GEREĞİNDEN ÇOK SERBESTLİK DERECELİ ROBOT KOLU KONTROL SİSTEMİ TASARIMI VE UYGULAMASI

İlker EREN

Haziran 2006
DENİZLİ

GEREĞİNDEN ÇOK SERBESTLİK DERECELİ ROBOT KOLU KONTROL SİSTEMİ TASARIMI VE UYGULAMASI

Pamukkale Üniversitesi
Fen Bilimleri Enstitüsü
Yüksek Lisans Tezi
Elektrik-Elektronik Mühendisliği Anabilim Dalı

İlker EREN

1.Danışman: Yard. Doç. Dr. Abdullah T. TOLA

2. Danışman: Doç. Dr. E. Şahin ÇONKUR

Haziran, 2006
DENİZLİ

YÜKSEK LİSANS TEZİ ONAY FORMU

İlker EREN tarafından Yard. Doç. Dr. Abdullah T. TOLA ve Doç. Dr. E. Şahin ÇONKUR yönetiminde hazırlanan **"Gereğinden Çok Serbestlik Dereceli Robot Kolu Kontrol Sistemi Tasarımı Ve Uygulaması"** başlıklı tez tarafımızdan okunmuş, kapsamı ve niteliği açısından bir Yüksek Lisans Tezi olarak kabul edilmiştir.

Doç. Dr. E. Şahin ÇONKUR
Jüri Başkanı

Yard. Doç. Dr. Murat AYDOS
Jüri Üyesi

Yard. Doç. Dr. Abdullah T. TOLA
Jüri Üyesi(Danışman)

Yard. Doç. Dr. Serdar İPLİKÇİ
Jüri Üyesi

Yard. Doç. Dr. Sezai TOKAT
Jüri Üyesi

Pamukkale Üniversitesi Fen Bilimleri Enstitüsü Yönetim Kurulu'nun
...../...../.....tarih ve .....sayılı kararı ile onaylanmıştır.

Prof. Dr. Mehmet Ali SARIGÖL
Müdür

TEŞEKKÜR

Yüksek Lisans öğrenimim sırasında ve tez çalışmalarım boyunca gösterdiği her türlü destek ve yardımlardan dolayı danışman hocam Yard. Doç. Dr. Abdullah T. TOLA'ya, ikinci danışman hocam Doç. Dr E. Şahin ÇONKUR'a, Makine Mühendisliği Bölümü yüksek lisans öğrencisi İsmail BOZTAY'a, Elektrik Elektronik Mühendisliği Bölümü yüksek lisans öğrencisi Nesrin KILIÇ'a ve ortağım Ali APALI'ya en içten dileklerimle teşekkür ederim.

Bu çalışma boyunca yardımlarını esirgemeyen Fen Bilimleri Enstitüsü çalışanlarına, projede kullanılan donanımları sağlayan TÜBİTAK'a, iş arkadaşlarıma ve yüksek lisans dönemi sınıf arkadaşlarıma teşekkürü borç bilirim.

Bu tezin tasarımı, hazırlanması, yürütülmesi, araştırılmasının yapılması ve bulgularının analizlerinde bilimsel etiğe ve akademik kurallara özenle riayet edildiğini; bu çalışmaların doğrudan birincil ürünü olmayan bulguların, verilerin ve materyallerin bilimsel etiğe uygun olarak kaynak gösterildiğini ve alıntı yapılan çalışmalara atfedildiğini beyan ederim.

İmza : 
Öğrenci Adı Soyadı : İlker EREN

ÖZET

GEREĞİNDEN ÇOK SERBESTLİK DERECELİ ROBOT KOLU KONTROL SİSTEMİ TASARIMI VE UYGULAMASI

EREN, İlker
Yüksek Lisans Tezi, Elektrik Elektronik Mühendisliği ABD
Tez Yöneticileri: Yard. Doç. Dr. Abdullah T. TOLA ve Doç. Dr. E. Şahin ÇONKUR

Haziran 2006, 49 Sayfa

Bu tezin amacı, gereğinden çok serbestlik dereceli bir robot kolunun bilgisayardan kontrolü için gerekli kontrol sisteminin tasarımının ve uygulamasının yapılmasıdır.

Bu çalışma TÜBİTAK tarafından desteklenen, tasarımı ve prototipi üniversitemizde yapılan ilk sanayi tipi robotu içeren projenin bir parçasıdır. Prototipin tahrik mekanizması AC servo motorlardan oluşmaktadır. Visual C++.NET bilgisayar programı içinden bir OCX program parçacığı vasıtasıyla eksen kontrol kartına emir yollanmakta, kart bu emirleri servo sürücülere iletmekte ve servo sürücüler de motorları sürmektedir.

Bu tezde, motorların kurulması, ayarlanması ve kontrol kartıyla bilgisayardan kontrolü gerçekleştirilmiştir. Buna ek olarak, bu prototipte kullanılan kontrol algoritmaları incelenmiştir. Robotun mafsalları üzerinde ek bir enkoder yerleştirilebilmesi için gerekli çalışmalar yapılmış, bu enkoderlerden alınan bilgi bilgisayar programına aktarılmıştır. Benzer şekilde robotun uç noktasına takılan bir lazer sensörüyle seri port vasıtasıyla iletişim kurulmuş ve sensör verisi bilgisayar programına aktarılmıştır.

Anahtar Kelimeler: Robot, Robot Kontrol Algoritmaları, Eksen Kontrol, Servo Motor, Servo Sürücü

Doç. Dr. E. Şahin ÇONKUR
Yard. Doç. Dr. Abdullah T. TOLA
Yard. Doç. Dr. Murat AYDOS
Yard. Doç. Dr. Serdar İPLİKÇİ
Yard. Doç. Dr. Sezai TOKAT

ABSTRACT

CONTROL SYSTEM DESIGN AND APPLICATION OF A REDUNDANT ROBOT MANIPULATOR

EREN, İlker
M. Sc. Thesis in Electrical&Electronics Engineering
Supervisors: Asst. Prof. Dr. Abdullah T. TOLA and Assoc. Prof. Dr. E. Şahin ÇONKUR

June 2006, 49 Pages

The aim of thesis is to design and implement a control system that is able to control a redundant robot by means of a computer.

This work is a part of the project supported by TUBITAK, including the first industrial type robot that are designed and manufactured in our university. The driving mechanism of the prototype consists of AC servo motors. Using an OCX component embedded in the Visual C++.NET program, communication with the motion control card connected to the computer is achieved. The motion control card sends the commands to the servo drivers and the servo drivers drive the servo motors.

