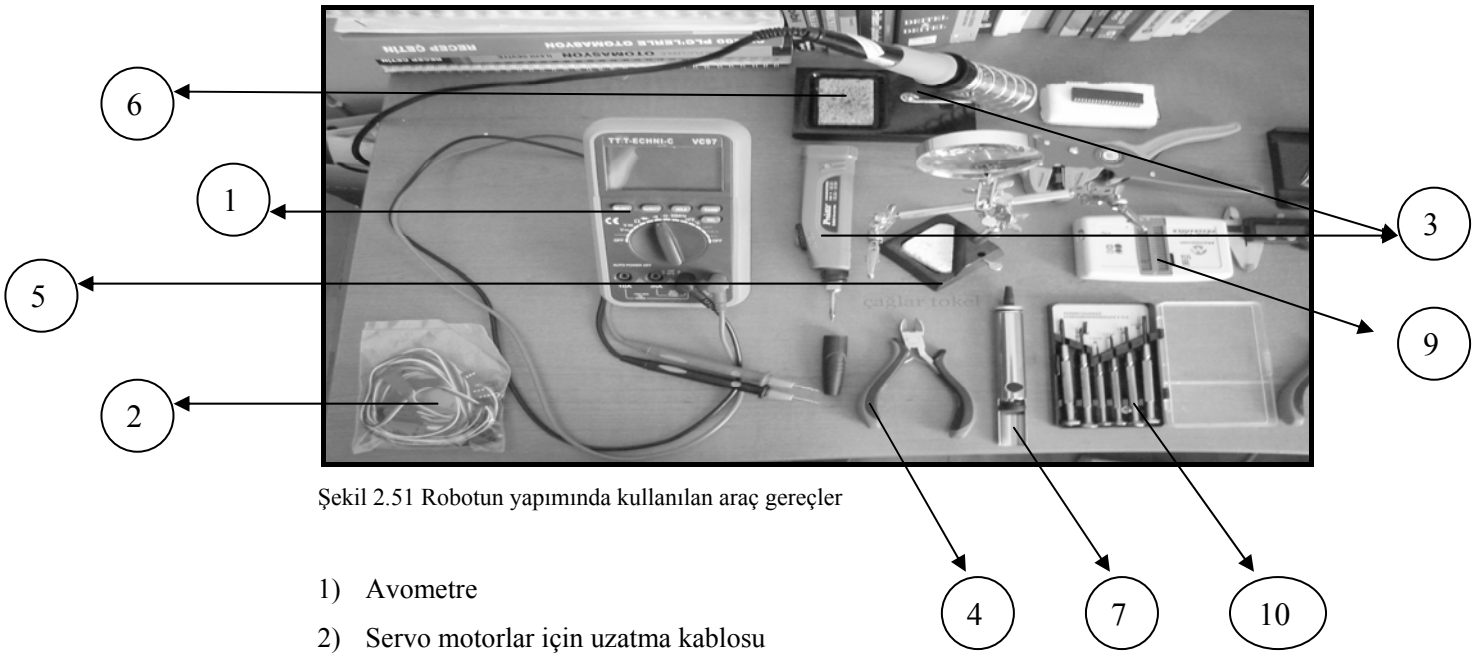


Şekil 2.49 Altı (DOF) serbestlik derecesine sahip end-effector'ünü gözlemleyen bileğe monte edilmiş bir kameraya sahip Puma 500 robotu



Şekil 2.50 Bir puma robotun basit bir 3 serbestlik dereceli hali benzetimi

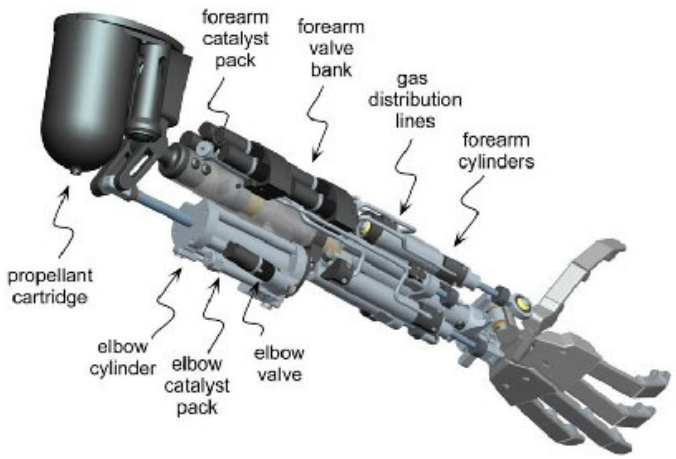
2.5 Çalışmalarda ve Elektronik Kartın Yapımında Kullanılan Malzemeler



Şekil 2.51 Robotun yapımında kullanılan araç gereçler

1) Avometre
2) Servo motorlar için uzatma kablosu
3) Lehim havyası ve veya lehim tabancası)
4) Yan keski
5) Plaket montaj aparatı
6) Lehim altlığı ve temizleme süngeri
7) Lehim pompası
8) Kablo kesici (wire stripper)
9) Pic programlayıcı (Usb'den)
10) Elektronikçi tornavida takımı
11) Hidrojen Peroksit
12) Ovalama Tozu
13) Tuz Ruhu
14) Bakır levha ve delikli pertineks
15) Masaüstü matkap tezgahı ve matkap
16) Manyetik olmayan izoleli elektronikçi cımbızı
17) 9V pil (sadece çalışmalar sırasında kullanılmış, bitmiş haldeki robotun motorları ve pic için gerekli güç, bir pc güç kaynağından sağlanacaktır.
18) Marker kalemi (ütülenme işleminden sonra küçük rütuşlar için kullanılabilir. Bizim kullandığımız teknikte buna gerek kalmamıştır.
19) Kumpas

Şekil 2.52 Robotun elektronik kartlarının yapımında kullanılan araç gereçler



Şekil 2.53 Bir insan kolunun robot koluna benzetim prototipi

BÖLÜM ÜÇ
ROBOTUN ELEKTRONİĞİ

3.1 Robotun Kontrolünde Kullanılan Elektronik Kartların Tasarlanması

Bu bölümde, robota hareket verebilmek için kullanılmış olan RS232 ve servo sürücü kartının bilgisayarda tasarımı yapılmış, PROTEUS programının simülasyon özelliği kullanılarak devreyi kurmadan önce çalışıp çalışmadığı gözlenmiş ardından prototipleme sürecinde breadboard'da çalışmalar sürdürülmüş ardından kartların üretimine geçilmiştir. Bu bölümde ayrıca asenkron seri iletişim yöntemine bir giriş yapılacaktır.

3.1.1 Elektronik Kartların Tasarım Aşaması

PCB'ler (printed circuit board) günümüzde evlerimizde kullandığımız çamaşır makinesi, dijital kapı zilleri, uzaktan kumandalı oyuncaklar, cep telefonları, bilgisayar içinde bulunan ana kart, ekran kartı, ses kartı, endüstriyel alanda kullanılan akıllı röle ve PLC'ler, uzaktan kumandalar ve bunun gibi sıralayabileceğimiz bire bir kullanım alanımız içinde olan her tür makinenin içinde rastlayabileceğimiz duruma gelmişlerdir.

Bugün gelişmiş ülkeleri, gelişmekte olan ülkelerden ayıran şey birinin üretici diğerinin uygulayıcı olmasıdır. Artık elektronik, bilgisayar ve makine iç içe yaşamaktadır. Ülkemiz sınırları içinde PCB kart tasarımı yapan ve üreten birçok üretici mevcut olsa da halen ileri düzey kart basımı konusunda yeterli olmamaktadırlar. Devreler ve elektronik konusunda yeterli donanıma sahip birinin kendi kartını kendi tasarlaması ve üretmesi hem zaman hem maddi açıdan oldukça gerekli bir iştir. Kendi tasarladığınız bir kartı bu firmalarda üretilen kartlar gibi profosyonel bir ürün haline getirmeniz mümkün olmasa da tasarladığınız her kartı bir baskı devre firmasına götürmek hem zaman hem para kaybı olacaktır.

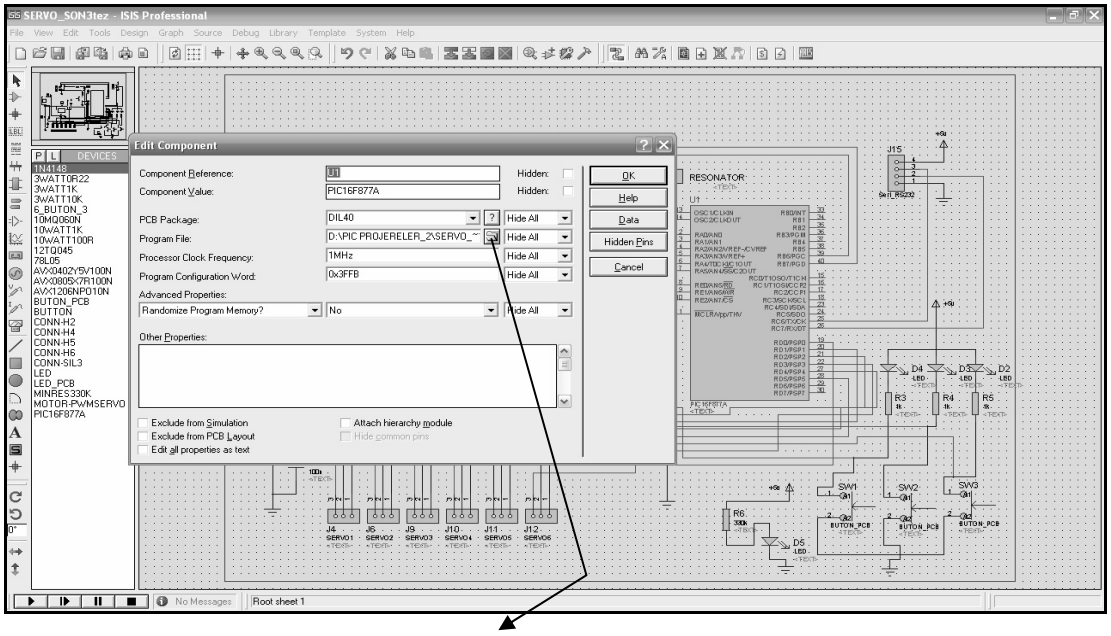
AMAÇ :

Bizim amacımız kendi elektronik kartlarımızı kendimiz imal etmek olmuştur. Tabi ki bu bir tercih meselesi olup kişiye kalmıştır.

Öncelikle bu işlemler için yapılması gerekenler tabi ki yeterli bir elektronik donanımı, ilgi araştırma ve gerekli programları kullanabilmektir. Projenin elektronik kısmını uygulayabilmek için proteus isis, ares, visual basic programlarını ve Jal veya C dillerini kullanabilmek aynı zamanda mikrodenetleyiciler konusunda yeterli donanıma sahip olmak gerekmektedir. Jal dili mikroişlemcilerle ilgilenen herkesin tercih etmeyebileceği bir dil olması itibariyle kodlarda ne gibi değişiklikler yapılıp nasıl C diline çevirileceği anlatılacaktır.

ÖZET:

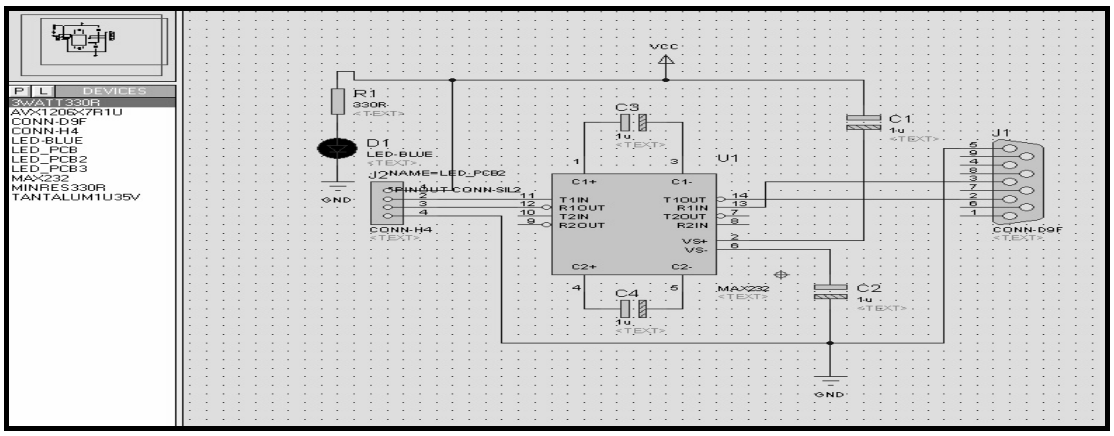
Robotun sürücü kartını tasarlamak için Proteus isis programını kullanıldı. Ardından devre şemasının çıkarıldığı Ares programında hazırlanan şemanın çıktısı lazer yazıcıdan alındı. (özel transfer kağıdına)



Şekil 3.1 Proteus'da simülasyon aşamasında mikroişlemciye 'hex' kodunun atılması için çıkan menü

3.1.2 Seri Porttan Haberleşme Devresinin Tasarımının Yapılması

Aynı bir bilgisayar ya da tv aldığımızda içinden kullanım klavuzu çıkması gibi her elektronik entegrenin ve elemanın bir datasheet'i vardır. Yukarıda Proteusta hazırladığımız devre görülmektedir. Pin bağlantılarını yaparken dikkat edilmesi gerek bu datasheet'in incelenmesi ve bu doğrultuda işlem yapmaktır. Led-blue isimli komponent, seri portun haberleşme durumunda olup olmadığını gözlemlemek için konulmuştur. Burada tek tek proteus programında işlemlerin nasıl yapıldığı açıklanmayacak olup istendiği takdirde piyasada bulunan kitaplar temin edinilerek ya da kendi kendine kafa yorarak öğrenmek yoluna gidilebilir.



Şekil 3.2 RS232 devresinin ISIS ortamından alınmış ekran görüntüsü

Bu kısımda RS232 nedir ne işe yarar üzerine değinilmesinde fayda var. İlerde asenkron ve senkron iletişim (usart) üzerine değinildiğinde bu konuya tekrar atıfta bulunulacaktır.

RS232, bilgisayarımızın seri portuyla haberleşme yoluyla veri alışverişi yapmamızı sağlayan bir devredir. PIC 5V ile çalışan bir entegre olduğundan ve bizim pc seri portumuz ise 12V ile çalıştığından dolayı bu voltaj farkını kapatmak gerekmektedir. RS232'nin amacı bu olup pc'nin seri portundaki 12V 'u 5V'a düşürmek, PIC'in çalıştığı 5V'da 12V'a çıkarmaktır. Seri portumuz ile PIC'imiz aynı voltajda olacak ve bu sayede hiçbir sorun olmadan çalışabileceklerdir.

Seri portla haberleşmek için Windows'un kendi programı olan hyperterminal kullanılabilir. Bu programa Başlat – tüm programlar – donatılar – iletişim hyperterminal bağlantısından ulaşılabilir.Ancak seri porttan haberleşmemizi visual basic programı aracılığıyla yapacağız. Visual basic'te 'Mscomm1' elemanını bunun için tasarım ortamında forma eklememiz gerekecek. Bu konuya yeri geldiğinde değinilecektir.

3.1.3 Seri İletişim

Seri iletişim gönderilecek ve alınacak olan verinin birer hat üzerinden belli bir protokol esasına göre yapılması işlemidir. Paralel iletişime göre pin sayısının az olması, daha basit bir donanım kullanılarak iletişimin sağlanması avantajlarıdır. Ama bunun yanındaki dezavantajı ise yazılımın paralel iletişime göre daha zor olmasıdır. Çünkü veriler paralel hatlar boyunca gitmeyecek paralelden seriye dönüştürülüp, seri olarak birbiri ardından bit bit gidecektir. Bu yüzden yazılımda zamanlama ve düzgün programlama yapmak çok büyük önem taşımaktadır. Nasıl ki veriler paralelden seri bilgiye dönüştürülerek yollanıyorsa gönderilen veriyi alacak olan sistemde bu seri bilgiyi paralele dönüştürmek durumundadır. Bu da yine yazılımla halledilmesi gereken bir durumdur.

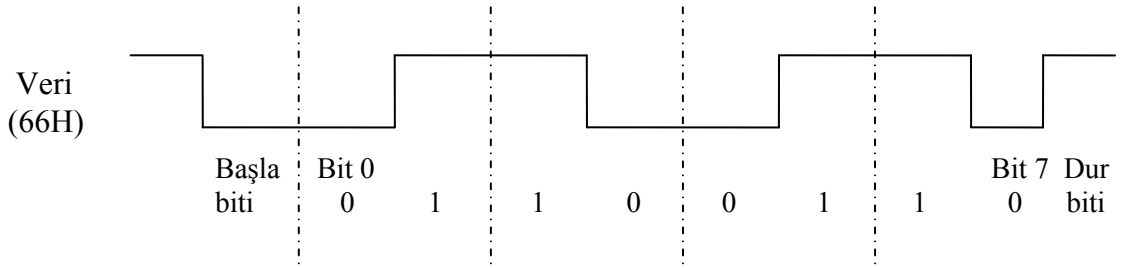
Seri veri iletişiminde asenkron ve senkron olmak üzere iki yöntem kullanılmaktadır. **Senkron** iletişimde, bir veri paketi transferi saat darbeleri (clock palsleri) ile senkronize edilerek iletilirken **asenkron** iletişimde bir byte'lık (8 bit) veri belirlenmiş saat dilimlerinde bit bit iletilir.

Her iki iletişim yazılımla yapılabileceği gibi donanımla da yapılabilmektedir. Hangi yöntem kullanılırsa kullanılsın, yazılım ile yapılan seri veri iletişimine SFR'siz (special function register) seri iletişim yöntemi adı verilir. Biz robot manipulatör ile seri porttan haberleşme yaparken bu yöntemi değil USART (Universal Synchronous Asynchronous Reciever Transmitter) metodunu kullandık.

3.1.4 SFR'siz Asenkron Seri İletişim

Bu yöntemde gönderdiğimiz verinin başında bir BAŞLA biti sonunda da DUR biti bulunmaktadır. Bu iki bit sebebiyle hız senkron iletişime göre biraz daha yavaştır. Fakat az hat kullanması sebebiyle bu yöntem seri iletişim gerçekleştiren sistemlerin çoğunda tercih edilmiştir. (Klavye, fare, modem,vb..)

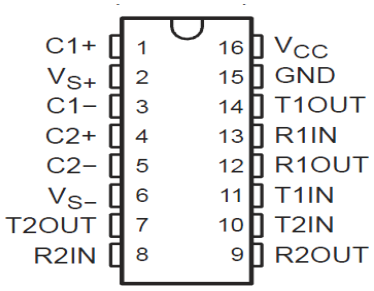
Asenkron veri iletişiminde karakterler ortak bir kod ile gönderilir. Bu kod genellikle ASCII (Amerikan Standard Code for Information Interchange) adı verilen bir sistemdir. Şimdi ASCII kodunda bir veri yollayalım.



Şekil 3.3 Asenkron veri iletişimi ile (66H =) 'B' karakterinin yollanması

B karakterinin ASCII kodu hexadecimal olarak 66'dır ve o da

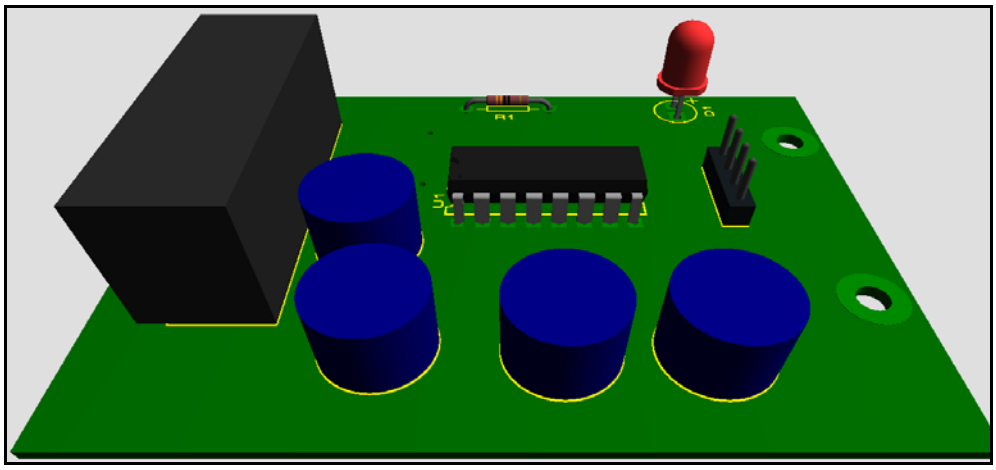
İkili sistemde 0 1 1 0 0 1 1 0 dır.



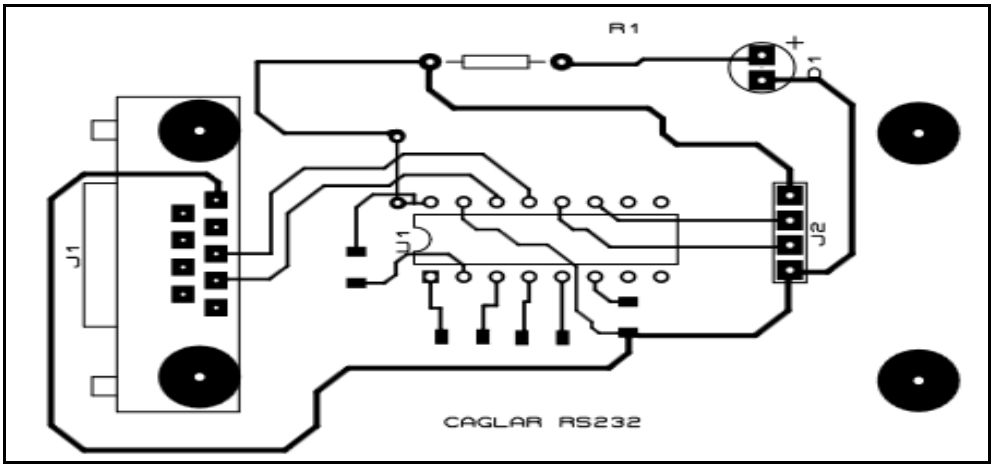
Şekil 3.4 MAX232 entegresi
pin tablosu grafiği

3.1.5 Lazer Çıktısı Alınacak RS232 Kartı Tasarımı ve Baskı Devresi

Bu bölümde bilgisayarın 12V'u ile PIC'in çalışma gerilimi olan 5V arasındaki uyumsuzluğu gidermek ve asenkron seri iletişim protokollerini kullanmak için RS232 devresi tasarımı ve üretimi yapılmıştır. Seri porttan iletişim kurduğumuzu gözlemlemek üzere devrede bir adet Led mevcuttur.



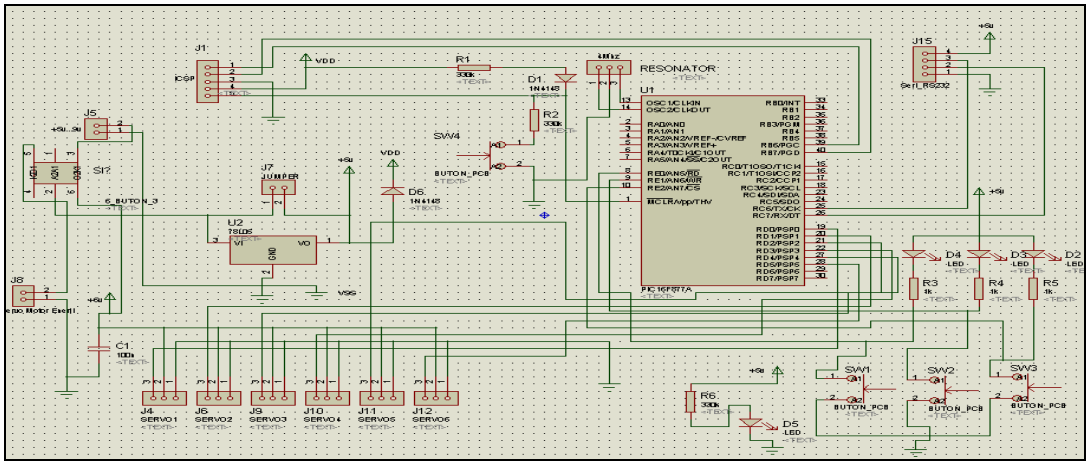
Şekil 3.5 RS232 devremizin ARES ortamından alınmış ekran görüntüsü



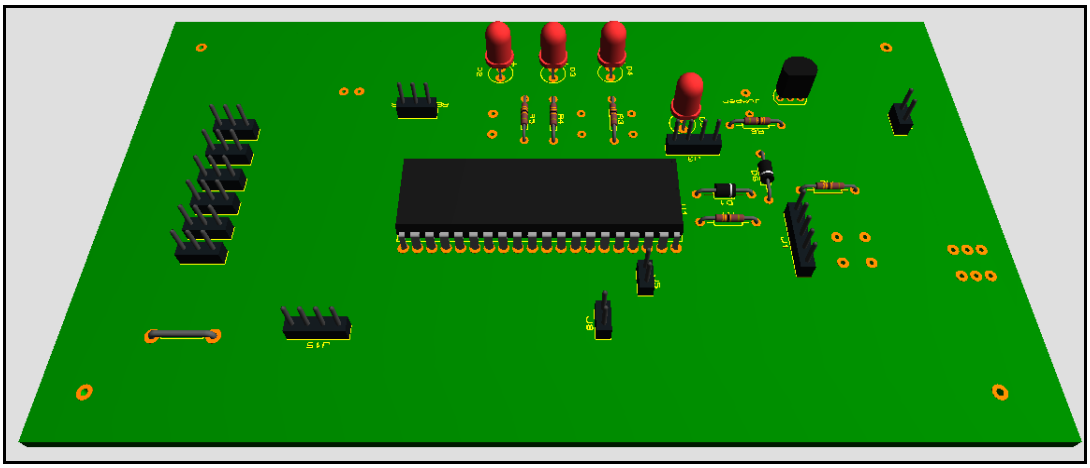
Şekil 3.6 Lazer çıktısı alınan RS232 baskı devremiz

3.1.6 Lazer Çıktısı Alınacak Rc Servo Sürücü Kartı Tasarımı ve Baskı Devresi

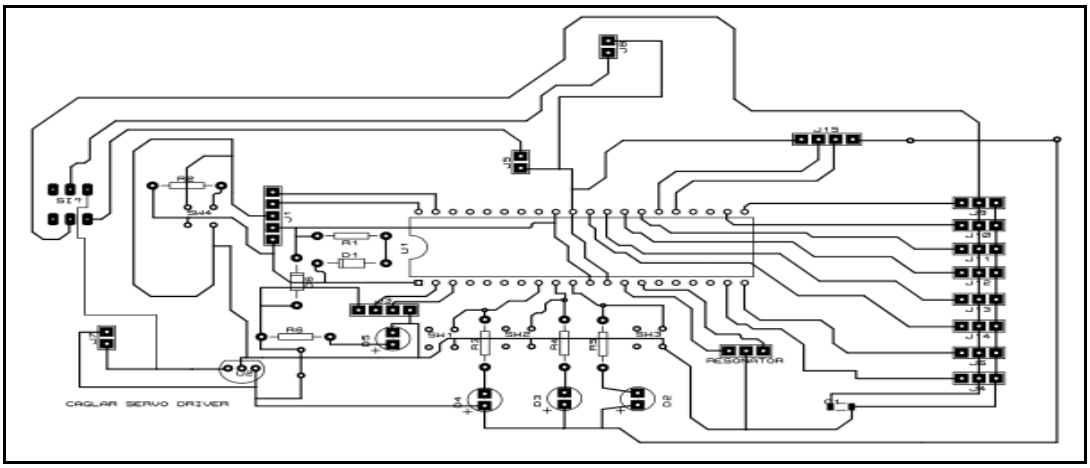
Rc servo motoru sürmek için baskı devre tekniği ile üretilen bir sürücü kartı tasarımı yapılmış bu versiyonu 6 adet rc servo motoru sürebilmektedir.



Şekil 3.7 ISIS ortamından alınmış 6 adet Rc servo motor sürebilen devre şeması



Şekil 3.8 Robotun 6 adet Rc servo motorunu sürmeyi sağladığımız sürücü kartı tasarım simülasyonu



Şekil 3.9 Altı adet Rc servo motoru sürmesi için tasarlanan sürücü kartı iletim hatlarının görünümü

BÖLÜM DÖRT
MİKRODENETLEYİCİNİN TANITILMASI

4.1 Kullanılan Mikrodenetleyicinin Tanıtılması

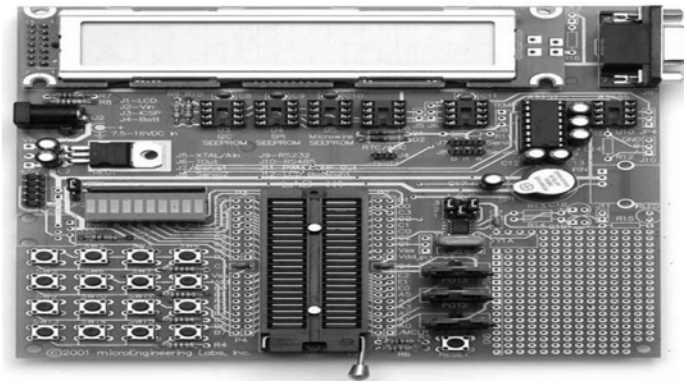
Bu kısımda bilgi karmaşasını önlemek ve bu bilgilere ulaşılabilecek yeterli kaynağın olması sebebiyle mikroişlemciler ve onları programlamak için kullanılan diller ve derleyicilerin anlatılmasından çok bu projeyi gerçekleştirebilmek için kullanılan bölümlerin anlatılması yoluna gidilmiştir. Bu bölüm bilgilendirme amaçlı olup gerek görüldüğü taktirde robotun yapım sürecine devam etmek için atlanıp
5.bölüme geçilebilir.

4.1.1 Mikrodenetleyiciler

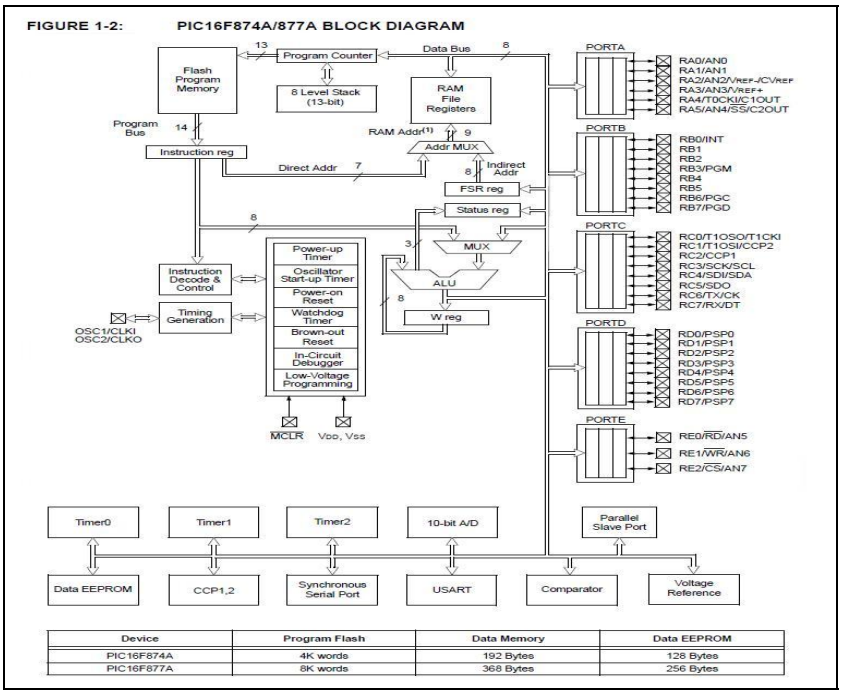
Bir mikrodenetleyici bir PC' nin özelliklerine sahip minik bir kopyasıdır. Aynı bilgisayarların yaptığı gibi komutları belleklerinden okurlar ve işlerler. Bir mikrodenetleyici basitçe bir ana işlemciden, giriş çıkış portlarından, programlama ve veri depolama için bellek, bir iç saat, zamanlayıcı, geri sayıcı, analog dijital çevirici, seri iletişim birimleri ve bekçi köpeği devrelerinden oluşur.
Bugün dünyadaki mikroişlemcilerin hepsi 8 ile 32 bir aralığındadır. Dünyadaki başlıca mikrodenetleyici üreticileri Intel (8051), Zilog (aynı mikrodenetleyicinin türevi denilebilecek Z-80 mikrodenetleyicisi), Motorola (68HC05), Atmel (AVR), Parallax (BASIC stamp) ve bizim de projemizde kullandığımız Microchip firmasının PIC ailesinin orta sınıf bir mimariye sahip mikrodenetleyicisi olan PIC16F877A'dir.

4.1.2 PIC16F877A 'nın tanıtılması

Orijinal olan General Instruments firması tarafından geliştirilen PIC1650'dir. Bu alete "Programmable Intelligent Computer" dolayısıyla PIC denmiştir. Ancak günümüzde PIC ismi, "Programmable Interface Computer" ile ilişkilendirilmiştir. Microchip firması marka ismi olarak PIC' i kullanmaktansa PICmicro'yu tercih etmektedir. Bunun popüler olan açıklaması PIC isminin Almanya'da kayıtlı bir marka olması ve bu yüzden uluslar arası olarak bu ismi kullanmayı tercih etmemesidir.



Şekil 4.1 Microchip firmasının ürettiği bir geliştirme kartı

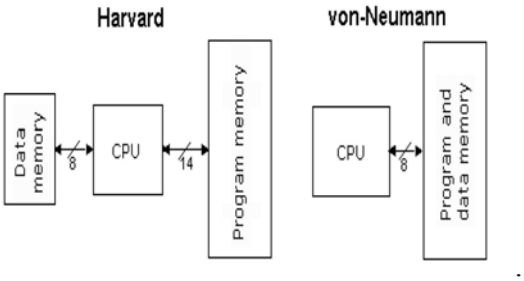


Şekil 4.2 Harward mimarisine göre tasarlanmış PIC16F877A'nın diyagramı

4.1.3 Harvard mimarisi

Pic mikrodenetleyicileri geleneksel von Neumann mimarisini kullanmayıp farklı bir donanım mimarisi kullanırlar ve bu mimariye de Harvard mimarisi denir. Orijinal olarak Harvard mimarisi verinin ve komut setlerinin farklı yollar ve depolanma alanları kullanan bir bilgisayar mimarisini temel almıştır. Başka bir deyişle veri ve komut seti aynı hafıza birimine koyulmazlar.

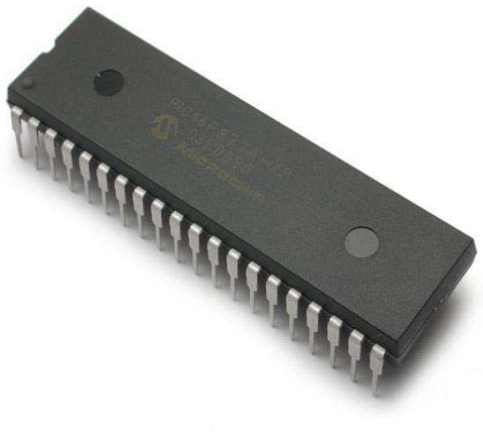
Geleneksel von Neumann mimarisinin bir handikapı (eksisi) işlemcisinin, komut setlerini ve bir veriyi aynı anda okuyabilir veya yazabilirken bu iki işlemi bir arada yapamamasıdır. Bunun sebebi verinin (data) ve komut setlerinin (instructions) aynı sinyal hatlarını kullanmasıdır.



Şekil 4.3 Harvard ve von-Neumann Mimarisi

Diğer taraftan Harvard mimarisine sahip bir mikrodenetleyicide işlemci komut setlerini ve veriyi (data) aynı anda hafızadan hem okuyabilir hem de aynı anda hafızaya yazabilir. Bu daha hızlı ve daha kompleks bir mikrodenetleyici demektir.

Harvard mimarisi üzerine en yakın zamandaki tartışma ana hafızaya erişim hızı olmuştur. CPU'nun hızını arttırırken hafızaya erişim aynı hızda kaldığında özellikle de çok sayıda hafıza girişi gerektiğinde bu durum kazancı düşürmektedir.



Şekil 4.4 PIC16F877A-I/P Mikrodenetleyicisi

4.1.4 RISC (Azaltılmış Komut Seti)

Microchip firmasının ürettiği mikrodenetleyiciler sahip oldukları Harvard mimarisi sayesinde program ve veri için birbirinden ayrı depolama birimlerine sahiptirler. Ve bu mikrodenetleyiciler RISC ‘e (Reduced instruction set) sahiptirler. Bu sayede 35 komuttan oluşan komut setiyle bir pic’i programlamak mümkün olmaktadır.

“CISC tasarımı (complete instruction set computer) düşük seviyeli her bir komut setinin birden fazla operasyon yapması üzerine kurulmuştur. Mikrodenetleyici programını genel olarak mikrodenetleyiciye belirli bir iş yaptırmak için verdiğimiz komutların oluştuğu bir grup olarak tanımlayabiliriz. Bu komutlar mikrodenetleyicinin hafıza bölümüne kaydedilir ve sırayla hepsi işlenir.

PIC16 serisi mikrodenetleyiciler daha önce de söylemiş olduğumuz gibi 35 komuta sahiptirler. Bu sayı sınırlı gözükmesine rağmen teorik olarak 35 komut ile 350 komut üzerinde komuta sahip bir Pentium 4 işlemci ile yapılabilen tüm işlemleri yapmak mümkündür.”

‘H.Şahin, A.Dayanık ve C.Altınbaşak (2006)’

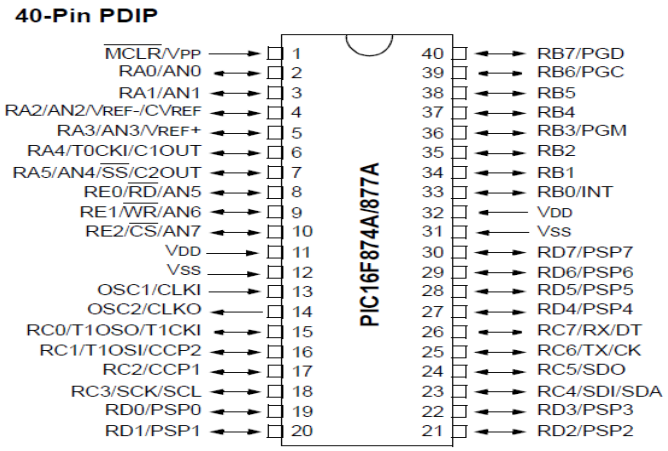
4.1.5 Giriş Çıkış Portları (I / O)

PIC’in pinleri dış dünyayla çeşitli işler yapması için özelleştirilmişlerdir. Biz projemizde çeşitli amaçlar için tüm portları kullandık ancak bir proje yapılacağı zaman illa ki bütün pinleri kullanmaya gerek olmayabilir. Bu portlar hakkında gerekli olan açıklamalar yapılacak olup daha fazla detay için 16F877’nin datasheet’i veya gerekli kitaplar incelenmelidir.

16F877A ‘nın PortA, PortB, PortC, PortD ve PortE olmak üzere 5 tane portu vardır. **Ekler** bölümündeki elektrik karakteristikleri tablosu incelenirse PIC’in bütün pinlerinin **çıkış** (output) olarak seçildiği takdirde dışarıya 25mA akım verebilmekte olduğu (source), veya 25mA akımın içeriye akmasına (sink) olanak sağladığı görülür. Giriş olarak seçildiğinde **Schmitt-trigger** olan RA4 pini dışındaki A portunun **TTL** seviyesine yapılan girişlerde 0V - 0,8V arası “lojik 0” olarak pic

tarafından bize garanti edilmiştir. 0,8V - 2V arası ise kararsız bölgedir. Yani bu aralıkta mikrodenetleyicimizin giriş olarak ayarladığımız bu pinlerine bir gerilim uygulandığında 0' dan 1' e bir geçiş olabilir. TTL 'de 2V - 5V arasında ise "lojik 1" değerini alır.