In this thesis, servo motor setup, servo tuning and control of the motors with the motion control card have been carried out. Furthermore, control algorithms used in the prototype has been considered. By doing some work on the encoders which are attached to the joints of the robot, the information obtained from these encoders has been carried to the computer. Similarly, communication with a laser sensor by means of the serial port has been established and the information obtained from the sensor has been carried to the computer.

Keywords: Robot, redundant robots, robot control algorithms, motion control, servo motor, servo drive

Assoc. Prof. Dr. E. Şahin ÇONKUR
Asst. Prof. Dr. Abdullah T. TOLA
Asst. Prof. Dr. Murat AYDOS
Asst. Prof. Dr. Serdar İPLİKÇİ
Asst. Prof. Dr. Sezai TOKAT

İÇİNDEKİLER

Sayfa

ŞEKİLLER DİZİNİ

Sayfa

TABLOLAR DİZİNİ

Sayfa

SİMGE VE KISALTMALAR DİZİNİ

PWM	Pulse Wide Modulation
RPM	Revulotion Per Minute
A	Amper
DSP	Digital Signal Processor
MDI	Multiple Documet Interface
DOF	Degrees Of Freedom

1. GİRİŞ

Sanayinin her alanında baş döndürücü bir hızla kullanılmaya başlanan robotlar, insanların yapabileceklerinin çok daha iyisini, kalitelisini ve hızlısını yapabilmektedirler. Üretimin bütün aşamalarında sanayi robotları kullanılmakta ve büyük zaman ve para tasarrufu sağlamaktadırlar.

Her geçen gün robotlar, kullanılan hareket algoritmalarının, servo motorların, sürücü ve diğer donanımların geliştirilmesiyle daha hızlı, daha kabiliyetli ve daha akıllı olmaktadırlar. Sanayi robotlarının mekanik dizaynının ilerlemesine ek olarak hareket algoritmalarının geliştirilmesi de çok önemlidir. Ayrıca algoritmadan gelen bilgilere göre robotun uzuvlarını hareket ettirecek elektronik donanımın kurulması ve çalıştırılması da gereklidir.

Bu tez çalışmasında, ard arda eklenen kollardan oluşan bir robot kolu için hareket algoritması geliştirilmesi ve servo motorların çalıştırılması için gereken elektronik donanımın kurulup çalıştırılması amaçlanmıştır.

Robot kolu için bir hareket algoritması Visual C++.NET programlama dilinde yazılmıştır. Bilgisayarın fare imlecini hedef kabul eden robot, imleci ekranda ve çalışma uzayında takip edebilmektedir. Ek olarak sekiz adet servo motorun çalıştırılıp bilgisayardan kontrol edilebilmesi için bir elektrik panosu döşenmiştir. Bilgisayarda çizim halindeki robotun eklemlerinin dönme miktarı kadar servo motorların da döndürülmesi gerçekleştirilmiştir.

1.1. Literatürün Gözden Geçirilmesi

Gereğinden çok serbestlik dereceli robot kolları, kendi uzuv değişkenlerine sonsuz sayıda çözüm üretebilen robot kolları olarak tanımlanır. Bu robot kolları iç ve dış engeller ortaya çıktığında değişik konfigürasyonlar seçerek bu engeller arasından geçebilirler. (E. S. Conkur ve R. Buckingham,1997) Başlıca kullanım alanları şu şekilde sıralanabilir(Ma S., Hirose S., Yoshinada H. 1995):

- Büyük makinelerin içlerine tamir ve bakım için girebileceklerdir.
- Serbestlik dereceleri sınırlı olan ve çalışma ortamları çok itinayla hazırlanması gereken günümüzdeki endüstriyel robotların hareket kabiliyetlerini çok fazla arttırabileceklerdir.
- Uzay istasyonu inşası gibi el becerisi, mikro-elektronik imalatı gibi vakum ortamı gerektiren çok çeşitli işleri yapabileceklerdir.
- Bazı beyin ameliyatlarında, cerrahın elle ulaşmasının çok zor olduğu beyin kısımlarına ulaşmada kullanılabileceklerdir.
- Depremde yıkık altında kalmış canlıların yer tespitini yapabilecek ve yıkıklar arasında kendi yolunu bulup ilerleyerek onlara ilk müdahaleyi yapabilecektir.

Bu alanlar çok daha fazla genişletilebilir. Bu tür robotlar, insanların ulaşması zor veya imkansız olduğu bölgelere girerek, el becerisi ve zeka gerektiren fakat insanların yapması zor ve tehlikeli olan birçok işi otomatik olarak yapabileceklerdir. Bu gibi işleri başaran bir robot kolu, hem zaman ve para tasarrufu sağlayacak hem de insanları bu işleri yaparken karşılaşacakları tehlikelerden koruyacaktır.

Çalışma alanında robotun uç noktasının yörüngesi verildiğinde geçerli bir mafsal yörüngesinin hesabına *gereğinden çok eklemli çözümleme* denir (A. A. Maciejewski ve C. A. Klein, 1985). Bu çözümleme sınıfında, *gradyan izdüşümü tekniği* boş uzay ile çeşitli performans kriterleri uygulayarak robot uzuvlarının kendi iç hareketini belirler. *Genişletilmiş Jacobian tekniği*, mafsal uzayı ile görev uzayı arasındaki ilişkiyi tanımlanan ek sınırlamaları kullanarak tam belirli olmayan bir sistemi belirli bir sistem haline dönüştürür (D. N. Nenchev,1989). Ek kinematik sınırlamalar engelden kaçınma

için de tanımlanabilir (C. L. Boddy ve J. D. Taylor, 1993). Gereğinden çok eklemli çözümleme çevredeki değişikliklere çok çabuk tepki verebilir, fakat yörünge planlaması bakımından yeterliliği tartışılır. Bu teknikler esas olarak yerel tekniklerdir, yani üretilen çözümler istenen hareket alanı dar olduğunda geçerli olur. Fakat az sayıda olmakla beraber gereğinden çok eklemli çözümlemeyi global olarak kullanan teknikler de vardır (Y. Nakamura, 1991).