RA4 pini ise kararsız bir durumun olmadığı bir A portu pinidir. Bunun sebebi bu pinin Schmitt trigger yapıda olmasıdır.RA4 1V'dan düşük gerilimleri "lojik 0" , 1V'dan yüksek olan gerilimleri ise "lojik 1" olarak kabul eder. Burada bahsedilen Schmitt trigger ve TTL (transistör transistör lojik) yapısı ve farkları Bölüm 4.1.7 de grafik yardımı ile daha detaylı olarak anlatılacaktır.



Şekil 4.5 PIC 16F877A mikrodenetleyicisi pin tablosu

4.1.6 PORTA VE TRISA

Bu bölüm 16F87XA'nın datasheetinden faydanılarak gerekli yerlerde yorum ve açıklamalar yapılarak oluşturulmuştur.

PortA, 6 bit genişliğinde çift yönlü bir porttur. PortA'nın pin yönlendirme yazmacı TrisA'dır. Yani TrisA bitini "1" yaptığımızda PortA'nın ilgili pini giriş olacaktır. Şimdi bu porta bağlı olan sürücüyü yüksek empedans moduna sokarak giriş olarak ayarladığımız A portuna bir high sinyali (lojik 1) yollayıp mikrodenetleyiciye iş yaptırabiliriz. TrisA bitini "0" yaptığımızda bu yazmacın bağlı olduğu PortA'nın ilgili olduğu pini çıkış yapmış oluruz. Bu sayede çıkış olarak ayarladığımız pinimize bağladığımız röle motor led gibi elemanları sürmeye hazırız.

Emin olmamız gereken tek şey giriş olarak kullandığımız portumuzu Tris yazmacını kullanarak giriş olarak ayarlamayı unutmamaktır.

Tablo 4.1 A Portu pinlerinin fonksiyon tablosu

Name	Bit#	Buffer	Function
RA0/AN0	bit 0	TTL	Input/output or analog input.
RA1/AN1	bit 1	TTL	Input/output or analog input.
RA2/AN2/VREF-/CVREF	bit 2	TTL	Input/output or analog input or VREF- or CVREF.
RA3/AN3/VREF+	bit 3	TTL	Input/output or analog input or VREF+.
RA4/T0CKI/C1OUT	bit 4	ST	Input/output or external clock input for Timer0 or comparator output. Output is open-drain type.
RA5/AN4/$\overline{SS}$/C2OUT	bit 5	TTL	Input/output or analog input or slave select input for synchronous serial port or comparator output.

Mikrodenetleyicilerin **Tablo 4.1**' de görüldüğü gibi pinlerinin birden fazla özelliği vardır. Bu yüzden bu özellikler arasında seçim yapmak için bir donanıma sahiptirler. A portunda A/D çevrim yapabilen yani bağlı bulunan bir sensör'den okunan analog sinyali 0-256 aralığında değişen dijital sinyale çeviren dönüştürücü bulunmaktadır. 16F877A'nın **Tablo.4.1**' deki pin tablosu incelenirse bu çevrim sırasında referans alınacak voltaj değerlerinin de RA2 ve RA3 pinlerinden girilebileceği görülebilir.(Vref+,Vref-). Aynı zamanda karşılaştırıcı (comparator) içinde referans değer bu pinlerden girilir. Bizim için önemli olan A portunun bu A/D dönüştürücü özelliğini kullanmayacağımızdan dolayı bu seçimi yazılımda kaldırmaktır.

4.1.7 PORTB VE TRISB

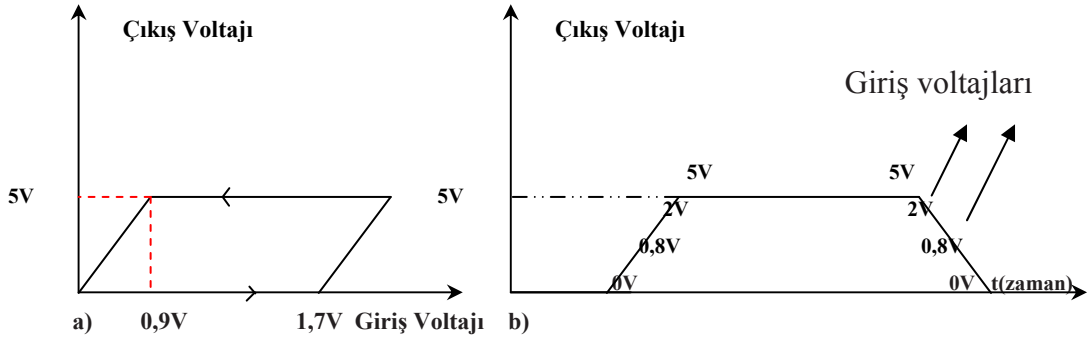
Bu bölüm 16F87XA'nın datasheetinden faydanılarak gerekli yerlerde yorum ve açıklamalar yapılarak oluşturulmuştur.

PortB 8 bit genişliğinde çift yönlü bir porttur. PortB'nın veri yönlendirme yazmacı TrisB'dır. Yani TrisB bitini (=1) yaptığımızda PortB'nın ilgili pini giriş olacaktır. Şimdi bu porta bağlı olan sürücüyü yüksek empedans moduna sokarak giriş olarak ayarladığımız B portuna bir high (lojik 1) sinyali yollayıp mikrodenetleyiciye iş yaptırabiliriz. TrisB bitini (=0) yaptığımızda bu yazmacın bağlı olduğu PortB'nın ilgili olduğu pini çıkış yapmış oluruz.

Tablo 4.2 B Portunun TTL/ST seviyedeki pinlerinin fonksiyon tablosu

Name	Bit#	Buffer	Function
RB0/INT	bit 0	TTL/ST(1)	Input/output pin or external interrupt input. Internal software programmable weak pull-up.
RB1	bit 1	TTL	Input/output pin. Internal software programmable weak pull-up.
RB2	bit 2	TTL	Input/output pin. Internal software programmable weak pull-up.
RB3/PGM(3)	bit 3	TTL	Input/output pin or programming pin in LVP mode. Internal software programmable weak pull-up.
RB4	bit 4	TTL	Input/output pin (with interrupt-on-change). Internal software programmable weak pull-up.
RB5	bit 5	TTL	Input/output pin (with interrupt-on-change). Internal software programmable weak pull-up.
RB6/PGC	bit 6	TTL/ST(2)	Input/output pin (with interrupt-on-change) or in-circuit debugger pin. Internal software programmable weak pull-up. Serial programming clock.
RB7/PGD	bit 7	TTL/ST(2)	Input/output pin (with interrupt-on-change) or in-circuit debugger pin. Internal software programmable weak pull-up. Serial programming data.

B portunun 3 pini RB3/PGM, RB6/PGC ve RB7/PGD düşük voltaj programlama ve devre üstünde hata ayıklayıcı olarak da seçilebilmektedir. Her bir PortB pini entegre içinden pull-up direnci kullanabilmektedir. Sadece bir kontrol biti bütün bu pull-up dirençlerini aktif hale getirebilir. Bu RBPU (OPTION_REG<7>) bitini temizleyerek yapılır. Pull-up'lar power-on reset durumunda kapalı konumdadır. B portu TTL (transistör transistör lojik) yapıdadırlar. Ancak RB0,RB6,RB7 pinleri seçimin şekline göre Schmitt trigger yapıya da sahip olabilirler. Bir pini giriş-çıkış yapmak için Schmitt Trigger TTL'e göre daha uygundur. Bunun sebebi **Schmitt trigger** yapıya sahip devrenin 0' dan 1' e ve 1' den 0' a geçiş süresi **TTL** yapıya göre daha kısa olmasıdır. Bu da daha yüksek frekans ve giriş / çıkış yapmak için daha uygun bir yapı demektir. Bu durum tepki süresini kısaltır. Bu da daha hızlı tepki süresi istediğimiz bir buton ya da anahtarla (switch) kumanda ettiğimiz zamanlamanın güvenlik açısından kritik olduğu bir işlem düşündüğümüzde ST'nin TTL'ye göre avantajını daha iyi anlarız. A portundaki schmitt trigger yapısı analog giriş sinyalleri ile çalışmak üzere özel olarak ayarlanmış bir lojik elamandır.



Şekil 4.6 Grafiklerde **a)** ST ve **b)** TTL arasındaki fark gösterilmiştir

4.1.8 PORTC VE TRISC

Bu bölüm 16F87XA'nın datasheet'inden faydanılarak gerekli yerlerde yorum ve açıklamalar yapılarak oluşturulmuştur.

PortC 8 bit işlem kabiliyetinde çift yönlü bir porttur. PortC'nin veri yönlendirme yazmacı TrisC'dir. Yani TrisC bitini (=1) yaptığımızda PortC'nin ilgili pini giriş olacaktır. Şimdi bu porta bağlı olan sürücüyü yüksek empedans moduna sokarak giriş olarak ayarladığımız C portuna bir high (lojik 1) sinyali yollayıp mikrodenetleyiciye iş yaptırabiliriz. TrisC bitini (=0) yaptığımızda bu yazmacın bağlı olduğu PortC'nin ilgili olduğu pini çıkış yapmış oluruz.

PortC birden fazla çevresel fonksiyon seçmeye izin veren seçici özelliğiyle donanmıştır. PortC pinleri **Schmitt Trigger** giriş ara belleğe (buffer) sahiptir. Bunun sebebi PortC'nin seri haberleşme için kullanılan pinleri TTL yapıda olsaydı kararsız bölge çok geniş bir bölgeyi işgal edeceğinden seri iletişimde yanlışlıklar olacaktı.

Tablo 4.3 C Portunun Schmitt Trigger seviyedeki pinlerinin fonksiyon tablosu

Name	Bit#	Buffer Type	Function
RC0/T1OSO/T1CKI	bit 0	ST	Input/output port pin or Timer1 oscillator output/Timer1 clock input.
RC1/T1OSI/CCP2	bit 1	ST	Input/output port pin or Timer1 oscillator input or Capture2 input/ Compare2 output/PWM2 output.
RC2/CCP1	bit 2	ST	Input/output port pin or Capture1 input/Compare1 output/ PWM1 output.
RC3/SCK/SCL	bit 3	ST	RC3 can also be the synchronous serial clock for both SPI and I^2C modes.
RC4/SDI/SDA	bit 4	ST	RC4 can also be the SPI data in (SPI mode) or data I/O (I^2C mode).
RC5/SDO	bit 5	ST	Input/output port pin or Synchronous Serial Port data output.
RC6/TX/$\overline{CK}$	bit 6	ST	Input/output port pin or USART asynchronous transmit or synchronous clock.
RC7/RX/DT	bit 7	ST	Input/output port pin or USART asynchronous receive or synchronous data.

Bizim tez için kullandığımız RC7 ve RC6 pinleri USART'a (Universal Asenkron Senkron Alıcı verici modülüne) sahiptir.

RC6 :asenkron seri iletişim kullanıldığı taktirde data çıkışı
RC7 :asenkron seri iletişim kullanıldığı taktirde data girişidir.

4.1.9 PORTD VE TRISD

Bu bölüm 16F87XA'nın datasheet'inden faydanılarak gerekli yerlerde yorum ve açıklamalar yapılarak oluşturulmuştur.

NOT: 28 pin sayısı olan cihazlar PortD ve TrisD'ye sahip değildirler.

PortD Schmitt Trigger giriş ara belleğine (buffer) sahip 8 bit bir porttur. Her bir pin ayrı ayrı giriş ya da çıkış olarak ayarlanabilir. PortD 8 bit genişliğinde mikroişlemci (microprocessor) portu olarak (Paralel / slave port) olarak ayarlanabilir. Bunun için PSPMODE(TRISE<4>) kontrol bitini ayarlamak gerekir. Bu modda iken giriş ara bellekleri (buffers) TTL seviyesindedir. (TTL & Schmitt Trigger farkı için **Şekil 4.5.'e bknz**.) Görüldüğü gibi D portunda yapılacak bir değişiklik için E portunda bir kontrol bitini kullandık.

Tablo 4.4 D Portu pinlerinin fonksiyon tablosu

Name	Bit#	Buffer Type	Function
RD0/PSP0	bit 0	ST/TTL(1)	Input/output port pin or Parallel Slave Port bit 0.
RD1/PSP1	bit 1	ST/TTL(1)	Input/output port pin or Parallel Slave Port bit 1.
RD2/PSP2	bit2	ST/TTL(1)	Input/output port pin or Parallel Slave Port bit 2.
RD3/PSP3	bit 3	ST/TTL(1)	Input/output port pin or Parallel Slave Port bit 3.
RD4/PSP4	bit 4	ST/TTL(1)	Input/output port pin or Parallel Slave Port bit 4.
RD5/PSP5	bit 5	ST/TTL(1)	Input/output port pin or Parallel Slave Port bit 5.
RD6/PSP6	bit 6	ST/TTL(1)	Input/output port pin or Parallel Slave Port bit 6.
RD7/PSP7	bit 7	ST/TTL(1)	Input/output port pin or Parallel Slave Port bit 7.

D portunun 8 adet bitinin dijital giriş çıkış ya da paralel-slave port olarak kullanıldığını görüyoruz. Görüldüğü üzere giriş (input) ara bellekleri (buffer) I/O modda iken Schmitt Trigger seviyede Paralel Slave Port modunda iken ise TTL seviyededir.

4.1.10 PORTE VE TRISE

Bu bölüm 16F87XA'nın datasheetinden faydanılarak gerekli yerlerde yorum ve açıklamalar yapılarak oluşturulmuştur

PortE giriş veya çıkış olarak seçmemize imkan veren üç adet (RE0,RE1,RE2) pinden oluşmaktadır. Bu pinler Schmitt Trigger giriş ara beleklerine (buffer) sahiptirler. PortE pinleri mikrobilgisayar portu için I / O (input – output) giriş çıkış kontrol giriş pinleri haline gelmişlerdir. Bunu yapan ise PSPMODE (TRISE<4>) bitini kurmaktır (set). Bu modda iken yapılması gereken TRISE yazmacının 0,1 ve 2. pinlerini kurmak (set) ve pinlerin dijital giriş olarak ayarlandığından emin olmaktır. Bunun yanında ADCON1 kaydedicisinin (register) dijital I / O olarak ayarlanması gerekir. Bu modda iken giriş ara belleği (buffer) TTL yapıdadır.

PortE pinleri A portu gibi analog giriş için seçimlendirilmişlerdir. Analog giriş olarak seçildiğinde bu pinler “0” (lojik 0) okuyacaklardır. TrisE yazmacı PortE pinlerinin yönünü analog giriş olarak seçilseler bile kontrol etmektedirler. Kullanıcı bu pinler analog giriş olarak seçilecekse pinlerin giriş olarak ayarlandığından emin olmalıdır.

Tablo 4.5 E Portu pinlerinin fonksiyon tablosu

Name	Bit#	Buffer Type	Function
RE0/$\overline{RD}$/AN5	bit 0	ST/TTL(1)	I/O port pin or read control input in Parallel Slave Port mode or analog input: $\overline{RD}$ 1 = Idle 0 = Read operation. Contents of PORTD register are output to PORTD I/O pins (if chip selected).
RE1/$\overline{WR}$/AN6	bit 1	ST/TTL(1)	I/O port pin or write control input in Parallel Slave Port mode or analog input: $\overline{WR}$ 1 = Idle 0 = Write operation. Value of PORTD I/O pins is latched into PORTD register (if chip selected).
RE2/$\overline{CS}$/AN7	bit 2	ST/TTL(1)	I/O port pin or chip select control input in Parallel Slave Port mode or analog input: $\overline{CS}$ 1 = Device is not selected 0 = Device is selected

* Idle :boş

4.1.11 PIC Komut Seti

Aşağıdaki tabloda yer alan 35 komuttan iki durum dışında tüm komutlar tek bir çevrim sürer. Osilatör'ümüzün saati bir komut çevrimini oluşturmaktadır. Örneğin 4Mhz frekansında bir osilatör kullanırsak bir komutun işlenmesi için gereken süre 1 µs olur. Eğer şartlı (BTFSC VE BTFSS) komutlar kullanılır ve şart da test ettikten sonra diğer satıra atlaması durumu gerçekleşirse 2 usn sürer.

1 hz = 1 / 1 sn olduğuna göre

Saat frekansı = 4Mhz /4 = 1 Mhz ⟶ 1/ 100000 saniye = 1 µs dir.

Tablo 4.6 35 komuttan oluşan PIC komut seti (instruction) tablosu

Mnemonic, Operands		Description	Cycles	14-Bit Opcode MSb LSb	Status Affected	Notes
BYTE-ORIENTED FILE REGISTER OPERATIONS						
ADDWF	f, d	Add W and f	1	00 0111 dfff ffff	C,DC,Z	1,2
ANDWF	f, d	AND W with f	1	00 0101 dfff ffff	Z	1,2
CLRF	f	Clear f	1	00 0001 lfff ffff	Z	2
CLRW	-	Clear W	1	00 0001 0xxx xxxx	Z	
COMF	f, d	Complement f	1	00 1001 dfff ffff	Z	1,2
DECF	f, d	Decrement f	1	00 0011 dfff ffff	Z	1,2
DECFSZ	f, d	Decrement f, Skip if 0	1(2)	00 1011 dfff ffff		1,2,3
INCF	f, d	Increment f	1	00 1010 dfff ffff	Z	1,2
INCFSZ	f, d	Increment f, Skip if 0	1(2)	00 1111 dfff ffff		1,2,3
IORWF	f, d	Inclusive OR W with f	1	00 0100 dfff ffff	Z	1,2
MOVF	f, d	Move f	1	00 1000 dfff ffff	Z	1,2
MOVWF	f	Move W to f	1	00 0000 lfff ffff		
NOP	-	No Operation	1	00 0000 0xx0 0000		
RLF	f, d	Rotate Left f through Carry	1	00 1101 dfff ffff	C	1,2
RRF	f, d	Rotate Right f through Carry	1	00 1100 dfff ffff	C	1,2
SUBWF	f, d	Subtract W from f	1	00 0010 dfff ffff	C,DC,Z	1,2
SWAPF	f, d	Swap nibbles in f	1	00 1110 dfff ffff		1,2
XORWF	f, d	Exclusive OR W with f	1	00 0110 dfff ffff	Z	1,2
BIT-ORIENTED FILE REGISTER OPERATIONS						
BCF	f, b	Bit Clear f	1	01 00bb bfff ffff		1,2
BSF	f, b	Bit Set f	1	01 01bb bfff ffff		1,2
BTFSC	f, b	Bit Test f, Skip if Clear	1 (2)	01 10bb bfff ffff		3
BTFSS	f, b	Bit Test f, Skip if Set	1 (2)	01 11bb bfff ffff		3
LITERAL AND CONTROL OPERATIONS						
ADDLW	k	Add Literal and W	1	11 111x kkkk kkkk	C,DC,Z	
ANDLW	k	AND Literal with W	1	11 1001 kkkk kkkk	Z	
CALL	k	Call Subroutine	2	10 0kkk kkkk kkkk		
CLRWDT	-	Clear Watchdog Timer	1	00 0000 0110 0100	$\overline{TO}$,$\overline{PD}$	
GOTO	k	Go to Address	2	10 1kkk kkkk kkkk		
IORLW	k	Inclusive OR Literal with W	1	11 1000 kkkk kkkk	Z	
MOVLW	k	Move Literal to W	1	11 00xx kkkk kkkk		
RETFIE	-	Return from Interrupt	2	00 0000 0000 1001		
RETLW	k	Return with Literal in W	2	11 01xx kkkk kkkk		
RETURN	-	Return from Subroutine	2	00 0000 0000 1000		
SLEEP	-	Go into Standby mode	1	00 0000 0110 0011	$\overline{TO}$,$\overline{PD}$	
SUBLW	k	Subtract W from Literal	1	11 110x kkkk kkkk	C,DC,Z	
XORLW	k	Exclusive OR Literal with W	1	11 1010 kkkk kkkk	Z	

PIC16 ailesi mikrodenetleyicilerinin komut setleri 3 temel kategoriye göre çalışırlar. Şimdi bu 3 kategoriyi örneklemelerle açıklayacağız.

Byte-yönlendirmeli operasyonlar

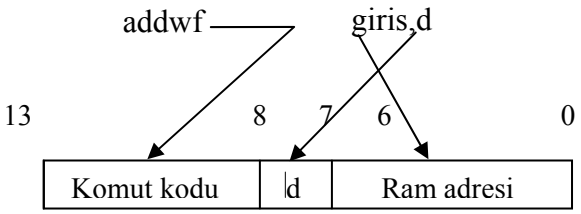
Bit-yönlendirmeli operasyonlar

Sabit işleyen ve kontrol operasyonları

4.1.12 Byte-Yönlendirmeli Operasyonlar

Komutun gösterim şekli: addw f,d

Komutun yaptığı işlem: w register'ını ram üzerindeki f register'ına ekle



d = 0 Sonuç W registerı üzerine kaydedilsin
d = 1 Sonuç RAM üzerine kaydedilsin

Şekil 4.7 Byte yönlendirme ile addw f,d komutu

Burada ' f ' her hangi bir dosya kaydını (register) temsil ederken ' d ' kaydın yapılacağı yani geçici bellek kaydedicisini temsil eder. ' f ' kayıt göstergesi (register designator) komut tarafından hangi kayıt dosyasının kullanılacağını belirler.

Yukarda grafikte de gösterdiğim gibi;

d = 0 ise sonuç W kaydedicisine konulur.
d = 1 ise sonuç komutta belirtilen kayıt dosyasına kaydedilir.

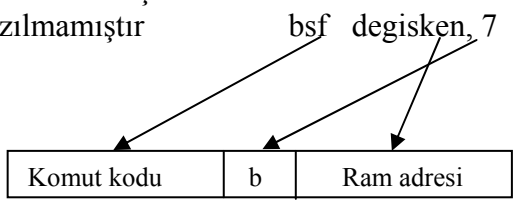
4.1.13 Bit-Yönlendirmeli Operasyonlar

Her bir PIC16 komut seti 14 bit word tipinde bölünmüş operasyon kodlarına sahiptir. Bu makine kodunun içerisinde **komut kodu** yanında 3 bitlik hangi bit üzerinden işlem yapılacağını belirten **bit numarası** ve 7 bitlik **veri adresi** yer alır.

Komutun gösterim şekli: bsf f,b (bit set f)

Komutun yaptığı işlem: f kaydedicisinin (register) b. bitini 1 yap

Not: 'degisken' kaydedicisindeki "g" harfi bilinçli olarak Türkçe karakter olarak yazılmamıştır



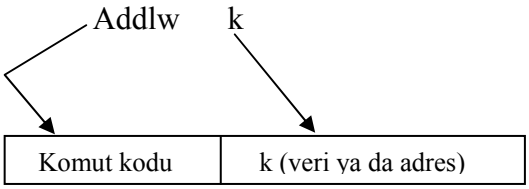
Şekil 4.8 Bit yönlendirmesi ile bsf degisken,7 komutu

4.1.14 Sabit İşleyen ve Kontrol Operasyonları

Bu operasyonlarda ‘k’ 8 ya da 11 bitlik veri ya da adres bilgisinden oluşurlar.

Komutun gösterim şekli: addlw k

Komutun yaptığı işlem: k değerini w kaydedicisine (register) ekle



Şekil 4.9 Sabit işleyen kod ile Addlw k komutu

4.1.15 16F877 USART modülü incelemesi

Bu bölüm 16F87XA’nın datasheetinden faydanılarak gerekli yerlerde yorum ve açıklamalar yapılarak oluşturulmuştur

16F87X ailesinde bulunan Universal Senkron Asenkron Alıcı Verici modülü aynı zamanda seri iletişim arayüzü (serial communication interface) olarak bilinir. Usart modülü, RS232 iletişimi destekleyen bilgisayar ve terminaller gibi başka cihazlarla iletişim kurmak için kullanışlıdır. Pic’ler bu modül sayesinde çift yönlü asenkron bir cihaz olarak ayarlanabilmektedirler.

16F877’de Usart işlemleri için RCSTA, TXREG, RCREG, TXSTA ve SPBRG kayıt tutucuları(registers) ilişkilendirilmiştir. Bunlardan ilk üçü bank 0’ da ve diğer ikisi bank 1’ de yer almaktadır.

TXSTA , veri gönderme (transmit) durumu ve kontrol kaydedicisidir.(register)

RCSTA, veri alma (recieve) durumu ve kontrol kaydedicisidir.(register)

TXREG, okuma / yazma verici ara belleğine sahip (buffer) kaydedicidir.

RCREG, çift ara bellekli (buffer) okuma / yazma kaydedicisidir.

SPBRG kaydedicisi (register) 8 bit’lik bir zamanlayıcının (timer) periyodunu kontrol eden kaydedicidir. Bir başka dilde baud rate generator register’ıdır.

4.1.15.1 Baud Rate

Baud Rate, bit / sn olarak ifade edilen seri iletişim hızına denir.Bir seri iletişim kurulduğunda bu iletişimdeki bir bitin Gönderilme süresi 52,083 μs ise baud rate = 1 / 52,083 μs = 19200 baud olur.

Biz robotun kontrolü sırasında visual basic ile arayüzden kontrol sırasında 19200 baud rate kullanırken klavye de 9600 baud 'da yeterli gelmiştir. Devre üzerinde prototip aşamasında her bir servo motoru bir yöne hareket ettirmek için 2'şer adet buton koyulmuş ve bu sırada klavye kontrolünde 9600 baud'dan aşağısı yeterli gelmemiş daha düşük hızlarda seri iletişimde kopmalar gözlemlenmiştir.

Pic'in seri porttan klavye ile yapılan çalışmalarında pic'e prosedürler içinde tanımlanan bir takım hareketler klavyenin çeşitli tuşlarına atanmış ve bu tuşlara basıldığı anda pic'in bu prosedürlere gitmesi ve motorlara istenen konuma gitmesi sağlanmıştır. Bu işlem esnasında 9600 baud'un altında seri iletişimde yine kopmalar gözlemlenmiş 9600 ve 19200 baud'da kusursuz sonuçlar alınmıştır.

Not: Bu bölümde seri iletişim için bu kadar açıklama yapılacak visual basic programı kullanarak pc-pic arasında gerçekleştireceğimiz seri iletişim için kullanılan komutlar ve seri iletişim hakkında detaylı bilgi ileriki bölümlerde yapılacaktır.

BÖLÜM BEŞ

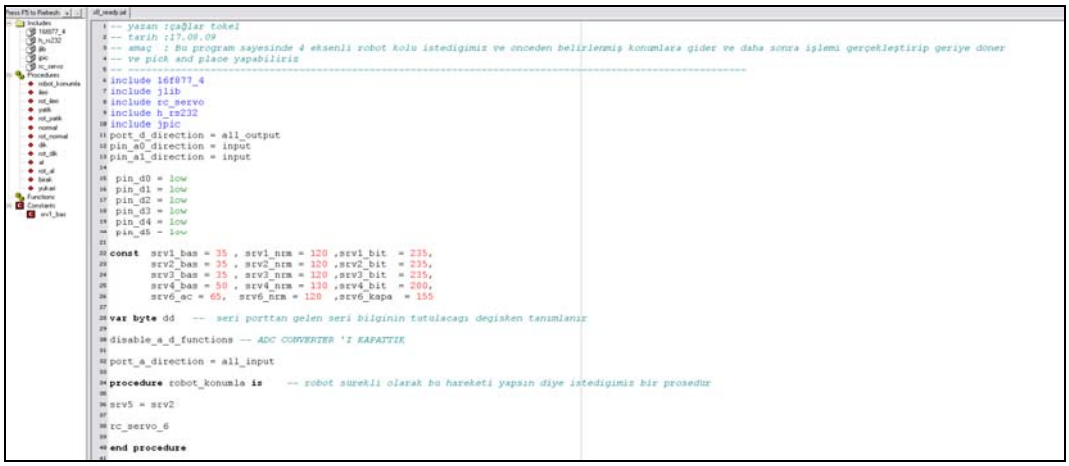
YAZILIM

5.1 Robotu Kontrol Etmek İçin Gerekli Yazılımlar

Bu bölümde robotun öğrettiğimiz bir iş yapması için ve ayrıca visual basic programıyla hazırladığımız arayüz yardımıyla kontrol edilmesi için yazılan program ve kodlar gösterilecektir. Bölüm **1.1'de** servo motorların nasıl kontrol edildiği anlatıldığından bu aşamada yinelenmeyecektir.

5.1.1 Klavyeden Kontrol için PIC'e Yüklenmesi Gereken Kod

Şekilde5.1 de görülebileceği gibi Jal dilinde program yazarken ana programda kullanmak üzere prosedürler, fonksiyonlar tanımlayabildiğimiz, içinde assembly dilini kullanmaya olanak vermesi sebebiyle tercih edilmiştir. Pic Programlama dilleri birbirine benzemekte olup isim atama farklılıkları ya da kütüphane dosyalarının farklılıklarıyla birbirinden ayrılmaktadır. Kullanılabilecek derleyiciler ccs c, winasm, mplab, picbasicpro, picbasic, cc5 C, picC şeklinde sayılabilir. Aynı zamanda ne yapıldığını daha iyi anlamak için en azından 35 komut setinden oluştuğunu söylediğimiz ve küçük bir giriş yaptığımız assembly dilini öğrenmekte fayda vardır. (bknz syf.79,80) Not: Kod yazarken alışkanlık haline getirmek için açıklama yazılarının da Türkçe karakter içermeden yazılmasında fayda var.



```
-- yazan :çağlar tokel
-- tarih :17.08.09
-- amaç  : Bu program sayesinde 4 eksenli robot kolu istediğimiz ve önceden belirlenmiş konumlara gider ve daha sonra işlemi gerçekleştirip geriye döner
-- ve pick and place yapabiliris
include 16f877_4
include jlib
include rc_servo
include h_rs232
include jpic
port_d_direction = all_output
pin_a0_direction = input
pin_a1_direction = input

 pin_d0 = low
 pin_d1 = low
 pin_d2 = low
 pin_d3 = low
 pin_d4 = low
 pin_d5 = low

const  srv1_bas = 35 , srv1_nrm = 120 ,srv1_bit  = 235,
       srv2_bas = 35 , srv2_nrm = 120 ,srv2_bit  = 235,
       srv3_bas = 35 , srv3_nrm = 120 ,srv3_bit  = 235,
       srv4_bas = 50 , srv4_nrm = 130 ,srv4_bit  = 200,
       srv6_ac = 65,  srv6_nrm = 120  ,srv6_kapa  = 155

var byte dd   --  seri porttan gelen seri bilginin tutulacagı degisken tanımlanır

disable_a_d_functions -- ADC CONVERTER 'I KAPATTIK

port_a_direction = all_input

procedure robot_konumla is     -- robot sürekli olarak bu hareketi yapsın diye istediğimiz bir prosedür

srv5 = srv2

rc_servo_6

end procedure
```

Şekil 5.1 Jalwin programının ekran görüntüsü

5.1.2 Robotun PIC / PC Haberleşmesi ile Seri Porttan Klavye ile Kontrolü

```
-- yazan: çağlar tokel
-- tarih: 23.01.09
-- revizyon: 17.08.09
-- amaç : Bu program sayesinde 4 eksenli robot kolu istedigimiz ve önceden
-- belirlenmiş konumlara gider bu sayede istediğimiz ve önceden tanımladığımız bir
-- tut ve bırak  (pick and place) işlemi yapabiliriz
-- -----------------------------------------------------------------------------------------------
include 16f877_4
include jlib
include rc_servo
include h_rs232
include jpic
port_d_direction = all_output
pin_a0_direction = input
pin_a1_direction = input
 pin_d0 = low
 pin_d1 = low
 pin_d2 = low
 pin_d3 = low
 pin_d4 = low
 pin_d5 = low
const  srv1_bas = 35 , srv1_nrm = 120 ,srv1_bit  = 235,
       srv2_bas = 35 , srv2_nrm = 120 ,srv2_bit  = 235,
       srv3_bas = 35 , srv3_nrm = 120 ,srv3_bit  = 235,
       srv4_bas = 50 , srv4_nrm = 130 ,srv4_bit  = 200,
       srv6_ac = 65,  srv6_nrm = 120  ,srv6_kapa  = 155

var byte dd   --  seri porttan gelen seri bilginin tutulacagı degisken tanımlanır

disable_a_d_functions -- ADC CONVERTER 'I KAPATTIK
```

```
port_a_direction = all_input

procedure robot_konumla is    -- robot sürekli olarak bu hareketi yapsın diye
-- istedigimiz bir prosedür
srv5 = srv2
rc_servo_6
end procedure

procedure ileri is

srv1 = srv1_nrm robot_konumla
srv2 = 140 robot_konumla   -- senkron motor 1
srv3 = srv3_nrm robot_konumla
srv4 = srv4_nrm robot_konumla
-- srv5 = srv5_nrm  robot_konumla ( 10 )  -- senkron motor 2
srv6 = srv6_nrm robot_konumla

end procedure

procedure rot_ileri is
srv1 = srv1_bit robot_konumla
srv2 = 140 robot_konumla   -- senkron motor 1
srv3 = srv3_nrm robot_konumla
srv4 = srv4_nrm robot_konumla
-- srv5 = srv5_nrm  robot_konumla ( 10 )  -- senkron motor 2
srv6 = srv6_nrm robot_konumla
end procedure

procedure yatik is

srv1 = srv1_bit robot_konumla   -- Govdeyi dondurme işlemi (taban servo motoru)
srv2 = 145 robot_konumla   -- eş zamanli (senkron motor 1)
```

```
srv3 = 85 robot_konumla
srv4 = 90 robot_konumla
-- srv5 = srv5_nrm  robot_konumla ( 10 )  -- eş zamanli(senkron motor 2)
srv6 = srv6_nrm robot_konumla
end procedure

procedure rot_yatik is

srv1 = srv1_nrm robot_konumla   -- base döndürme işlemi servo motoru
srv2 = 145 robot_konumla   -- senkron motor 1
srv3 = 85 robot_konumla
srv4 = 90 robot_konumla

-- srv5 = srv5_nrm  robot_konumla ( 10 )  -- senkron motor 2
srv6 = srv6_nrm robot_konumla

end procedure
procedure normal is

srv1 = 120 robot_konumla   -- base dondurme işlemi servo motoru
srv2 = 85 robot_konumla   -- 1.senkron servo motoru
srv3 = srv3_bas robot_konumla   -- kritik eksen servo motoru
srv4 = 70 robot_konumla   -- gripper yukarı asagı servo motoru
-- srv5 = srv5_nrm  robot_konumla ( 10 )  -- 2.senkron servo motoru
srv6 = srv6_nrm robot_konumla   -- gripper servo motoru

end procedure

procedure rot_normal is

srv1 = 235 robot_konumla   -- base döndürme işlemi servo motoru
srv2 = 85 robot_konumla   -- 1.senkron servo motoru
```

```
srv3 = srv3_bas robot_konumla   -- kritik eksen servo motoru
srv4 = 70 robot_konumla   -- gripper yukarı asagı servo motoru
-- srv5 = srv5_nrm  robot_konumla ( 10 )  -- 2.senkron servo motoru
srv6 = srv6_nrm robot_konumla   -- gripper servo motoru

end procedure

procedure dik is

srv1 = srv1_nrm robot_konumla -- base döndürme işlemi servo motoru
srv2 = 65 robot_konumla      -- senkron motor 1
srv3 = 145  robot_konumla      -- kritik eksen servo motoru
srv4 = srv4_nrm robot_konumla -- gripper yukarı asagı servo motoru
-- srv5 = srv5_nrm  robot_konumla ( 10 )  -- senkron motor 2
srv6 = srv6_nrm robot_konumla -- gripper servo motoru

end procedure

procedure rot_dik is

srv1 = 180 robot_konumla -- base döndürme işlemi servo motoru
srv2 = 65 robot_konumla      -- senkron motor 1
srv3 = 145  robot_konumla      -- kritik eksen servo motoru
srv4 = srv4_nrm robot_konumla -- gripper yukarı asagı servo motoru
-- srv5 = srv5_nrm  robot_konumla ( 10 )  -- senkron motor 2
srv6 = srv6_nrm robot_konumla -- gripper servo motoru

end procedure



procedure al is
```

```
srv1 = 60 robot_konumla   -- base döndürme işlemi servo motoru
srv2 = 130 robot_konumla   -- 1.senkron servo motoru
srv3 = 160 robot_konumla   -- kritik eksen servo motoru
srv4 = 150 robot_konumla   -- gripper yukarı asagı servo motoru
-- srv5 = srv5_nrm  robot_konumla ( 10 )  -- 2.senkron servo motoru
srv6 = 100 robot_konumla              -- gripper'ı aç
srv6 = srv6_kapa robot_konumla   -- gripper servo motoru (gripper'ı kapa)

end procedure

procedure rot_al is

srv1 = 175 robot_konumla   -- base döndürme işlemi servo motoru
srv2 = 150 robot_konumla   -- 1.senkron servo motoru
srv3 = 160 robot_konumla   -- kritik eksen servo motoru
srv4 = 150 robot_konumla   -- gripper yukarı asagı servo motoru
-- srv5 = srv5_nrm  robot_konumla ( 10 )  -- 2.senkron servo motoru
srv6 = 100 robot_konumla
delay_1s
srv6 = srv6_kapa robot_konumla   -- gripper servo motoru
-- srv3 = 110 robot_konumla
end procedure

procedure birak is
srv1 = 205 robot_konumla   -- base döndürme işlemi servo motoru
srv2 = 170 robot_konumla   -- 1.senkron servo motorusrv3 = 170 robot_konumla   --
-- kritik eksen servo motoru
srv3 = 170 robot_konumla   -- kritik eksen servo motoru
srv4 = 80 robot_konumla   -- gripper yukarı asagı servo motoru
-- srv5 = srv5_nrm  robot_konumla ( 10 )  -- 2.senkron servo motoru
srv6 = 100 robot_konumla   -- gripper servo motoru
```

```
srv1 = 60
end procedure

procedure yukari is

srv1 = srv1_nrm robot_konumla
srv2 = 90 robot_konumla   -- senkron motor 1
srv3 = srv3_nrm robot_konumla
srv4 = srv4_nrm robot_konumla
-- srv5 = srv5_nrm  robot_konumla ( 10 )  -- senkron motor 2
srv6 = srv6_nrm robot_konumla

end procedure

srv1 = srv1_nrm
srv2 = srv2_nrm
srv3 = srv3_nrm
srv4 = srv4_nrm
srv6 = srv6_nrm

srv5 = srv2

serial_setup ( 192 )
```

```
forever loop
if h_asynch_poll ( dd ) then

 if dd == "n" then normal end if
 if dd == "m" then rot_normal end if
 if dd == "y" then yatik end if
 if dd == "u" then rot_yatik end if
 if dd == "d" then dik end if
 if dd == "f" then rot_dik end if
 if dd == "a" then al end if
 if dd == "z" then rot_al end if
 if dd == "b" then birak end if
 if dd == "o" then al birak end if
 if dd == "i" then ileri end if
 if dd == "t" then yukari end if

 h_asynch = dd

end if

if  pin_a0 == high then al birak end if

srv5 = srv2

rc_servo_6  ( 1 )
```

```
if pin_a1 == high then
h_asynch = "4"
h_asynch = " "
h_asynch = "E"
h_asynch = "K"
h_asynch = "S"
h_asynch = "E"
h_asynch = "N"
h_asynch = " "
h_asynch = "R"
h_asynch = "O"
h_asynch = "B"
h_asynch = "O"
h_asynch = "T"
h_asynch = " "
h_asynch = "K"
h_asynch = "O"
h_asynch = "L"
h_asynch = 13
h_asynch = 10
delay_100ms ( 7 )

end if
end loop
```