Eğer görev uç noktanın belirli bir hedef noktaya ulaşması olarak verilirse, robot mafsallarının yörünge hesaplaması *yörünge planlama problemi* olarak isimlendirilir ve *hareket planlaması* içinde değerlendirilir (J. Latombe, 1991). Geometrik hareket planlama algoritmaları robotun tamamı için engellerle çarpışmayan yörüngeler hesaplayabilir. Birçok hareket planlama algoritması arasında öne çıkan genel yaklaşımlar *yol haritaları, hücre ayrıştırma ve potansiyel alan* metotlarıdır. Bu yaklaşımlar hem çalışma uzayında hem de konfigürasyon uzayında uygulanabilir. Çalışma uzayı robotun içinde hareket ettiği üç boyutlu uzayı temsil ederken, konfigürasyon uzayı robotun mümkün olan bütün konfigürasyonlarını temsil eder. Robotun çalışma uzayındaki yörünge planlaması konfigürasyon uzayında bir noktanın yörünge planlamasına indirgenir (N. Amato ve Y. Wu, 1996). Yol haritaları çalışma alanının serbest bölgeleri arasındaki bağlantıları tek boyutlu doğrular setine indirger. Yol haritaları, görünürlük grafikleri, Voronoi diyagramları ve serbest yol ağları ile oluşturulur. Bu yöntemin önemli bir dezavantajı verimsizliğe sebep olan çok sayıda düğüm içerebilmesidir. Hücre ayrıştırma metodu serbest bölgeleri hücrelere ayırır ve hücreler arasındaki bitişiklikleri temsil eden bağlantı grafiğini oluşturur. Bu grafik daha sonra hedef noktası ile başlangıç noktasını bağlayan birbirine bitişik bir hücre grubu bulmak için taranır (A. Hayashi, 1994).

Potansiyel alan metodunda, çalışma alanı suni bir potansiyel alanın etkisi altında tutulur. Engeller itme etkisi verirken hedef noktası çekme etkisi oluşturur. Bu iki etkinin negatif gradyanının toplamı, robot uzuvlarındaki kontrol noktaları vasıtasıyla robot hareketinin kontrolünde kullanılır. Potansiyel alan metodunda en büyük problem, robot hedefe varmadan önce yerel minimumlardan birinde takılıp kalmasıdır. Bu soruna değişik çözümler düşünülmüştür. Bunlardan biri yerel minimumları arayıp bularak devre dışı bırakmaktır. Bir diğeri de yerel minimumları olmayan potansiyel alanlar

oluşturmaktır. Potansiyel alan metodu gerçek zamanlı yerel uygulamalarda kullanılabilir. Fakat engellerin sayısı arttığında engele çok yaklaşmak imkansız hale gelebilmektedir. Potansiyel alan metodu global olarak da uygulanabilir. Bu, sayısal potansiyel alanların yerel minimumsuz olarak bir ızgara üzerinde tanımlanması ile olur. Global bir yörünge için iyi bilinen yöntemleri bir şekilde kullanan bazı algoritmalar vardır. Fakat bu algoritmalar uç nokta yerine robot uzuvlarının yörünge planlamasını yapar ve bunun için farklı yöntemler kullanır. Örneğin, sensör verisi kullanarak ve robot uzuvlarının engellere hafif dokunmasına izin vererek robotun kinematik kontrolü başarılmıştır (D. Reznik ve V. Lumelsky, 1995). Robot konfigürasyonunu bir omurga eğrisine uygun hale getiren bir kontrol modeli oluşturulmuştur. Belirli bir eğriye robot konfigürasyonunu uygun hale getirmek için yeni kinematik denklemler geliştirilmiştir. Yörünge planlaması penaltı fonksiyonlarını içeren bir dizi minimizasyon problemi olarak incelenmiştir. Sonsuz derecede esnek robot kontrolü, Catmull-Rom eğrileri veya elipsoidler kullanılarak kontrol edilmiştir. Konfigürasyon uzayının hesap zorluklarını hafifletmek için boyutu konfigürasyon uzayından daha az olan duruş uzayı tasarlanmıştır. Gereğinden çok serbestlik dereceli bir robot dizayn edilmiş ve duruş uzayı kullanılarak kontrol edilmiştir (S. Ma ve I. Kobayashi, 2000). Gereğinden çok serbestlik dereceli robot kollarının mekanik dizaynı ile ilgili literatürde az sayıda yayın vardır. Mekanik dizayn genelde üç kategoride incelenir. Birincisi uzuvları birbirine eklenerek oluşturulan seri uzuv kollardır. İkincisi uzuvlar yerine örneğin hidrolik silindirler kullanılarak şekli değiştirilebilen yapılar kullanılan robotlardır. Üçüncüsü ise ardı ardına eklenen modüllerden oluşan robotlardır. Ayrıca, endoskop gibi tamamen esnek yapılar da vardır (P. J. Choi, J. A. Rice ve J. C. Cesarone, 1993)

1.2. Tezin Amacı

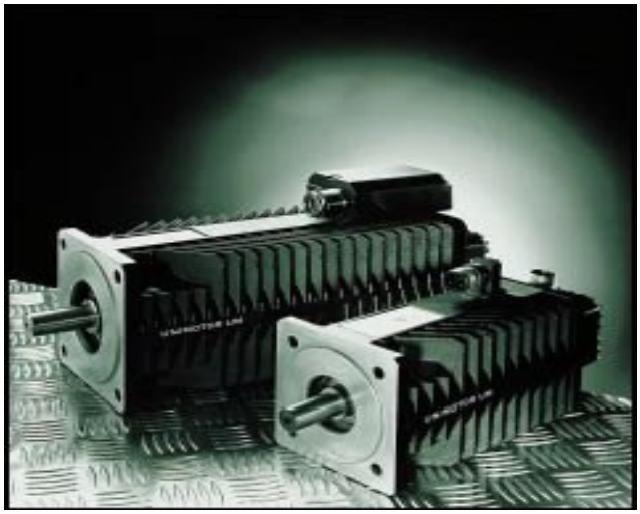
Günümüzde üretilen teknolojinin kullanılmasının yanı sıra teknoloji üretmenin önemi, teknoloji üreten ülkelerin lider olduğu dünya dengelerinde her geçen gün daha da artmaktadır. Bu bağlamda 21. yüzyılın robot çağı olacağı, robotların sadece sanayi ortamında kalmayıp, günlük yaşamın içinde yer alacağı düşünülerek, kendi hareket algoritmalarımızı ve yazılımlarımızı yazarak, robot üretim teknolojisinin ülkemize kazandırılmasında alt yapıya destek olacak çalışmaların yapılmasını amaçladık.

2. ELEKTRİK VE ELEKTRONİK DONANIM

2.1. Servo Motorlar

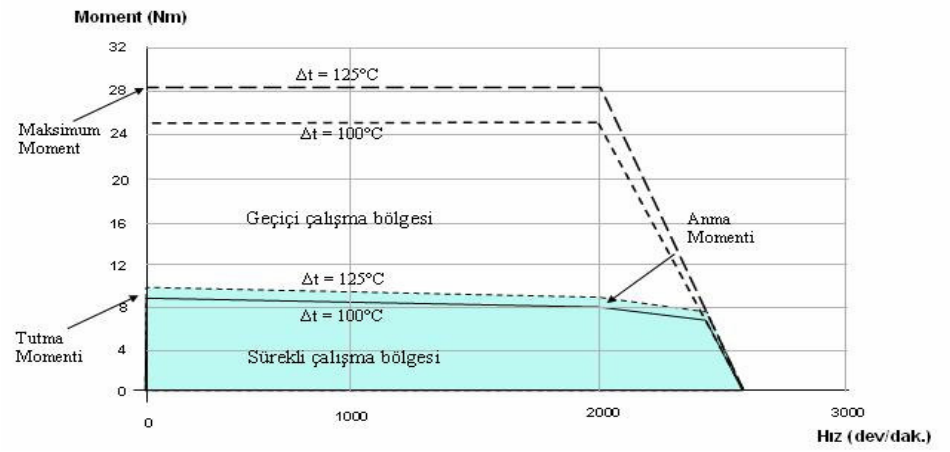
2.1.1. Servo Motor Genel Özellikleri

Servo motorlar, dinamik yük ve hız değişikliği, pozisyonlama, peryodik çalışma, yüksek kararlılık ihtiyaçlarında kullanılırlar. Rotorunda sabit mıknatıslar bulunan, modern elektronik sürücüler ile kontrol edilen senkron motorlardır. DC servo motorlardaki gibi komutatör ve fırça elemanları olmadığından güvenilir, kararlı ve küçük boyutlarda imal edilirler. Üç faz sargılarına uygulanan sinüs şeklindeki akım ile hava aralığında bir döner alan oluşturulur (Şekil 2.1).



Şekil 2.1 Servo motor (WEB_1, 2006)

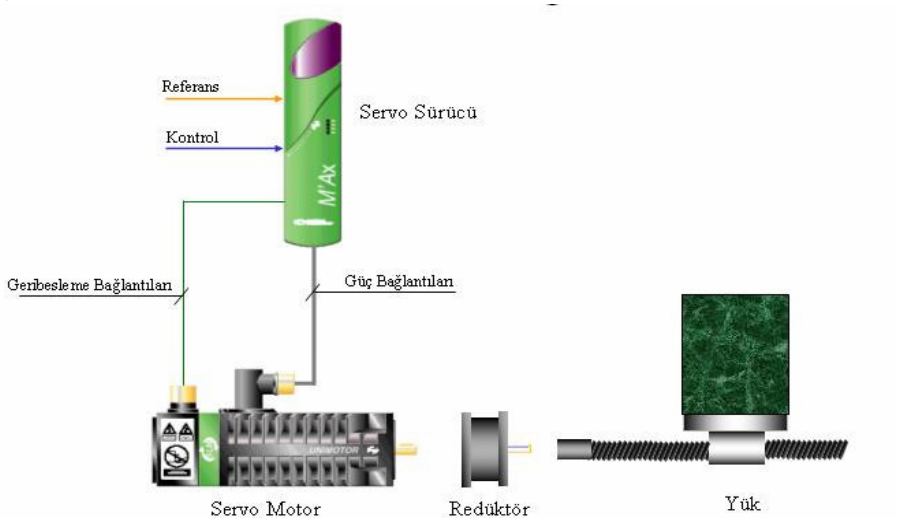
Servo motorların en önemli özelliklerinden birisi sıfır devirde nominal moment değerlerini verebilmesidir. Bu özellik Şekil 2.2'de gösterilmiştir.



Şekil 2.2 Servo motor özeğrileri (WEB_2, 2006)

2.1.2. Servo Motor Bağlantıları

Servo motorda, güç ve geribesleme kablosu olmak üzere iki kablo bağlantısı vardır. Güç kablosu, servo sürücüye bağlanarak 3 faz dalga genişlik modülasyonu ile enerji beslemesi yapılmaktadır. Geribesleme elemanı ise servo motorun milinin konum bilgisini vermektedir. Uygulamaya göre en yaygın olarak enkoder, resolver v.b. donanımlar geribesleme elemanı olarak seçilmekte ve bu donanımlar motorla birlikte orijinal olarak akuple gelmektedir. Servo sürücülerde geribesleme elemanı bağlamak için hazır soket bulunmaktadır. Servo motor genel bağlantıları Şekil 2.3'de gösterilmiştir.

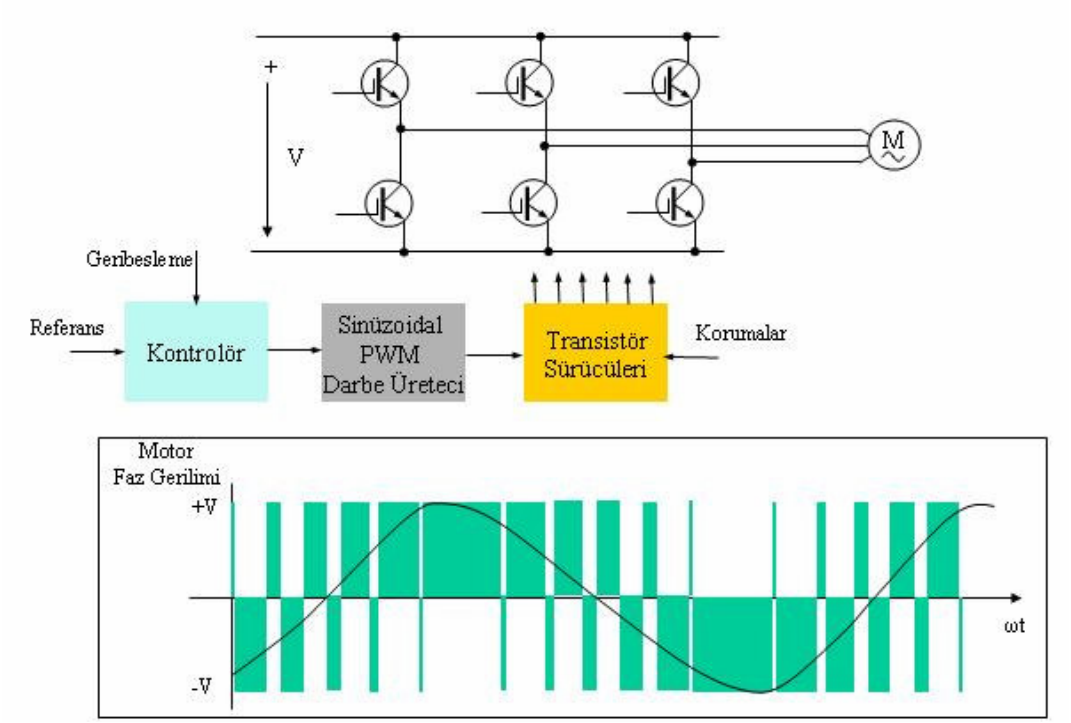


Şekil 2.3 Servo motor genel bağlantıları (WEB_3, 2006)