5.1.3 Jal İçinde Assembly Dilinin Kullanılması

```
assembler
local T1
t1:
  nop  -- 1 çevrim süresi sürer bu da 4mhz için 1usn'dir.
goto t1
end assembler
```

5.1.4 Yaratılan Kütüphane Dosyası 'rc_servo' Prosedürünün Kodları

Bu kütüphane haricindeki kütüphane dosyaları orijinal olan jalwin programının kütüphane dosyaları kullanılarak programa eklenmiştir.

```
-- dosya : rc_servo.jal
-- yazar: Çağlar Tokel
-- tarih: 12-06-2009
-- 6 tane RC servo motoru kontrol eder
-- Jal'ın yaratıcısı Wouter van Ooijen'in orijinal kütüphane dosyalarından örnek alınarak oluşturulmuştur.
-- -----------------------------------------------------------------------------------------
include rc_base_tez
include jlib
procedure delay_1usn ( byte in x ) is
-- bu 1 usn jalwin kütüphane dosyasi içinde bulunan 1us lik gecikmeden farklı olup
-- daha keskin bir gecikme saglamak için yapilmistir
-- servonun düzgün çalışması için tam düzgün çalışan bir 1usnlik
-- gecikme altprogramına ihtiyaç vardır küçük gecikmeler en iyi assembler
-- sayesinde elde edilebilir
-- Bu alt program 4 mhz'lik pic için hazırlanmıştır
-- diğer frekanslar için yeniden düzenlenmelidir
```

```
 assembler
  local T1
 t1:
   nop  -- 1 çevrim süresi sürer 4mhz için 1usn sürer
   goto t1
 end assembler
end procedure

procedure delay_8usn ( byte in x ) is

  assembler
  local T1
 t1:
   nop  -- 1 çevrim süresi sürer 4mhz için 1usn sürer  pic bu süre boyunca hiç bir
işlem yapmaz
            nop  -- 1 çevrim süresi sürer 4mhz için 1usn sürer
            nop  -- 1 çevrim süresi sürer 4mhz için 1usn sürer
            nop  -- 1 çevrim süresi sürer 4mhz için 1usn sürer
            nop  -- 1 çevrim süresi sürer 4mhz için 1usn sürer
   decfsz x,f
   goto t1
 end assembler

end procedure
-- -----------------------------------------------------------------------------

procedure delay_10usn ( byte in x ) is
-- bu 10 usn jalwin kütüphane dosyasi içinde buluna 10us lik gecikmeden farklı olup
daha keskin bir gecikme saglamak için yapilmistir.
-- servonun düzgün çalışması için tam düzgün çalışan bir 10usnlik
-- gecikme alt programına ihtiyaç vardır küçük gecikmeler ise en iyi assembler
-- sayesinde elde edilebilir
```

```
 assembler
 local T1
t1:
  nop  -- 1 çevrim süresi sürer 4mhz için 1usn sürer
          nop  -- 1 çevrim süresi sürer 4mhz için 1usn sürer
          nop  -- 1 çevrim süresi sürer 4mhz için 1usn sürer
          nop  -- 1 çevrim süresi sürer 4mhz için 1usn sürer
          nop  -- 1 çevrim süresi sürer 4mhz için 1usn sürer
          nop  -- 1 çevrim süresi sürer 4mhz için 1usn sürer
          nop  -- 1 çevrim süresi sürer 4mhz için 1usn sürer
          decfsz x,f  -- 1 çevrim süresi ya da 2 çevrim süresi sürer 4mhz için 1usn veya
2usn sürer
          goto t1     -- 1 çevrim süresi sürer 4mhz için 1usn sürer
          -- toplam 10usn gecikme yapıldı
 end assembler

end procedure
-- ---------------------------------------------------------------------------
procedure rc_servo_1 ( byte in x = 1 ) is -- 1 tane Dirsekteki RC servoyu kontrol
-- etmek için pin_d2
for x loop                          -- klavye kontrolünde kullanılan prosedür
 for 10 loop
    pin_d2 = high
    delay_10usn ( srv3 )
    pin_d2 = low
                  delay_1ms ( 15 )
end loop
end loop
end procedure
-- ---------------------------------------------------------------------------
procedure rc_servo_gya ( byte in x = 1 ) is -- 1 tane gripper'i yukarı asagi hareket
veren motor
```

```
for x loop
 for 10 loop
    pin_d3 = high
    delay_10usn ( srv4 )
    pin_d3 = low
                    delay_1ms ( 16 )
end loop
end loop
end procedure
-- -----------------------------------------------------------------------------
procedure rc_servo_one ( byte in x = 1 ) is -- 1 tane ve base servo motoru sürmek
için
for x loop
 for 10 loop
    pin_d0 = high
    delay_10usn ( srv1 )
    pin_d0 = low
                    delay_1ms ( 16 )
end loop
end loop
end procedure
-- -----------------------------------------------------------------------------
procedure rc_servo_grip ( byte in x = 1 ) is -- 1 tane ve gripper aç kapa motoru
sürmek için
for x loop
  for 10 loop
    pin_d5 = high
    delay_10usn ( srv6 )
    pin_d5 = low
                    delay_1ms ( 16 )
end loop
end loop
```

```
end procedure
-- -----------------------------------------------------------------------------
procedure rc_servo_kritik ( byte in x = 1 ) is -- 1 tane kritik eksendeki servoyu –
-- sürmek için
for x loop                              -- visual basic'le kontrolde kullanılır
 for 10 loop
    pin_d2 = high
    delay_10usn ( srv3 )
    pin_d2 = low
                  delay_1ms ( 18 )
end loop
end loop
end procedure
-- -----------------------------------------------------------------------------
procedure rc_servo_2 ( byte in x = 1 ) is -- 2 tane senkron calısan RC servo kontrol
etmek için
for x loop
 for 10 loop
   pin_d1 = high
   delay_10usn ( srv2 )
   pin_d1 = low

   pin_d4 = high
   delay_10usn ( srv5 )
   pin_d4 = low

                  delay_1ms ( 18 )
end loop
end loop
end procedure
-- -----------------------------------------------------------------------------
```

```
procedure rc_servo_two ( byte in x = 1 ) is -- base ve gripper yukarı asagi servo
motoru için
for x loop
 for 10 loop
    pin_d0 = high
    delay_10usn ( srv1 )
    pin_d0 = low

    pin_d3 = high
    delay_10usn ( srv4 )
    pin_d3 = low

                    delay_1ms ( 17 )
end loop
end loop
end procedure
-- ---------------------------------------------------------------------------
procedure rc_servo_uc ( byte in x = 1 ) is -- base servosu ve 2 adet senkron servo
motoru kontrol için
for x loop
 for 10 loop
    pin_d0 = high
    delay_10usn ( srv1 )
    pin_d0 = low

    pin_d1 = high
    delay_10usn ( srv2 )
    pin_d1 = low


    pin_d4 = high
    delay_10usn ( srv5 )
```

```
   pin_d4 = low

                  delay_1ms ( 17 )
end loop
end loop
end procedure
-- -----------------------------------------------------------------------
procedure rc_servo_3 ( byte in x = 1 ) is -- 3 tane RC servo kontrol etmek için
for x loop
 for 10 loop
   pin_d0 = high
   delay_10usn ( srv1 )
   pin_d0 = low

   pin_d1 = high
   delay_10usn ( srv2 )
   pin_d1 = low

   pin_d2 = high
   delay_10usn ( srv3 )
   pin_d2 = low

            delay_1ms ( 14 )
end loop
end loop
end procedure
-- -----------------------------------------------------------------------



procedure rc_servo_4 ( byte in x = 1 ) is -- 4 tane RC servo kontrol etmek için
for x loop
```

```
 for 10 loop
    pin_d0 = high
    delay_10usn ( srv1 )
    pin_d0 = low

    pin_d1 = high
    delay_10usn ( srv2 )
    pin_d1 = low

    pin_d2 = high
    delay_10usn ( srv3 )
    pin_d2 = low

    pin_d3 = high
    delay_10usn ( srv4 )
    pin_d3 = low

                    delay_1ms ( 13 )
end loop
end loop
end procedure
-- -----------------------------------------------------------------------------
procedure rc_servo_5 ( byte in x = 1 ) is -- 5 tane RC servo kontrol etmek için
for x loop
   for 10 loop
    pin_d0 = high
    delay_10usn ( srv1 )
    pin_d0 = low
    -- pin_d1 = high
    -- delay_10usn ( srv2 )
    -- pin_d1 = low
```

```
      pin_d2 = high
     delay_10usn ( srv3 )
     pin_d2 = low


  -- pin_d2 = high
  -- delay_10usn ( srv3 )
  -- pin_d2 = low

     pin_d3 = high
     delay_10usn ( srv4 )
     pin_d3 = low

    pin_d1 = high
    delay_10usn ( srv2 )
    pin_d1 = low

    pin_d4 = high
    delay_10usn ( srv5 )
    pin_d4 = low
  -- pin_d4 = high
  -- delay_10usn ( srv5 )
  -- pin_d4 = low

             delay_1ms ( 11 )
end loop
end loop
end procedure
-- -----------------------------------------------------------------------------
 procedure rc_servo_6 ( byte in x = 1 ) is -- 6 tane RC servo kontrol etmek için
-- bu  satırı seri porttan s_ready programı haricinde x = 1 olarak kullan
for x loop                        -- ( byte in x = 1 ) olarak kullan s_ready hariç olanlarda
```

```
 for 10 loop
    pin_d0 = high
    delay_10usn ( srv1 )
    pin_d0 = low

    pin_d1 = high
    delay_10usn ( srv2 )
    pin_d1 = low

    pin_d4 = high
    delay_10usn ( srv5 )
    pin_d4 = low

    pin_d2 = high
    delay_10usn ( srv3 )
    pin_d2 = low

    pin_d3 = high
    delay_10usn ( srv4 )
    pin_d3 = low

    pin_d5 = high
    delay_10usn ( srv6 )
    pin_d5 = low
                  delay_1ms ( 10 )
  end loop
end loop
end procedure
```

Kod yazılım kısmında - - çift çizgisi ile yapılan açıklamalar okuyucu için açık olması amacıyla yapılmıştır.Bütün derleyiciler bu açıklamalara izin vermektedir.Bunun için jal'da yapılan - - ardından açıklama yapmaktır.Yeşil renge

dönüşen bu kısım hex koduna dönüştürülmez böylelikle sadece programı yazan veya okuyan kişinin görebileceği bir açıklama kısmına kavuşmuş olunur.

Not: Kod içinde bazı komutlardan önce de çift çizgi konmuş ve onlarında hex koduna dönüşmesi engellenmiştir. Bunun sebebi robot üzerinde çalışmalar yaparken bazı servo motorlara hareket vermeden sadece seçili olan motorları kumanda etmek ve adım adım bütüne gitmektir. Bu açıklama çizgileri kaldırılıp kod içine konulduğu taktirde robotun çalışması için gerekli olan bütün motorlar için gerekli sinyalleri verebiliriz.

5.1.5 C Dili ile Jal Dilinin Benzeşiminin Yapılması

Bu bölümde pic programlamayla ilgilenen herkesin Jal ile program yapmak istemeyebileceği ve daha çok kullanıcı tarafından tercih edilen ve üstüne çok sayıda bilgi bulunabilmesi ve birçok kitap çıkmış olması sebebiyle yazılan kodda yapılan değişikler ile nasıl C kodu haline getirilebileceği anlatılmaya çalışılmıştır.

C dili ile Jal dili birbirine çok benzer dillerdir. Bir dili bilen programcı diğer dille de rahatlıkla programlar yazabilir. Yukarda **5.1.4** ‘de Jal ile yazdığımız kodda aşağıdaki düzenlemeler yapıldığında bir C derleyicisinde derledikten sonra bu kod’da çalışabilecektir.
Not: Jal ‘da aynı satıra birden fazla komut yazılabilmektedir.

Port_a_direction == all_output yerine c ‘de TRISA = 0 x 00 yapılırsa A portu çıkış jal’daki komut gibi A portunun tüm pinlerini çıkış yapılır.
Port_a_direction == all_input yerine c ‘de TRISA = 0 x 07 yapılırsa A portu giriş jal’daki komut gibi A portunun tüm pinlerini giriş yapılır.

- Procedure …… is
- Komut veya komutlar
- end procedure

Yerine :

- Void …( )
- {
- Komut veya komutlar
- }

Yazılırsa kod aynı işe yarar.

- forever loop
- Komut veya komutlar
- end loop

Yerine :

- Void main ( )
- {
- Komut veya komutlar
- }

Yazılırsa kod aynı işe yarar.

Serial_setup (192) ,	yerine →	Usart_Init(19200) yazılmalıdır.
port_d = 0 ,	yerine →	PORTD = 0 x 00 yazılmalıdır.
Disable_a_d_functions ,	yerine →	ADCON1 = 7 yazılmalıdır.
h_asynch_poll (dd) ,	yerine →	(Usart_data_ready ( )) dd = Usart_Read( )yazılmalıdır.

Not: Jal kütüphaneleri incelenerek her bir fonksiyon ve prosedürün nasıl oluşturulduğunu anladıktan sonra kendi prosedür ve fonksiyonlarınızı rahatlıkla oluşturabilirsiniz. Programda üstteki değişiklikler yapılması ve jal'da vermiş olduğum kütüphane dosyasının C için derlenmesi halinde bir C derleyicisinde derlenen kodda aynı işi görecektir.

5.1.6 Visual Basic ile Yarattığımız Arayüzden Robotun Kontrolünün Yapılması için PIC'e Yüklenmesi Gereken Kod

```
include 16f877_4

include jlib
include h_rs232
include rc_servo

  var byte dd,c2,c3,c4,c5,c6,b0,b1,b2,b3,b4,b5,x = 0

  pin_b4_direction = output
  port_d_direction = all_output
  disable_a_d_functions

  pin_d0 = low
  pin_d1 = low
  pin_d2 = low
  pin_d3 = low
  pin_d4 = low
  pin_d5 = low
  pin_d6 = low
  pin_b4 = low

  serial_setup ( 96 )

forever loop
 if h_asynch_poll ( dd )   then
     pin_b4  = high
     -- delay_100us(5)
 if        x == 0  & dd == "a"        then    b0 = dd   h_asynch = "c"      x = x + 1
delay_1ms(300)
```

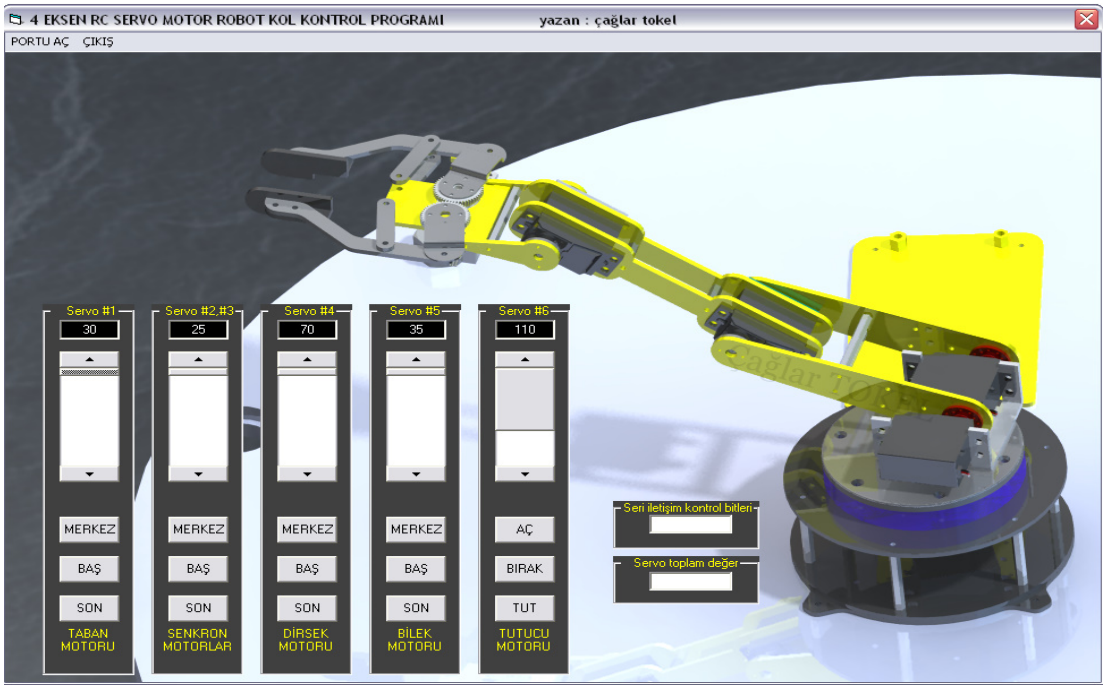
```
elsif     x == 1                    then    c1 = dd                        x = x + 1
delay_1ms(300)

 elsif    x == 2  & dd == "b"       then    b1 = dd    h_asynch = "a"      x = x + 1
 delay_1ms(300)
 elsif     x == 3                   then    c2 = dd                        x = x + 1
 delay_1ms(300)

 elsif     x == 4  & dd != "c"      then        b3 = dd    h_asynch = "g"  x = x + 1
 delay_1ms(300)
 elsif     x == 5                   then        c3 = dd                     x = x + 1
 delay_1ms(300)

 elsif     x == 6   & dd != "d"     then        b4 = dd     h_asynch = "l"  x = x + 1
 delay_1ms(300)
 elsif     x == 7                   then        c4 = dd                     x = x + 1
 delay_1ms(300)

elsif     x == 8   & dd != "e"      then        b5 = dd     h_asynch = "l"  x = x + 1
delay_1ms(300)
elsif     x == 9                    then        c5 = dd                     x = x + 1
delay_1ms(300)
 else x = 0
end if
 if x == 10 then
    srv1 = c1  -- base servo motoru
    srv2 = c2  --  senkron servo motorlar eşit olmasi gerektiginden esitlenmistir
    srv5 = c2  --  senkron servo motorlar eşit olmasi gerektiginden esitlenmistir
    srv3 = c3  -- Dirsekteki servo motor
    srv4 = c4  -- gripper yukarı asagi servo motoru
    srv6 = c5  -- gripper ac kapa servo motoru
     pin_b4 = low  -- b4 pinine bagli led söndürülür.test amaçlıdır.eger led sönerse
```

```
-- x ==10 olmuş ve servo degiskenlerine degerlerinin atanmis oldugu sonucunu –

-- çıkarırız

x = 0

 end if

 --  delay_100us(10)

rc_servo_two ( 1 )

  -- pin_b4 = high

  -- delay_100us(10)

  -- pin_b4 = low

  -- delay_100us(10)

end loop
```

NOT: Bu kod içinde dikkat edilmesi gereken yine (- -) komutudur. Bu çift çizgiden sonra yazılacak olan açıklamadan önce mutlaka bir karakter boşluk bırakılmalıdır.Bu kod ile gereken düzenlemeler yapılıp aynı anda bütün motorlar yerine sadece istenen motorlarda arayüzden kontrol edilebilir.

5.1.7 Robot Manipülatör'ün Arayüzden Kontrolü

Bu bölümde Visual Basic 6.0 programıyla bir arayüz oluşturulmuş ve robot manipulatör seri iletişim teknikleri kullanılarak haberleşip kontrol edilmiştir.