2.2. Servo Sürücüler

2.2.1. Servo Sürücü Genel Özellikleri

Motor, aktarma organı ve yükten oluşan mekanik servo sisteminin *hız*, *moment* veya *pozisyon* değişkenlerinden herhangi birinin, bu değişkenle ilgili verilen referans değerine uygun olarak hareket ettirilmesini sağlayan elektronik güç elemanlarıdır. Sinüzoidal darbe genişlik modülasyonu ile çalışan, analog veya dijital yapıda sürücülerdir. Geribesleme olarak hall sensör, resolver, artımlı encoder veya mutlak (sin/cos) encoder kullanılır. Dinamik performansı yüksek, kullanımı bilgi gerektiren görece pahalı sürücülerdir.



Şekil 2.4 Servo sürücü çalışma prensibi (WEB_4, 2006)

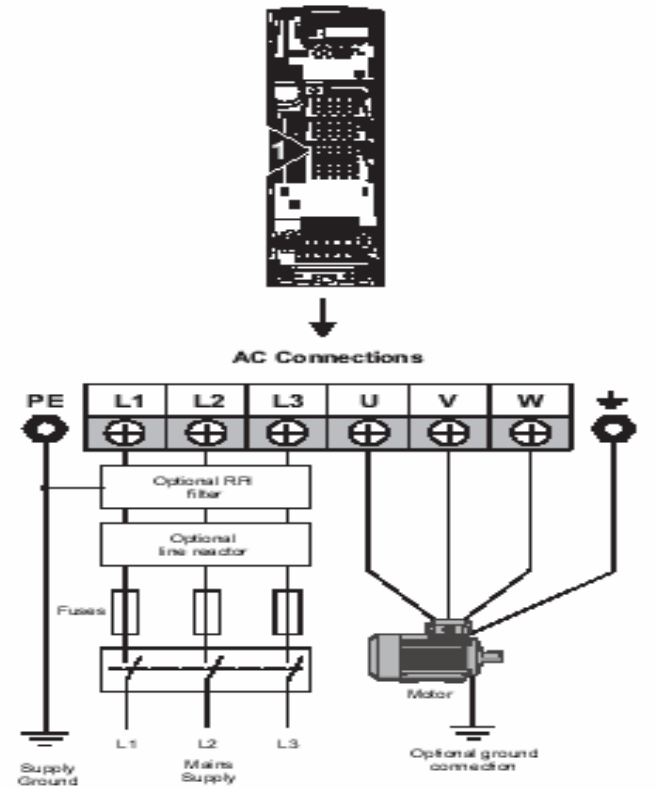
Şekil 2.4'de gösterildiği gibi, servo sürücüler kompakt bir yapıya sahip olup donanımında mikroişlemci bulunan bir kontrolör, darbe genişlik modülasyonu yaparak sinüzoidal çıkış gerilimini oluşturan bir PWM darbe üreticisi ve güç katını oluşturan bir transistorlü sürücü katı vardır.

İçerdiği mikroişlemci sayesinde sürücüler, kullanıcı dostu yani kolay kullanımlıdır. Ayrıca aşağıda bir kısmı verilmiş bir çok arızada otomatik motor koruma özelliği vardır.

- Aşırı akım
- Çıkış faz ve toprak kısa devresi
- DC bara yüksek ve düşük gerilim koruması
- Aşırı hız
- Giriş gerilimi düşük koruması
- Yarıiletken aşırı sıcaklık koruması
- Motor aşırı sıcaklık koruması
- Geribesleme hatası
- Güç kaynağı hatası
- Frenleme direnci hatası

2.2.2. Servo Sürücü Bağlantıları

2.2.2.1 Servo Sürücü Güç Bağlantıları

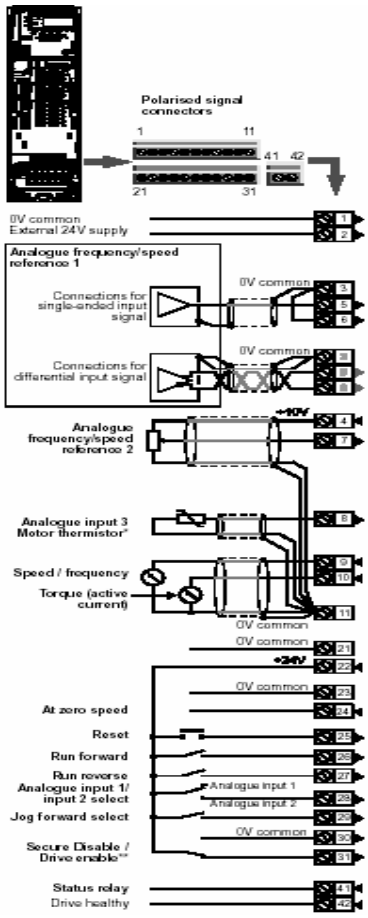


Şekil 2.5 Servo sürücü güç bağlantıları (WEB_5, 2006)

Şekil 2.5'de gösterildiği üzere L1, L2, L3 sürücünün üç faz enerji beslemesidir. U, V, W uçları ise servo motor bağlantı uçlarıdır. Burada dikkat edilmesi gereken hususlardan birincisi motor kablosundaki U, V, W uçlarının sürücüdeki U, V, W bağlantı klemensine sıralı şekilde bağlanmasıdır. İkinci husus ise sürücü L1, L2, L3 enerji bağlantısının mutlaka uygun bir hızlı sigorta ile sürücüye bağlanmasıdır. Hızlı sigorta karakteristik değerler olarak normal sigorta eğrilerinden daha hızlı cevap verebilmekte ve sürücünün zarar görmesini engellemektedir.

2.2.2.2. Servo Sürücü Kontrol Bağlantıları

Servo sürücü, kontrol bağlantıları, sürücü ve motoru çalıştırıp durduran, hız ayarlarını yapan, arıza bilgilerini, devir, akım gibi bilgileri çıkış sinyali olarak verebilen hazır klemens bağlantılarıdır.



Şekil 2.6 Servo sürücü kontrol bağlantıları (WEB_6, 2006)

Şekil 2.6'da sürücü kontrol bağlantıları gösterilmiştir. Bu çalışmada kullanılan klemens numaraları ve fonksiyonları şu şekildedir:

- 3 ile 5 nolu klemens: Eksen kontrol kartından gelen hız referansı
- 26 ile 31 nolu klemens: Sürücü ve motor ileri çalış
- 30 ile 31 nolu klemens: Acil stop durdurma, sürücü aktif veya aktif değil

2.2.2.3. Servo Sürücü Geribesleme (Enkoder) Bağlantıları

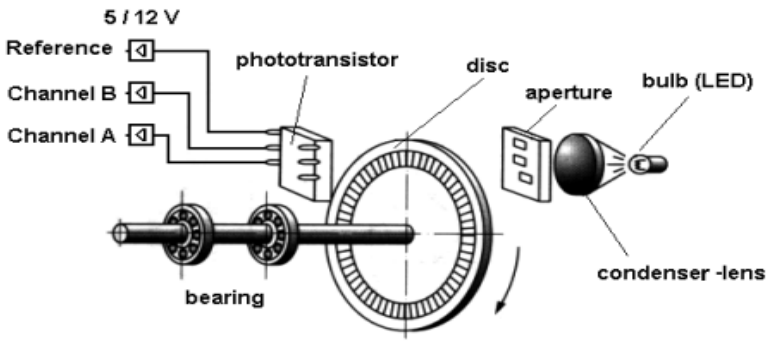
Servo motorlu sistemlerde geribesleme elemanı, motorun gerçek pozisyon bilgisini sürücüye ve kontrol sistemine aktarır. Tablo 2.1'de bir enkoderin, sürücü terminallerine bağlantı bilgileri verilmiştir.

<table>
<tr><td rowspan="2">Terminal</td><td colspan="12">Setting of Pr 3.38</td></tr>
<tr><td>Ab
(0)</td><td>Fd
(1)</td><td>Fr
(2)</td><td>Ab.SErVO
(3)</td><td>Fd.SErVO
(4)</td><td>Fr.SErVO
(5)</td><td>SC
(6)</td><td>SC.HiPEr
(7)</td><td>EndAt
(8)</td><td>SC.EndAt
(9)</td><td>SSI
(10)</td><td>SC.SSI
(11)</td></tr>
<tr><td>1</td><td>A</td><td>F</td><td>F</td><td>A</td><td>F</td><td>F</td><td colspan="2">Cos</td><td rowspan="4"></td><td>Cos</td><td rowspan="4"></td><td>Cos</td></tr>
<tr><td>2</td><td>A\</td><td>F\</td><td>F\</td><td>A\</td><td>F\</td><td>F\</td><td colspan="2">Cosref</td><td>Cosref</td><td>Cosref</td></tr>
<tr><td>3</td><td>B</td><td>D</td><td>R</td><td>B</td><td>D</td><td>R</td><td colspan="2">Sin</td><td>Sin</td><td>Sin</td></tr>
<tr><td>4</td><td>B\</td><td>D\</td><td>R\</td><td>B\</td><td>D\</td><td>R\</td><td colspan="2">Sinref</td><td>Sinref</td><td>Sinref</td></tr>
<tr><td>5</td><td colspan="6">Z*</td><td rowspan="2"></td><td colspan="6">Encoder input - Data (input/output)</td></tr>
<tr><td>6</td><td colspan="6">Z*</td><td colspan="6">Encoder input - Data\ (input/output)</td></tr>
<tr><td>7</td><td colspan="3">Simulated encoder
Aout, Fout**</td><td colspan="3">U</td><td colspan="6">Simulated encoder
Aout, Fout**</td></tr>
<tr><td>8</td><td colspan="3">Simulated encoder
Aout\, Fout**</td><td colspan="3">U\</td><td colspan="6">Simulated encoder
Aout\, Fout**</td></tr>
<tr><td>9</td><td colspan="3">Simulated encoder
Bout, Dout**</td><td colspan="3">V</td><td colspan="6">Simulated encoder
Bout, Dout**</td></tr>
<tr><td>10</td><td colspan="3">Simulated encoder
Bout\, Dout**</td><td colspan="3">V\</td><td colspan="6">Simulated encoder
Bout\, Dout**</td></tr>
<tr><td>11</td><td colspan="3" rowspan="2"></td><td colspan="3">W</td><td colspan="2" rowspan="2"></td><td colspan="4">Encoder input - Clock (output)</td></tr>
<tr><td>12</td><td colspan="3">W\</td><td colspan="4">Encoder input - Clock\ (output)</td></tr>
<tr><td>13</td><td colspan="12">+V***</td></tr>
<tr><td>14</td><td colspan="12">0V common</td></tr>
<tr><td>15</td><td colspan="12">th****</td></tr>
</table>

Tablo 2.1 Enkoder geribesleme elemanı bağlantı terminalleri (WEB_7, 2006)

2.3. Geribesleme Elemanları

Bu çalışmada geribesleme elemanı olarak enkoder kullanılmıştır. Motor miline takılan bir enkoder, (servo motorlarda, enkoder motora akuple gelmektedir) motorun bir turunu kendi pals sayısı kadar çözünürlüğe ayırmaktadır. Enkoder çalışma prensibi Şekil 2.7'de verilmiştir.



Şekil 2.7 Enkoder çalışma prensibi (WEB_8, 2006)

2.4. Servo Sürücülerin Devreye Alınması

Sistem bağlantıları yapıldıktan sonra, enerji açılıp, servo sürücülerin parametre ayarları yapılmalıdır. Bu ayarlar; motor ve geribesleme elemanının parametrelerinin tanıtılmasını, sisteme ait maksimum hız, maksimum akım, hız kontrol sisteminin PID katsayılarının ayarlanmasını kapsar. Sürücüler çok geniş uygulamaları kapsayacak şekilde dizayn edildiği için yaklaşık 20 menü ve her menüde 50 parametre olmak üzere 1000 den fazla parametre ayarı bulunmaktadır. Parametre numaraları, menü ve parametre bilgisini içermektedir. Örneğin 0.38 nolu parametre; 0. menünün 38 nolu parametresini göstermektedir. Kullanıcı ihtiyaca göre bu parametre menülerini ve parametreleri ayarlamak durumundadır. Bu menü ve parametreler sürücü ile birlikte gelen kullanıcı klavuzunda ve ayrıca CD formatında bulunmaktadır. Bu prototipte Control Techniques servo sürücüler kullanılmıştır.