Şekil 5.2 Visual basic 6 ile hazırladığımız ara yüzün ekran görüntüsü

```
' YAZAN:  ÇAĞLAR TOKEL
' 16.10.2009
' 4 EKSEN ROBOT MANİPULATÖR ARAYÜZÜ

Private Sub bas_Click(Index As Integer)
  sldservo(0).Value = 30
  sldServo_Change (0)
End Sub

Private Sub bas1_Click(Index As Integer)
 sldservo(1).Value = 30
 sldServo_Change (1)
End Sub
```

```
Private Sub center_Click(Index As Integer)
 sldservo(Index).Value = 127

 sldServo_Change (Index)
End Sub

Private Sub center2_Click(Index As Integer)
 sldservo(Index).Value = 170
 sldservo(2).Value = 70
 sldServo_Change (2)
End Sub

Private Sub bas3_Click(Index As Integer)
 sldservo(3).Value = 35
 sldServo_Change (3)
End Sub

Private Sub bas4_Click(Index As Integer)
 sldservo(4).Value = 80
 sldServo_Change (4)

 sldServo_Change (Index)
End Sub

Private Sub son_Click(Index As Integer)
 sldservo(0).Value = 210
 sldServo_Change (0)
End Sub

Private Sub son1_Click(Index As Integer)
 sldservo(1).Value = 184
 sldServo_Change (1)
```

```
End Sub

Private Sub son2_Click(Index As Integer)
 sldservo(2).Value = 254
 sldServo_Change (2)
End Sub

Private Sub son3_Click(Index As Integer)
 sldservo(3).Value = 240
 sldServo_Change (3)
End Sub

Private Sub son4_Click(Index As Integer)
 sldservo(4).Value = 130
 sldServo_Change (4)
End Sub



Private Sub mnuCOM_Click(Index As Integer)

    On Error Resume Next
    If MSComm1.PortOpen = True Then
        MSComm1.PortOpen = False
    End If
    MSComm1.Settings = "19200,N,8,1"
    MSComm1.CommPort = 1
    MSComm1.PortOpen = True
End Sub

Private Sub mnuCOM1_Click(Index As Integer)
```

```
    On Error Resume Next
 If MSComm1.PortOpen = True Then
        MSComm1.PortOpen = False
    End If
    MSComm1.Settings = "9600,N,8,1"
    MSComm1.CommPort = 1
    MSComm1.PortOpen = True
End Sub
Private Sub mnuCOM2_Click(Index As Integer)
    On Error Resume Next
    If MSComm1.PortOpen = True Then
        MSComm1.PortOpen = False
    End If
    MSComm1.Settings = "2400,N,8,1"
    MSComm1.CommPort = 1
    MSComm1.PortOpen = True
End Sub

Private Sub Form_Load()
Show


       txtsrv(0).Text = 30
       txtsrv(1).Text = 25
       txtsrv(2).Text = 70
       txtsrv(3).Text = 35
       txtsrv(4).Text = 110
       Text6.Text = ""
    If MSComm1.PortOpen = True Then 'port acik mi kontrol et
MSComm1.PortOpen = False 'acik ise kapat
End If
MSComm1.CommPort = 1 '1 numarali com port kullanilacak
```

```
MSComm1.Settings = "9600,N,8,1" '9600 baud parity biti yok 8 data biti 1 stop biti
olarak data yollanacak
MSComm1.PortOpen = True 'portu ac
wait (1000)
servo_cont.Caption = "4 EKSEN RC SERVO MOTOR ROBOT KOL KONTROL
PROGRAMI              Robot kol ve pc arasında seri iletişim kuruldu..."

 MSComm1.InBufferCount = 0  'Formu actigimizda Buffer belleği temizleyelim ki
‘en sonda bufferda istenmeyen bir data var ise robot zarar görmesin

Text6.Text = ""                    ' buffer’da tutulur bu data inputlen komutu ile
‘okunurken karakter sayısı da inputbuffercount ile bir text dosyasına yazilabilir ve
‘kullanılabilir.
End Sub

Private Sub Form_Unload(Cancel As Integer)

MSComm1.PortOpen = False
End Sub
Private Sub mnuexit_Click()

Unload Me
End Sub
Private Sub sldServo_Change(Index As Integer)
MSComm1.SThreshold = 20  ‘Bu satır çıkartılsa da program çalışacaktır.Sigorta
amaçlı konulmuştur. Gönderilecek datanın minimum kaç karakterden olusacagini
belirler
txtsrv(Index).Text = Str(sldservo(Index).Value)
Text6.Text = Val(Str(sldservo(0).Value)) + Val(Str(sldservo(1).Value)) +
Val(Str(sldservo(2).Value)) + Val(Str(sldservo(3).Value)) +
Val(Str(sldservo(4).Value))
```

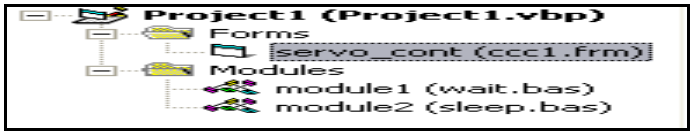
```
MSComm1.Output = "a"
wait (200)
MSComm1.Output = Chr(sldservo(0).Value)
wait (200)
MSComm1.Output = "b"
wait (200)
MSComm1.Output = Chr(sldservo(3).Value)
wait (200)
MSComm1.Output = "c"
MSComm1.Output = Chr(sldservo(2).Value)
wait (500)
MSComm1.Output = "d"
MSComm1.Output = Chr(sldservo(3).Value)
wait (500)
MSComm1.Output = "e"
MSComm1.Output = Chr(sldservo(4).Value)
wait (500)

Text9.Text = MSComm1.Input

End Sub
```

NOT: Program içinde yapılan açıklamalarla birlikte açık hale getirilmeye çalışılmıştır. Bu program ile 4 eksenli artı bir adet tutucuya (gripper) sahip bir robot manipülatörün 6 motorunun kontrolü 2 adedi eş zamanlı kontrol edebilmekle birlikte gerçekleştirilebilmektedir. Aşağıdaki 7 satırda pic'ten pc'ye data yollandığında kullanmak için olan mscomm1 komponent'i için kullanılması gereken komutlardır ve bilgi amaçlı olarak gösterilmek istenmiştir. Bu kodları kullanarak pic'ten pc'ye veri gönderiminin ayarlamalarını yapabilirsiniz.

```
Text6.Text = MSComm1.Input 'Textbox a gelen bilgiyi yaz
MSComm1.InputLen = 1       '1 karakterlik blok halinde buffer dan bilgi oku
MSComm1.InputLen = 0       'Bufferdaki tüm karakterleri oku
MSComm1.InputMode = 1 'text modunda bilgi okur
MSComm1.InputMode = 0 'binary modunda bilgi okur
MSComm1.RThreshold = 15 bu durumda alınacak datanın minimum 15 karakterdir.
MSComm1.InBufferSize = 2048 ara belleği (buffer )varsayılan 512'den 2048'eçıkar.
```

5.1.7.1 Bekle (wait) ve Uyu (sleep) modülleri

Formumuzda bu modülleri kullanabilmek için sleep ve wait modüllerini projenin içine 'add module' yordamını kullanarak eklemeliyiz. Bu modüller visual basic içinde standart olarak gelmediğinden aşağıda bu modülleri yaratıp içine yazılması gereken kodlar gösterilmiştir.



Şekil 5.3 Sleep ve wait modülleri

1.Modülü '**wait'** yaratıp aşağıdaki satırı bu modülün içine yazmalıyız

```
Public Declare Function GetTickCount Lib "kernel32" () As Long
    Public Sub wait(ByVal dblMilliseconds As Double)
        Dim dblStart As Double
        Dim dblEnd As Double
        Dim dblTickCount As Double
        dblTickCount = GetTickCount()
        dblStart = GetTickCount()
        dblEnd = GetTickCount + dblMilliseconds
        Do
        DoEvents
        dblTickCount = GetTickCount()
        Loop Until dblTickCount > dblEnd Or dblTickCount < dblStart
    End Sub
```

2.Modülü '**sleep'** yaratıp aşağıdaki satırı bu modülün içine yazmalıyız.

```
Declare Sub sleep Lib "kernel32" Alias "Sleep" (ByVal dwMilliseconds As Long)
```

BÖLÜM ALTI
BASKI DEVRE TEKNİĞİ

6.1 Baskı Devre Tekniği ile Kartların Yapımı

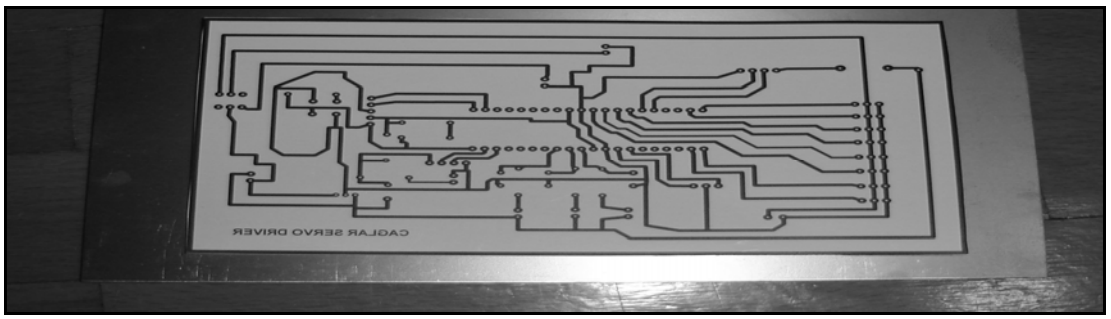
Bu bölümde serigrafi, ipek baskı ve pozitif 20 tekniği üstüne durulmayacak olup, Tez için kullandığımız ütüleme yöntemi ile baskı devre yöntemi üzerine durulacaktır. Bu teknik ile tez kapsamında yapılan kartlarda çok iyi sonuçlar alınmış burada anlattığım şekli ile bu teknik kullanıldığı gerekli malzemeler doğru kullanıldığı takdirde çok iyi sonuçlar alınabilir.

6.1.1 Baskı devrenin hazırlanışı

- Bu yöntem için gerekenler
- Tuz ruhu
- Ovalama tozu
- Hidrojen peroksit (kimyasal madde satan yerlerde bulunabilir.)
- Ütü
- Matkap

Not: hidrojen peroksit (H_2O_2)yerine eczanelerde daha yoğun halde satılan bir türü olan perhidrol de kullanılabilir fakat daha pahallıdır.

Öncelikle bakır plakamızı temiz olduğundan emin olmak için üzerinde parmakla temastan doğan yağ partiküllerinden arındırmalıyız. Ardından ilk iş olarak tasarım programında (proteus kullanıldı) hazırlanan baskı devrenin lazer çıktısı alınmalı. Bu işlemin yapılmasının sebebi lazerin, mürekkebi çok yüksek bir sıcaklıkta transfer (aktarma) kağıdına yapıştırmasıdır. Fotokopi aynı işi yapmadığından bu amaç için kullanılmamalıdır. Tecrübeyle sonuçları sabitlenmiştir.



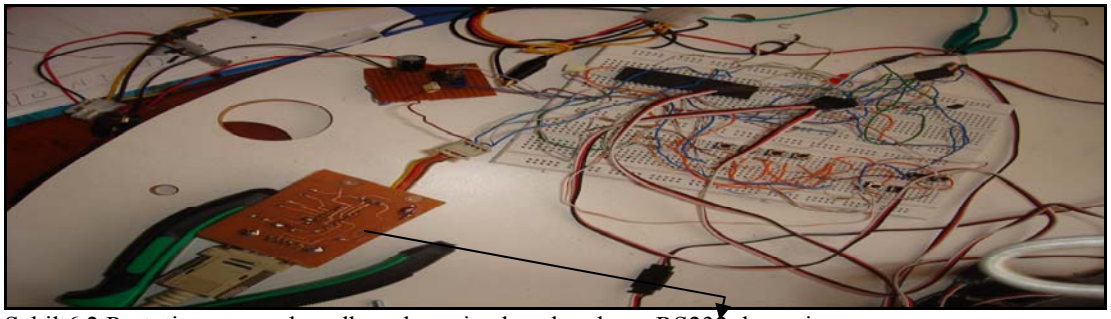
Şekil 6.1 Robotun motor sürücü kartı transfer kağıdından ütüleme işlemine geçmeden önce (transfer kağıdı bu şekliyle değil ters çevirilip ütülenmelidir.

Şekil.6.1 de görülen baskı devre kağıdı, benim bir çok kağıt türünü baskı devre kullanmak için deneyip en iyi sonucu aldığım tez kapakları için kullanılan karton kağıdıdır. Bu kağıda lazerden aldığımız çıktıyı şekilde görülen yüzü altındaki bakır plaketin bakır olan yüzüne öpüştürülmeli ardından ütüleme işlemine geçilmelidir. Ütüleme işleminin yapmamızın amacı transfer kağıdının üzerinde bulunan toner'den Oluşan devre çizimini bakır plakaya aktarmaktır. Bu işlem ütünün doğru ısı ayarını buluncaya kadar tecrübe edilmelidir. Benim kullandığım Philips marka bir buharlı ütüydü. Burada dikkat edilmesi gereken buharlı bir ütüye sahip iseniz buhar özelliğini kullanmamaktadır.

Ütüleme işlemi kişinin tecrübesine ve tekniğine göre kendisinin belirlemesi gereken bir işlemdir. Ben bu işlemi hem bakır plakanın iletim hatları hem de üst yüzdeki komponentlerin sembollerini çıkartmak için kullanıyorum. Doğal olarak iki işlem aynı süre yapılmamalıdır. Üst yüzeye yapılan baskı tercihe kalmış olup görselliği dışında bir önemi yoktur.

Ben, tez sırasında yaptığım kartlar için 15 dakikalık bir ütüleme süresi belirledim. Tabi ki bu süre kartın büyüklüğüne ve iletim hatlarının yoğunluğuna göre değişim gösterecektir. Ütüleme işlemi bu yöntemin en kritik ve en hassas noktasını oluşturmaktadır. Düzgün bir ütüleme için ilk önce ütüyü üstüne 4 noktasından bantladığımız bakır plaketin üstünde bir süre (1dakika kadar) bastırarak beklettikten sonra plaketin her bölümünü aralıksız ve atlamadan belli bir güç uygulayarak gezdirmektir. Ardından ütünün sivri ucu kullanılarak bu kez yine kartın her noktası taranarak bastırılmalıdır. Bu işlemin ardından da ütünün geniş arka tarafı (ütü yukarı bakacak şekilde) kart üzerinde bastırılarak sürülmelidir. Bu işlemler tecrübeyle kişinin kendi ritüel' ini oluşturması gereken bir süreçtir.

Lazerden transfer kağıdına aktardığımız baskı devreyi çözelti içine attığımızda Lazerin toner'inin bu çözeltiye dayanıklı olması ve bu çözeltinin bakırı eritebilme özelliği sayesinde iletim hatlarındaki bakır kısım erimeyecek ve bakır plakanın üstündeki bakır olan bölgeler eriyecektir. Bu sayede geriye sadece istediğimiz baskı devre kalmış olacaktır.



Şekil 6.2 Prototip aşaması breadboard üzerine kurulan devre,RS232 devresi

Seri port kablosu

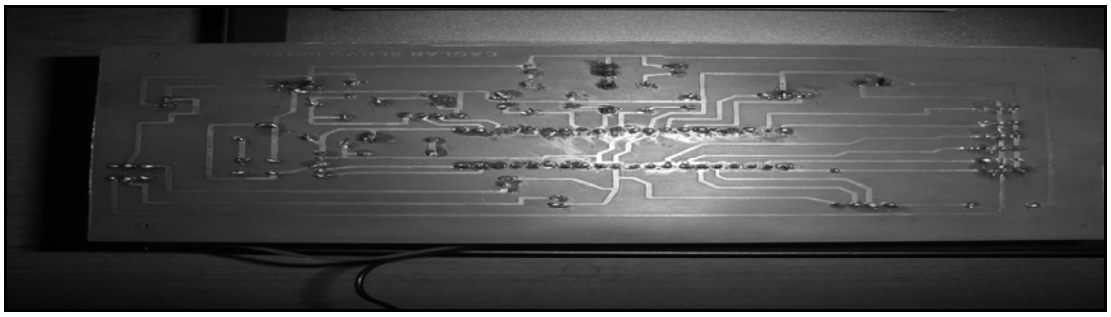
Breadboard prototip devresi (servo motorları sürmek için kurulan devre)

Prototip aşamasının atlanmaması gereken bir süreç olduğu vurgulanmak istenmiştir. Geliştirme için öncelikle breadboard üzerinde devre kurulmalı ve istenen donanım bu prototip sürecinde belirlenene kadar bu yönde çalışılmaya devam edilmelidir. Unutulmamalıdır ki bir baskı devre kartı, üzerinde değişiklik yapılması artık pek kolay olmayan bir karttır.

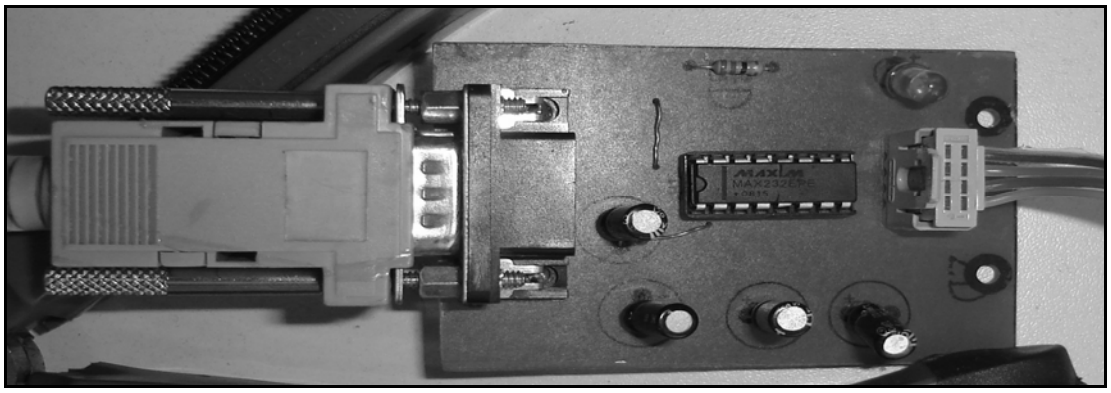
Ütüleme işleminin ardından hidrojen peroksit (H_2O_2) ve tuz ruhundan oluşacak çözelti hazırlanır. 1 ölçek (tuz ruhunun kapağı ile) hidrojen peroksit ve 3 veya 4 ölçek tuz ruhu karışımı ile çözelti hazırlanmalıdır.