Bu çalışmada servo modu ve klemensten çalış, dur, hızlan modu kullanılmıştır. Bu mod için şu ayarlar yapılmıştır.

- **Parametre 3.38: Ab.Servo** Geribesleme elemanı tipi servo enkoder seçilmiştir.
- **Parametre 3.36: 5V** Enkoder besleme voltaji 5 V seçilmiştir. Bu değer motorun üzerindeki geribesleme elemanı üzerinden öğrenilir.
- **Parametre 3.34: 4096** Enkoder pals sayısı. Bu değer motorun üzerindeki geribesleme elemanı üzerinden öğrenilir. Enkoderin 1 tur döndüğünde ürettiği pals sayısıdır.

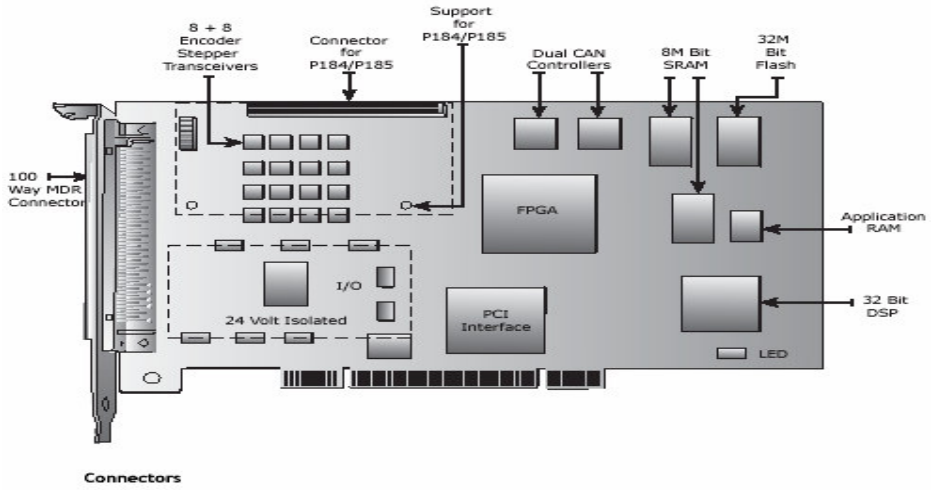
- **Parametre 0.46: 1.6 A** Motor akım değeri. Bu değer motor plaka etiketinden bakılır.
- **Parametre 0.42: 6** Motor kutup sayısı. Bu değer motor plaka etiketinden bakılır.
- **Parametre 0.02: 3000 rpm** Maksimum hız (devir/dk). Sisteme göre seçilir.
- **Parametre 0.03: 1 s** Hızlanma rampası (sn). Maksimum hıza çıkılacak süre. Sisteme göre seçilir.
- **Parametre 0.04: 1 s** Yavaşlama rampası (sn). Maksimum hızdan sıfır hıza düşülecek süre. Sisteme göre seçilir.

  Aşağıda verilen parametrelerden de gerçeklenen değerler izlenebilir:

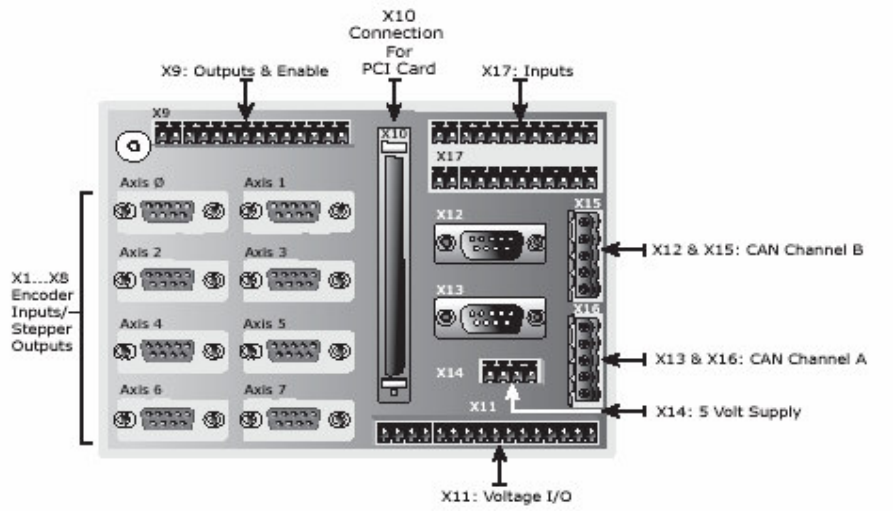
- **Parametre 0.10:** Motor devri (devir/dk)
- **Parametre 0.13:** Motor akımı A (amper)
- **Parametre 3.27:** Enkoder pozisyonu

2.5. Eksen Kontrol Kartı

Eksen kontrol kartları (Şekil 2.8), bilgisayarların PCI slotlarına takılan, kendi hızlı işlemcisi ve giriş çıkış donanımları bulunan DSP'li kontrol kartlarıdır. En büyük özellikleri 8 eksen için 8 enkoder bağlantısının direkt olarak yapılabilmesi, bu kartların kendi hafızalarının bulunması ve interpolasyon yapabilmeleridir.



Şekil 2.8 Eksen kontrol kartı genel görünümü (WEB_9, 2006)



Şekil 2.9 Eksen kontrol kartı terminal kartı (WEB_10, 2006)

Şekil 2.9'da gösterilen terminal kartı, bilgisayarın PCI slotuna takılan eksen kontrol kartının diğer donanımlarla bağlantısını sağlar. Bu çalışmada kullanılan bağlantılar şu şekildedir;

Axis 0 dan Axis 7 ye kadar olan konnektörler: Motorların enkoder bağlantılarının yapıldığı noktalardır. Şekil 2.10'da enkoder bağlantısı ayrıntıları verilmiştir.

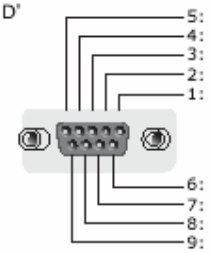
X17 konnektörü: Çalış, dur gibi veya sınır switchleri gibi dijital girişlerin bağlantılarının yapıldığı noktalardır.