Şekil 6.3 Çözelti içine atılan plaketin bakır olan bölümlerinin erimeye başladığı süreç

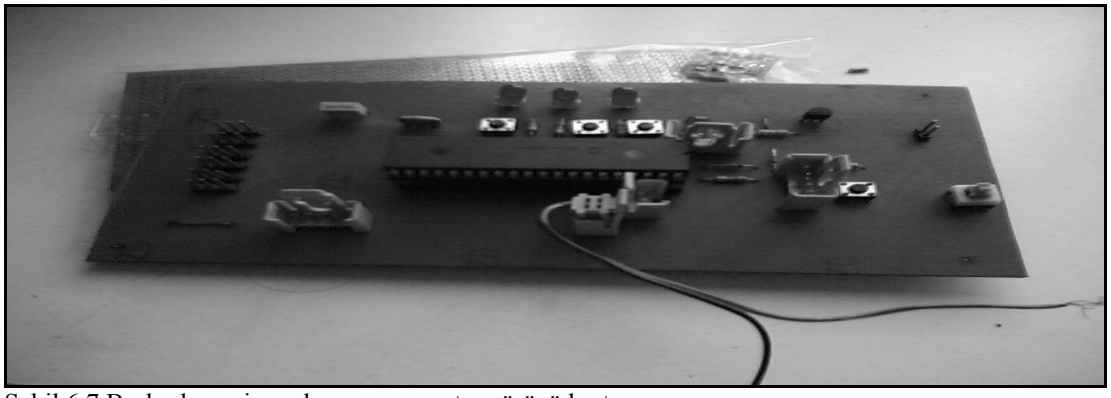
Şekil 6.4 Çözelti içinde bekleyen plakanın bakır kısımları eridikten sonra çözeltinin boşaltılıp bir süre beklendikten sonra iletim hatlarının ortaya çıkması



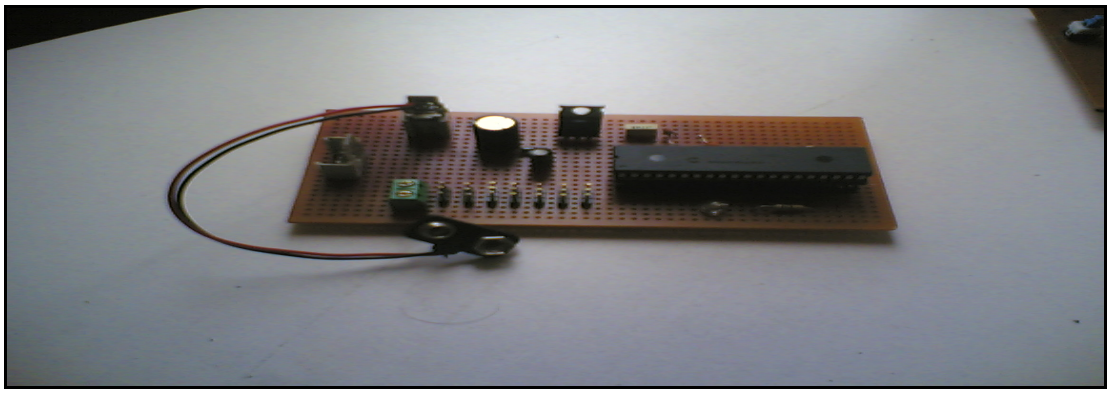
Şekil 6.5 Rc servo motorları sürmek için tasarlanan sürücü kartın lehim yapılmış hali



Şekil 6.6 RS232 devresi kartı seri portla haberleşme kurarken



Şekil 6.7 Baskı devresi yapılmış servo motor sürücü kartı

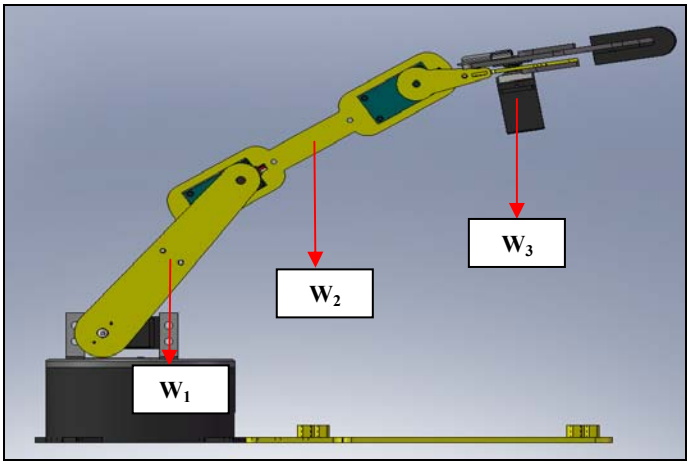


Şekil 6.8 Delikli pertinekse kurulmuş servo motor sürücü

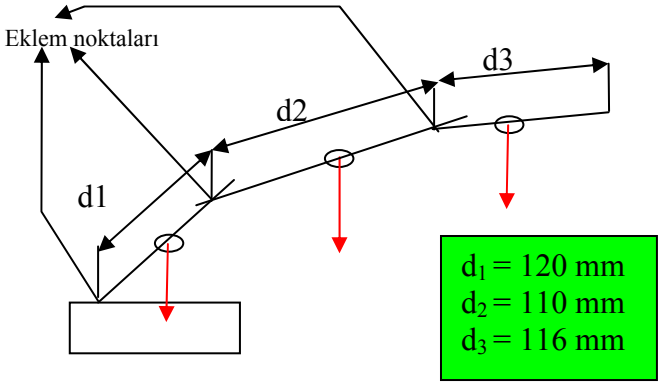
BÖLÜM YEDİ
STATİK TORK ANALİZİ

7.1 Omuz, Dirsek ve Bilek Motorlarının Seçimi

Şimdi her bir uzvun ağırlığını sanki ortalarında tek bir kütle varmış gibi kabul edip daha basit bir sisteme indirgeyelim



Şekil 7.1 Uzuv kütlelerini kendi ağırlık merkezlerine indirgersek



Şekil 7.2 Robotun eklem noktaları ve uzuv boyları

Robotun eklem noktaları ve d1,d2,d3 uzunluklarındaki uzuv ölçüleri

W1 = uzuv1a + uzuv1b + (3) x (aralayıcı)

W2 = uzuv2a + uzuv2b + (2) x (rc servo) + 2 x (yeşil renkli parçalar)

W3 = uzuv3a + uzuv3b + 4 x (parmak) +2 x (iletim org) + 2 x (tutamaç) + 2 x (hassas parmak) + 1 x (rc servo) + 1 x (tut_gövde)

W1 = 28,47 gr + 28,57 gr + 3 x 3,89 gr = 69 gr

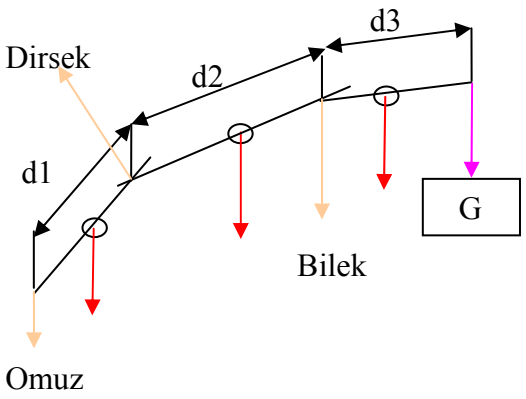
W2 = 20,48 gr + 21,59 gr + 2 x 37 gr = 122 gr

W3 = 2 x4,9gr+4x2,84gr + 2 x6,92gr + 8,03gr + 2 x3,41gr + 37gr +12,36 gr = 99 gr

Şimdi şekilde gösterdiğim 3 eklem noktasına göre moment alıp ihtiyacımız olan motor torklarını bulalım. Burada unutmamamız gereken şey gripper ile bir objeyi taşıyacak olduğumuza göre uç işlevcimize (end-effector ya da gripper) taşımayı istediğimiz objenin kütlesini eklememiz gerektiğidir. Taşımak istediğimiz objeyi maksimum 200gr olarak belirledik. Bu durumda üstteki sonuçları tekrar düzenlersek;

W1 = 69 gr
W2 = 122 gr
W3 = 99 gr
G = 200 gr

elde ederiz



Şekil 7.3 Tutulacak objenin SCD'de gösterilmesi

7.1.1 Omuz Motoru

Şimdi ;

Bu eklemlerimizi omuz, dirsek ve bilek olarak adlandıralım.

$\mathbf{d_1}$ = 120mm olduğuna göre $\mathbf{d_1}$ / 2 = 60 mm

$\mathbf{d_2}$ = 110mm olduğuna göre $\mathbf{d_2}$ / 2 = 55 mm

$\mathbf{d_3}$ = 116mm olduğuna göre $\mathbf{d_3}$ / 2 = 58 mm

Şimdi **Omuz**'a göre moment alırsak;

M1 = W1 x d1/2 +W2 x (d1+d2/2) +W3 x (d1 + d2 +d3/2) +G x (d1+d2+d3).....den

M1 = 69 x 60 + 122 x (120 + 55) + 99 x (120 + 110 + 58) + 200 x (120+110+116)=123202gr.mm = 12,32kg.cm

Demek ki omuz bölgesinde 3.2kg.cm tork kapasiteli 1 adet standart rc servo motor kullanırsak 200 gr'lık bir objeyi kaldıramayacağız. Sahip olduğumuz yapıda 2 adet standart servo motorunda işimizi görmeyeceği ortadadır. Bu yüzden Futaba'nın yüksek tork kapasitesine sahip S3350 modelini kullandık. Bu motorlar 6V verildiği taktirde 9kg.cm'lik tork değerlerine ulaşabiliyorlar. Bu da bu model iki motor kullanarak 18kg.cm'lik bir tork değerine ulaşabileceğimizi gösteriyor.

7.1.2 Dirsek Motoru

Şimdi **Dirsek**'e göre moment alırsak;

M2 = W2 x d2/2 + W3 x (d2 +d3/2) + G x (d1+d2 +d3) – W1 x d1/2…..den

M2 = 122 x 55 +99 x (110 + 60) + 200 x(120+110+116) – 69 x 60 = 88600gr.mm

8,86kg.cm

Demek ki dirsek ekseninde de standart bir servo motor ile 200 gr taşımak istediğimizde bunu gerçekleştiremeyeceğiz. Biz bu eksen için tork kapasitesi oldukça yüksek olan towerpro mg995 model bir rc servo motor kullandık. Tasarımda tek bir motor olarak kullanılmak üzere tasarlanan bir eksen olduğundan Futabas3350 model bir yüksek torklu servo motoru bile oldukça zorlayabileceği hesaplarda ortaya çıkmış bu yüzden 6V'da 13kgcm tork verebilen bu motor tercih edilmiştir.

7.1.3 Bilek Motoru

Şimdi **Bilek**'e göre moment alırsak;

M3 = W3 x d3/2 + G x d3 – W2 x d2/2 – W1 x (d2 + d1/2)….den

M3 = 99 x58+200 x116 –122 x55 – 69 x (110 + 60) = 10502grmm

1,05 kg.cm

Görüldüğü üzere bilek ekseninde 3,2kg.cm'lik standart bir servo motor işimizi rahatlıkla görecektir. Bu noktadan sonra z ekseninde dönme işlemini yapan motor için herhangi bir analiz yapmaya gerek yoktur. Çünkü bu eksen yerçekimine karşı bir iş yapmamakta z ekseni etrafında robotun hareket etmesini (dönmesini) sağlamaktadır. Benzer bir mantık gripper'ın açma kapama işlemini yapan motor için düşünülebilir. Bu eksende bir açma kapama işlemi yapılıp bir ağırlık kaldırma işlemi bulunmamaktadır.

NOT: Moment analizleri statik durum için ve robot yatay pozisyonda iken yapılmıştır. Yani burada yapılan analizler, robotun tüm uzuvları yere paralel konumda iken yapılmıştır.

BÖLÜM SEKİZ

SONUÇ

Bu bölüm ile birlikte 6 adet rc servo motor tarafından sürülen 4 eksen ve bir gripper' a sahip olan bir robot manipülatörün mekanik tasarımı yapılmış, üretim teknikleri göz önüne alınarak üretilmesi, kontrolleri, visual basic ile yazılan program ile arayüzden kumanda edilmesi veya istendiği takdirde mikroişlemci içine gömülen kodlarla karttan oto kontrolü sağlanmıştır. Bu manipülatörde yapılacak çalışmalar ile tutma bırakma, objeyi bir yerden alıp istenen başka bir noktaya götürüp bırakma işlemleri, pozisyon ve hız kontrolü, bir robot manipülatör için en önemli şey olan tekrarlanabilirlik (repeatability) iyileştirmeleri, gerekli hesaplar ve yazılıma arayüzdeki eklenecek kodlar ile birlikte ileri ve ters kinematik hesaplar yapılıp istenen pozisyona gitmesi için motorların alması gereken konumları bulmak, açılar verildiği taktirde de robotun çalışma uzayında gideceği pozisyonları bulmak gibi çalışmalar yapılabilir.

Bu çalışmada rc servo motorları bir motor sürücü kartı yaparak nasıl sürüldüğü ve bu motorlar kullanılarak tasarlanan bir robot manipülatörde kullanılması işlemi gerçekleştirilmiş ve üstün özellikleri nedeniyle robotik uygulamalarda kullanmak üzere rc servo motorların ne kadar başarılı oldukları görülmüştür.

Yazılan arayüz aracılığıyla seri porttan pc ve pic arasında iletişim kurulmuş ve bu sayede kullanılan asenkron ve senkron iletişim protokolleri üzerine çalışmalar yapılmıştır. Baskı devre tekniği kullanılarak prototip bir devre kartı tasarımının nasıl yapıldığı üzerine çalışılmış ve tezde kullanılan kartlarda yüzde yüz başarı elde edilmiştir. Tasarım için kullanılan Solidworks programında statik tork analizleri neticesinde elde edilen gerekli görülen tork değerlerine göre motor seçimleri yapılmış ve tasarımda buna göre optimizasyon çalışmaları yapılmış, dayanıklı (robust) bir sistem tasarımı ortaya çıkmıştır. Bir robot manipülatörden beklenen tekrarlanabilirlik (repeatability) incelenmiş ve robotun aynı pozisyona çok büyük bir başarımla gittiği gözlemlenmiştir.

KAYNAKLAR

Axelson, J. (1998). *Serial Port Complete Programming and Circuits for RS-232 and RS-485*. (1th ed.). Madison USA. Lakeview Research

Ayyıldız, S. (2006). *Jal İle Pic Programlama*. (1. Baskı). İstanbul. Altaş Yayıncılık

Bingül, Z., Küçük, S. (2005). *Robot Tekniği I*. (1. Baskı). İstanbul. Birsen Yayınevi

İbrahim, D. (2006). *Pc ve Pic Seri İletişim Projeleri*. (2. Baskı). İstanbul. Bileşim Yayınevi

Sanchez, J., Canton, M.P. (2007). *Microcontroller Programming the Microchip PIC*. (1th ed.). USA. CRC Press.

Sandin, P.E. (2003). *Robot Mechanisms And Mechanical Devices Illustrated*. (1th ed.). USA. McGraw - Hill

Şahin, H., Dayanık, A., Altınbaşak, C. (2006). *Pic Programlama Teknikleri Ve Pic16f877A*. (9. Baskı). İstanbul. Altaş Yayıncılık

EKLER

PIC16F877A'nın datasheet'inden alınmıştır.

17.0 ELECTRICAL CHARACTERISTICS

Absolute Maximum Ratings †

Ambient temperature under bias ..... -55 to +125°C
Storage temperature ..... -65°C to +150°C
Voltage on any pin with respect to Vss (except VDD, $\overline{MCLR}$, and RA4) ..... -0.3V to (VDD + 0.3V)
Voltage on VDD with respect to Vss ..... -0.3 to +7.5V
Voltage on $\overline{MCLR}$ with respect to Vss **(Note 2)** ..... 0 to +14V
Voltage on RA4 with respect to Vss ..... 0 to +8.5V
Total power dissipation **(Note 1)** 1.0W
Maximum current out of Vss pin ..... 300 mA
Maximum current into VDD pin ..... 250 mA
Input clamp current, IIK (VI < 0 or VI > VDD) ..... ± 20 mA
Output clamp current, IOK (VO < 0 or VO > VDD) ..... ± 20 mA
Maximum output current sunk by any I/O pin ..... 25 mA
Maximum output current sourced by any I/O pin ..... 25 mA
Maximum current sunk by PORTA, PORTB and PORTE (combined) **(Note 3)** ..... 200 mA
Maximum current sourced by PORTA, PORTB and PORTE (combined) **(Note 3)** ..... 200 mA
Maximum current sunk by PORTC and PORTD (combined) **(Note 3)** ..... 200 mA
Maximum current sourced by PORTC and PORTD (combined) **(Note 3)** ..... 200 mA

Note 1: Power dissipation is calculated as follows: Pdis = VDD x {IDD - ∑ IOH} + ∑ {(VDD - VOH) x IOH} + ∑(VOl x IOL)

2: Voltage spikes below Vss at the $\overline{MCLR}$ pin, inducing currents greater than 80 mA, may cause latch-up. Thus, a series resistor of 50-100Ω should be used when applying a "low" level to the $\overline{MCLR}$ pin rather than pulling this pin directly to Vss.

3: PORTD and PORTE are not implemented on PIC16F873A/876A devices.

† NOTICE: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at those or any other conditions above those indicated in the operation listings of this specification is not implied. Exposure to maximum rating conditions for extended periods may affect device reliability.

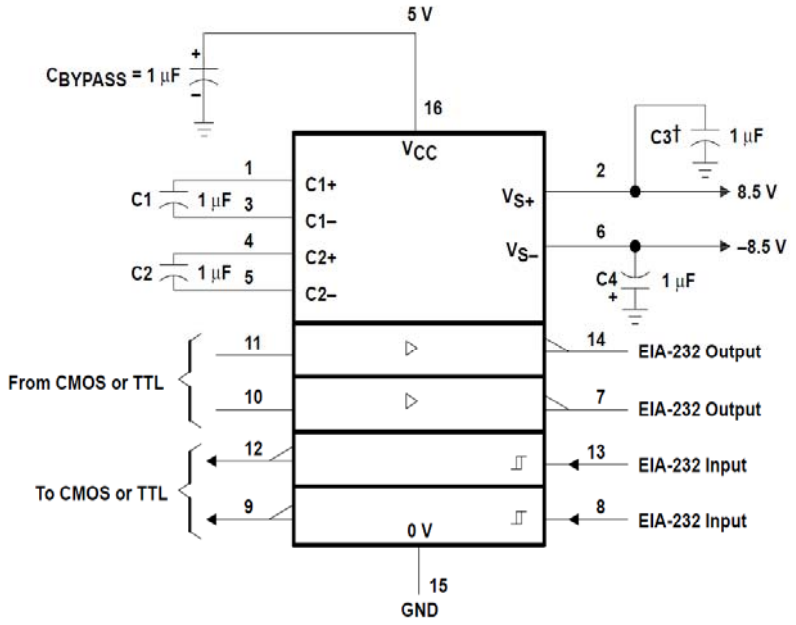
7805'in datasheet'inden alınmıştır.

Electrical Characteristics (KA7805/KA7805R)

(Refer to test circuit ,0°C < T_J < 125°C, I_O = 500mA, V_I =10V, C_I= 0.33µF, C_O=0.1µF, unless otherwise specified)

Parameter	Symbol	Conditions		KA7805 Min.	KA7805 Typ.	KA7805 Max.	Unit
Output Voltage	V_O	T_J =+25 °C		4.8	5.0	5.2	V
		5.0mA ≤ Io ≤ 1.0A, P_O ≤ 15W V_I = 7V to 20V		4.75	5.0	5.25	
Line Regulation (Note1)	Regline	T_J=+25 °C	V_O = 7V to 25V	-	4.0	100	mV
			V_I = 8V to 12V	-	1.6	50	
Load Regulation (Note1)	Regload	T_J=+25 °C	I_O = 5.0mA to1.5A	-	9	100	mV
			I_O =250mA to 750mA	-	4	50	
Quiescent Current	I_Q	T_J =+25 °C		-	5.0	8.0	mA
Quiescent Current Change	ΔI_Q	I_O = 5mA to 1.0A		-	0.03	0.5	mA
		V_I= 7V to 25V		-	0.3	1.3	
Output Voltage Drift	$\Delta V_O/\Delta T$	I_O= 5mA		-	-0.8	-	mV/ °C
Output Noise Voltage	V_N	f = 10Hz to 100KHz, T_A=+25 °C		-	42	-	µV/V_O
Ripple Rejection	RR	f = 120Hz V_O = 8V to 18V		62	73	-	dB
Dropout Voltage	V_{Drop}	I_O = 1A, T_J =+25 °C		-	2	-	V
Output Resistance	r_O	f = 1KHz		-	15	-	mΩ
Short Circuit Current	I_{SC}	V_I = 35V, T_A =+25 °C		-	230	-	mA
Peak Current	I_{PK}	T_J =+25 °C		-	2.2	-	A

MAX232'nin datasheetinden alınmıştır.



† C3 can be connected to V_{CC} or GND.

Figure 4. Typical Operating Circuit

FUTABA

S3003 RC SERVO MOTOR

Control System: +Pulse Width Control 1520usec Neutral

Required Pulse: 3-5 Volt Peak to Peak Square Wave

Operating Voltage: 4.8-6.0 Volts

Operating Temperature Range: -20 to +60 Degree C

Operating Speed (4.8V): 0.23sec/60 degrees at no load

Operating Speed (6.0V): 0.19sec/60 degrees at no load

Stall Torque (4.8V): 44 oz/in. (3.2kg.cm)

Stall Torque (6.0V): 56.8 oz/in. (4.1kg.cm)

Operating Angle: 45 Deg. one side pulse traveling 400usec

360 Modifiable: Yes

Direction: Counter Clockwise/Pulse Traveling 1520-1900usec

Current Drain (4.8V): 7.2mA/idle

Current Drain (6.0V): 8mA/idle

Motor Type: 3 Pole Ferrite

Potentiometer Drive: Indirect Drive

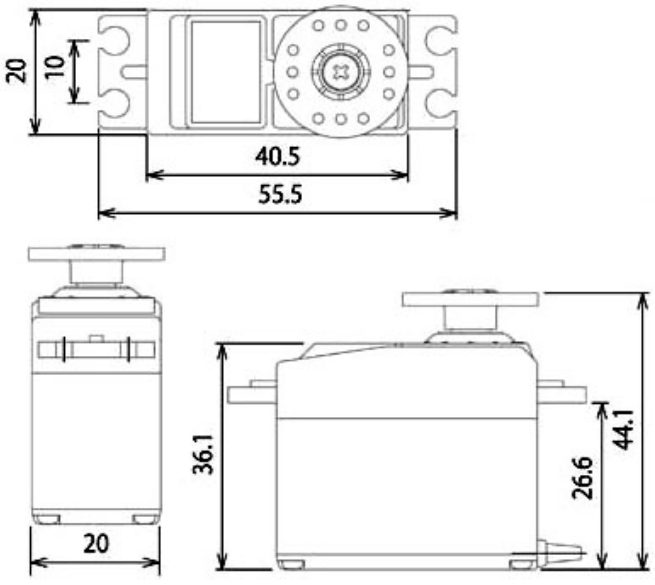
Bearing Type: Plastic Bearing

Gear Type: All Nylon Gears

Connector Wire Length: 12"

Dimensions: 1.6" x 0.8"x 1.4" (41 x 20 x 36mm)

Weight: 1.3oz. (37.2g)



FUTABA

S3305 RC SERVO MOTOR

Control System: +Pulse Width Control 1520usec Neutral

Required Pulse: 3-5 Volt Peak to Peak Square Wave

Operating Voltage: 4.8-6.0 Volts

Operating Temperature Range: -20 to +60 Degree C

Operating Speed (4.8V): 0.25sec/60 degrees at no load

Operating Speed (6.0V): 0.20sec/60 degrees at no load

Stall Torque (4.8V): 99 oz/in. (7.1kg.cm)

Stall Torque (6.0V): 124 oz/in. (8.9kg.cm)

Operating Angle: 45 Deg. one side pulse traveling 400usec

360 Modifiable: Yes

Direction: Counter Clockwise/Pulse Traveling 1520-1900usec

Motor Type: 3 Pole Ferrite

Potentiometer Drive: Indirect Drive

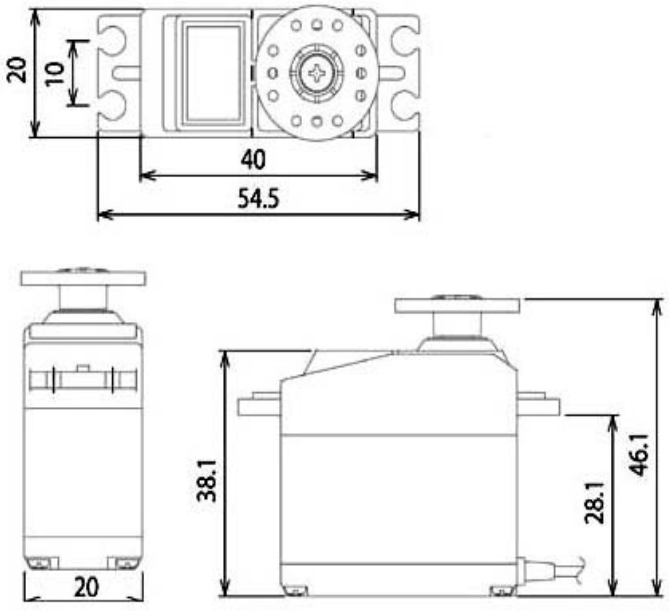
Bearing Type: Dual Ball Bearing

Gear Type: All Metal Gears

Connector Wire Length: 12"

Dimensions: 1.6" x 0.8"x 1.5" (41 x 20 x 38mm)

Weight: 1.64oz. (46.5g)



TOWERPRO

MG995 RC SERVO MOTOR

Dimensions: 40 x 20 x 36.5mm

Weight: 48g

Operating Speed (4.8V no load) : 0.17sec / 60 degrees

Operating Speed (6.0V no load) : 0.13sec / 60 degrees

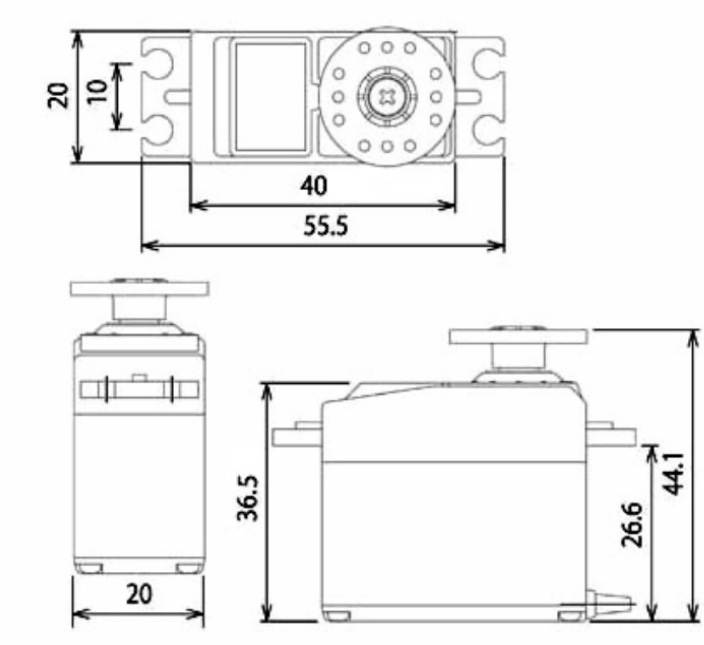
Stall Torque (4.8V): (13kg/cm) (180oz/in.)

Stall Torque (6.0V): (15kg/cm) (208oz/in.)

Temperature Range: -30 to +60 Degree C

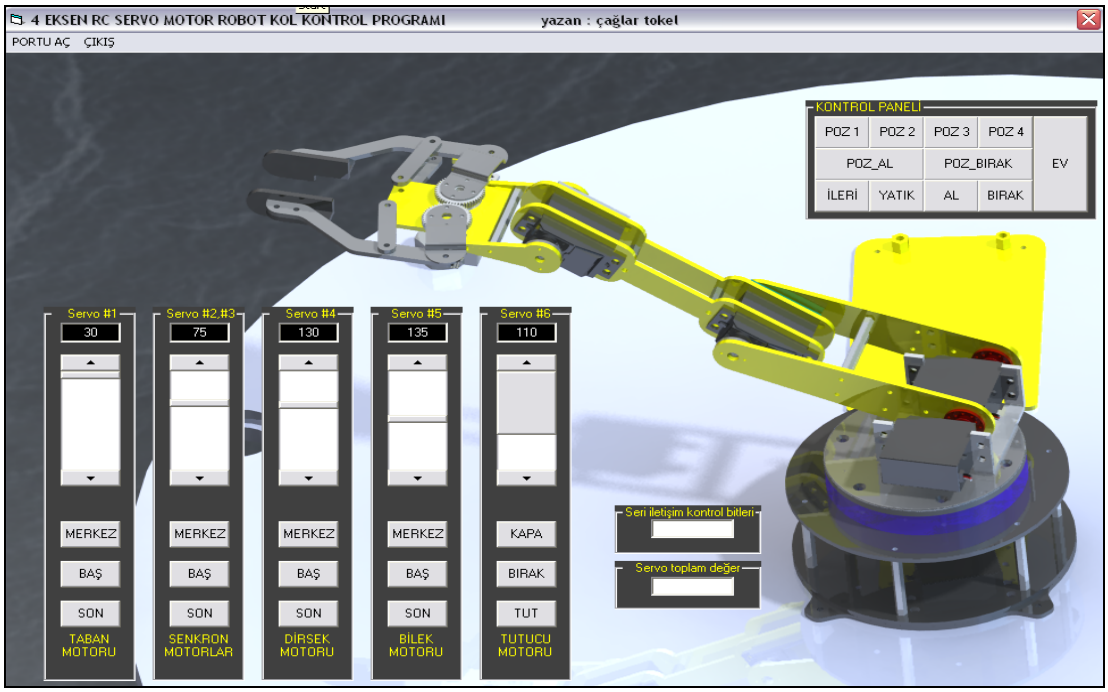
Dead Band Width: 4usec

Operation Voltage: 3.5 - 8.4Volts



Kontrol Programına Bir Kontrol Paneli Ekleyerek Robot Manipülatörün İstenen Pozisyonlara Gitmesini Sağlamak İçin Yapılan Geliştirme

Robot manipülatörü kontrol etmek için yazılan programda yapılan geliştirme ile yapması istenen hareket için motorların alması gereken pozisyonlarının PIC mikrodenetleyicimiz içerisine yazılan koda gömülmesi ile elde edilen yeni arayüz.



Bu işlem için yapılması gereken aşağıda gösterilen kod parçacıklarını yazılımda anlatılan kodlar içine gömmektir.

1. Visual basic ile yazılan koda yapılan ekleme
2. PIC mikrodenetleyicisi içerisindeki koda yapılan ekleme

1. Visual basic ile yazılan programda yapılması gereken değişiklik

```
Private Sub sldServo_Change(Index As Integer)

txtsrv(Index).Text = Str(sldservo(Index).Value)
```

```
Text6.Text = Val(Str(sldservo(0).Value)) + Val(Str(sldservo(1).Value)) +
Val(Str(sldservo(2).Value)) + Val(Str(sldservo(3).Value)) +
Val(Str(sldservo(4).Value))

MSComm1.Output = "a"
wait (300)

'MSComm1.InBufferCount = 0

MSComm1.Output = Chr(sldservo(0).Value)
wait (300)



MSComm1.Output = "b"
wait (300)

'MSComm1.InBufferCount = 0

MSComm1.Output = Chr(sldservo(1).Value)
wait (300)

MSComm1.Output = "c"
wait (300)

MSComm1.Output = Chr(sldservo(2).Value)
wait (300)

MSComm1.Output = "d"
wait (300)

MSComm1.Output = Chr(sldservo(3).Value)
wait (300)
```

```
MSComm1.Output = "e"
wait (300)
MSComm1.Output = Chr(sldservo(4).Value)
wait (300)

Text9.Text = MSComm1.Input

End Sub

Private Sub ileri_Click()
  MSComm1.Output = "i"
End Sub

Private Sub poz1_Click()
  MSComm1.Output = "y"
End Sub

Private Sub yatik_Click()
  MSComm1.Output = "o"
End Sub

Private Sub al_Click()
  MSComm1.Output = "j"
End Sub

Private Sub pozal_Click()
  MSComm1.Output = "h"
End Sub

Private Sub poz_birak_Click()
  MSComm1.Output = "t"
End Sub
```

```
Private Sub birak_Click()
  MSComm1.Output = "k"
End Sub

Private Sub ev_Click()
  MSComm1.Output = "n"
End Sub

Private Sub poz2_Click()
  MSComm1.Output = "p"
End Sub

Private Sub poz3_Click()
  MSComm1.Output = "v"
End Sub

Private Sub poz4_Click()
  MSComm1.Output = "z"
End Sub
```

Not : Visual basic'teki kodun bütünü bu kısımda gösterilmemiş olup, sadece içinde değişiklik yapılan "**Private Sub sldServo_Change(Index As Integer)**" Subroutine'i verilmiştir.

2. PIC mikrodenetleyicisi içerisindeki koda yapılan ekleme

```
include 16f877_4
include jlib
include h_rs232
include rc_servo_k

var byte dd,c1,c2,c3,c4,c5,c6,d1,d2,d3,d4,d5,ct,x = 0

pin_b4_direction = output
port_d_direction = all_output
disable_a_d_functions

pin_d0 = low
pin_d1 = low
pin_d2 = low
pin_d3 = low
pin_d4 = low
pin_d5 = low
pin_d6 = low
pin_b4 = low

const  srv1_bas = 35 , srv1_nrm = 120 ,srv1_bit  = 235,
       srv2_bas = 35 , srv2_nrm = 120 ,srv2_bit  = 235,
       srv3_bas = 35 , srv3_nrm = 120 ,srv3_bit  = 235,
       srv4_bas = 50 , srv4_nrm = 130 ,srv4_bit  = 200,
       srv6_ac  = 65 , srv6_nrm = 120 ,srv6_kapa = 155
```

```
procedure robot_konumla is    -- robot sürekli olarak bu hareketi yapsin diye
istedigimiz bir prosedür

srv5 = srv2

rc_servo_6

end procedure

procedure ileri is

srv1 = srv1_nrm robot_konumla
srv2 = 140 robot_konumla   -- senkron motor 1
srv3 = srv3_nrm robot_konumla  -- dirsek motoru
srv4 = srv4_nrm robot_konumla  -- bilek motoru
-- srv5 = srv5_nrm  robot_konumla   -- senkron motor 2
-- srv6 = srv6_nrm robot_konumla    -- gripper motoru

end procedure
procedure poz_1 is

srv1 = srv1_bit robot_konumla   -- base döndürme işlemi servo motoru
srv2 = 95 robot_konumla   -- senkron motor 1
srv3 = 165 robot_konumla    -- dirsek motoru
srv4 = 90 robot_konumla    -- bilek motoru
-- srv5 = srv5_nrm  robot_konumla   -- senkron motor 2
-- srv6 = srv6_nrm robot_konumla
end procedure

procedure yatik is

srv1 = 165 robot_konumla   -- base döndürme işlemi servo motoru
```

```
srv2 = 180 robot_konumla   -- senkron motor 1
srv3 = 105 robot_konumla
srv4 = 145 robot_konumla
-- srv5 = srv5_nrm  robot_konumla  -- senkron motor 2
-- srv6 = srv6_nrm robot_konumla
end procedure

procedure poz_al is
srv6 = 85 robot_konumla
srv4 = 105 robot_konumla
srv1 = 115 robot_konumla   -- base döndürme işlemi servo motoru
srv2 = 185 robot_konumla   -- senkron motor 1
srv3 = 80 robot_konumla

-- srv5 = srv5_nrm  robot_konumla   -- senkron motor 2

end procedure

procedure poz_birak is

srv1 = 200 robot_konumla   -- base döndürme işlemi servo motoru
srv2 = 170 robot_konumla   -- senkron motor 1
srv3 = 105 robot_konumla
srv4 = 145 robot_konumla
-- srv5 = srv5_nrm  robot_konumla   -- senkron motor 2

-- srv6 = srv6_ac
end procedure


procedure al is
```

```
-- srv1 = 60 robot_konumla   -- base döndürme işlemi servo motoru
-- srv2 = 130 robot_konumla   -- 1.senkron servo motoru
-- srv3 = 160 robot_konumla   -- kritik eksen servo motoru
-- srv4 = 150 robot_konumla   -- gripper yukarı assagı servo motoru
-- srv5 = srv5_nrm  robot_konumla ( 10 )  -- 2.senkron servo motoru
srv6 = 85 robot_konumla

srv6 = srv6_kapa robot_konumla   -- gripper servo motoru

end procedure

procedure birak is

-- srv1 = 60 robot_konumla   -- base döndürme işlemi servo motoru
-- srv2 = 130 robot_konumla   -- 1.senkron servo motoru
-- srv3 = 160 robot_konumla   -- kritik eksen servo motoru
-- srv4 = 150 robot_konumla   -- gripper yukarı assagı servo motoru
-- srv5 = srv5_nrm  robot_konumla ( 10 )  -- 2.senkron servo motoru
srv6 = 80 robot_konumla

end procedure

procedure ev is

 srv1 = 45 robot_konumla    -- base döndürme işlemi servo motoru
 srv2 = 120 robot_konumla   -- 1.senkron servo motoru
 srv3 = 120 robot_konumla   -- kritik eksen servo motoru
 srv4 = 120 robot_konumla   -- gripper yukarı assagı servo motoru
 srv5 = 120 robot_konumla   -- 2.senkron servo motoru
 srv6 = 100 robot_konumla

end procedure
```

```
procedure poz_2 is

srv1 = srv1_bit robot_konumla   -- base döndürme işlemi servo motoru
srv2 = 165 robot_konumla   -- senkron motor 1
srv3 = 85 robot_konumla    -- dirsek motoru
srv4 = 145 robot_konumla    -- bilek motoru
-- srv5 = srv5_nrm  robot_konumla   -- senkron motor 2
-- srv6 = srv6_nrm robot_konumla

end procedure

procedure poz_3 is

srv1 = 165 robot_konumla   -- base döndürme işlemi servo motoru
srv2 = 135 robot_konumla   -- senkron motor 1
srv3 = 180 robot_konumla    -- dirsek motoru
srv4 = 90 robot_konumla    -- bilek motoru
-- srv5 = srv5_nrm  robot_konumla   -- senkron motor 2
-- srv6 = srv6_nrm robot_konumla

end procedure

procedure poz_4 is

srv4 = 175 robot_konumla    -- bilek motoru
srv1 = 150 robot_konumla   -- base döndürme işlemi servo motoru
srv2 = 85 robot_konumla   -- senkron motor 1
srv3 = 210 robot_konumla    -- dirsek motoru
srv1 = 35 robot_konumla

-- srv5 = srv5_nrm  robot_konumla   -- senkron motor 2
-- srv6 = srv6_nrm robot_konumla
```

```
end procedure

-- -----------------------------------------------------------------------------
  serial_setup ( 96 )

  srv1 = 45 srv2 = 120 srv3 = 120 srv4 = 120 srv5 = 120 srv6 = 100   -- bu kısım
klavye kontrolünde aktif hale getirilecek

forever loop
 if h_asynch_poll ( dd )   then
      pin_b4  = high
      -- delay_100us(5)

      if dd == "i" then ileri     end if

      if dd == "y" then poz_1     end if

      if dd == "o" then yatik     end if

      if dd == "j" then al        end if

      if dd == "h" then poz_al    end if

      if dd == "t" then poz_birak end if

      if dd == "k" then birak     end if

      if dd == "n" then ev        end if

      if dd == "p" then poz_2     end if

      if dd == "v" then poz_3     end if
```

```
if dd == "z" then poz_4     end if

if      x == 0  & dd == "a"    then d1 = dd  x = x + 1  h_asynch = "c" -- "c"
karakteri alınır
 delay_1ms(5)

 elsif      x == 1   then c1 = dd  x = x + 1
 delay_1ms(5)

 elsif      x == 2  & dd == "b"   then d2 = dd h_asynch = "a"  x = x + 1
 delay_1ms(5)

 elsif      x == 3   then c2 = dd  x = x + 1
 delay_1ms(5)


 elsif      x == 4  & dd == "c"  then d3 = dd  x = x + 1 h_asynch = "g"
 delay_1ms(5)

 elsif      x == 5   then c3 = dd  x = x + 1

 delay_1ms(5)

 elsif      x == 6  & dd == "d"  then d4 = dd  x = x + 1 h_asynch = "l"
 delay_1ms(5)

 elsif      x == 7   then c4 = dd  x = x + 1

 delay_1ms(5)

 elsif      x == 8  & dd == "e"  then d5 = dd  x = x + 1 h_asynch = "a"
 delay_1ms(5)
```

```
elsif      x == 9   then c5 = dd  x = x + 1

        delay_1ms(5)

         else x = 0

         end if

 end if

if x == 10 then

       srv1 = c1  -- base servo motoru
       srv2 = c2  --  senkron servo motorlar eşit olması gerektiginden esitlenmistir
       srv5 = c2  --  senkron servo motorlar eşit olması gerektiginden esitlenmistir
       srv3 = c3  -- dirsek eklemdeki servo motor
       srv4 = c4  -- bilek yukarı assagı servo motoru
       srv6 = c5  -- gripper aç kapa servo motoru

   pin_b4 = low  -- b4 pinine baglı led söndürülür.test amaçlıdır.eger led sönerse x =
10 olmus ve servo degişlenlerine degerlerinin atanmıs oldugu sonucunu çıkarırız.

    x = 0
 end if

rc_servo_6 ( 1 )

end loop
```