X9 konnektörü: Sürücü ileri çalış, geri çalış gibi dijital çıkışların bağlantılarının yapıldığı noktalardır.

X11 konnektörü: Sürücü hız referanslarının verildiği anolog çıkışların bağlantılarının yapıldığı noktalardır.

Encoder/Stepper Connectors:

Axis Ø...Axis 7
Female 'D'



Pin	Servo Axis	Stepper Axis
1	Enc. A	Step +
2	Enc. /A	Step -
3	Enc. B	Direction +
4	Enc. /B	Direction -
5	GND	GND
6	Enc. Z	Boost +
7	Enc. /Z	Boost -
8	5v	5v
9	Not Connected	Not Connected
shell	Screen	Screen

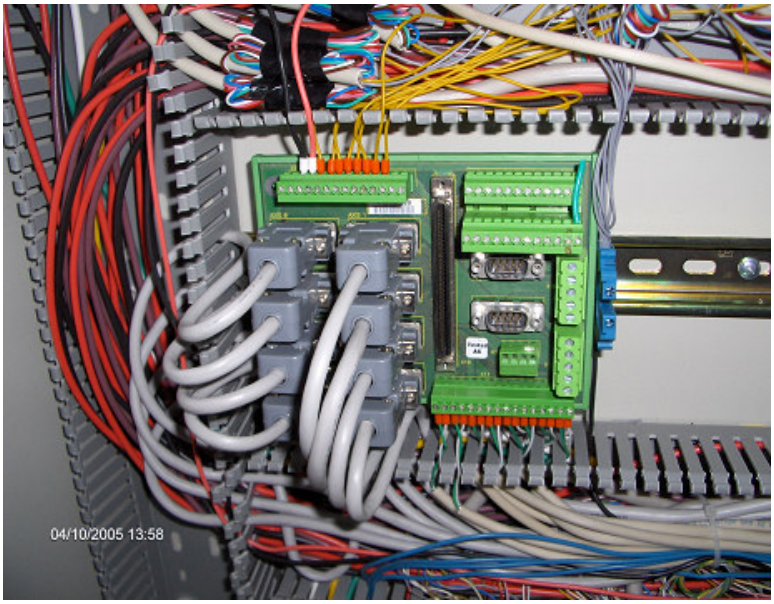
Şekil 2.10 Eksen kontrol kartı enkoder bağlantıları (WEB_11, 2006)

2.6. Sekiz Eksen Robot Kontrol Panosu

Bu çalışmada, Makine Mühendisliği Bölümü'nde Doç. Dr. E. Şahin ÇONKUR yönetimindeki TÜBİTAK destekli proje (Şekil 2.16) için sekiz ekseni kontrol edebilen servo sistemi ve elektrik panosu kurulmuş ve devreye alınmıştır (Şekil 2.11). Kullanılan motorlar Şekil 2.12'de gösterilmiştir. Enkoderler eksen kartı terminal bağlantılarına Şekil 2.13'te gösterildiği gibi bağlanmıştır. Şekil 2.14'te servo sürücüler, Şekil 2.15'te servo motorlar gösterilmiştir.

Şekil 2.11 Sekiz eksen robot kolu için yapılan elektrik panosu ve servo sistem

Şekil 2.12 Sekiz eksen robot kolu için kullanılan motorlar



Şekil 2.13 Eksen kontrol kartı enkoder bağlantıları, hız referans bağlantıları, çalış dur bağlantıları

Şekil 2.14 Servo sürücüler

Şekil 2.15 Servo motor ve kablo bağlantıları

Şekil 2.16 TÜBİTAK projesi için yapılan panonun kontrol ettiği robot kolu

3. ROBOT KOLU KİNEMATİĞİ

3.1. Robot Kolu Kinematiği Genel Özellikleri

Robotikte kinematik, hareket inceleme bilimidir. Robot kolu uzuvları referans koordinat çerçevesine göre dönebilir veya ötelenebilir. Denavit ve Hartenberg tarafından geliştirilen sistematik ve genel bir yaklaşım robotun uç noktası ile robot kolu uzuvlarının toplam yer değiştirmeleri arasındaki ilişkiyi kurar (Fu 1987). Uzuvlar arasındaki açısal ve doğrusal yer değiştirmeler mafsal koordinatları olarak adlandırılır ve uzuv değişkenleri tarafından tanımlanır. Uç noktasının referans koordinat sistemine göre dönme ve öteleme miktarını belirlemek için, her uzuv dönme ve öteleme miktarlarını gösteren A matrisleri sırayla birbiriyle çarpılır. Uç noktasının koordinatlarının verilmesi durumunda, geriye doğru gidilerek uzuv değişkenleri elde edilebilir. Bu işlemler *ileri* ve *ters* kinematik olarak isimlendirilir. Bundan sonraki kısımda ileri ve ters kinematiğin nasıl belirleneceği anlatılacaktır. Genel transformasyon matrisi basit robotlar için bile oldukça karışık olabilmektedir. Standart robotlar için olan Jacobian matrisi (McKerrow 1991) ve (Koivo 1989) gibi standart ders kitaplarında bulunabilir.

3.2. Genel Bir Robot Kolu İçin Koordinat Çerçeveleri ve Transformasyon Matrisleri

n boyutlu bir pozisyon vektörünün *n*+1 boyutlu bir vektör ile gösterilmesine homojen koordinat gösterimi denir. Aşağıda koordinat çerçeveleri arasında bir pozisyon vektörünü homojen koordinatlarda gösteren 4×4'lük bir matris görülmektedir (Fu, 1987).

$$ {}^{R}\mathbf{T}_{H} = \left[\begin{array}{c|c} \begin{matrix} dönme \\ (3*3) \end{matrix} & \begin{matrix} öteleme \\ (3*1) \end{matrix} \\ \hline \begin{matrix} 0 \\ (1*3) \end{matrix} & \begin{matrix} 1 \\ (1*1) \end{matrix} \end{array}\right] = \begin{bmatrix} x_x & y_x & z_x & p_x \\ x_y & y_y & z_y & p_y \\ x_z & y_z & z_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$\mathbf{p} = p_x\mathbf{i} + p_y\mathbf{j} + p_z\mathbf{k}$ yeni çerçevenin orijininin vektörünü,

$\mathbf{x} = x_x\mathbf{i} + x_y\mathbf{j} + x_z\mathbf{k}$ yeni çerçevenin *x* ekseninin doğrultu vektörünü,

$\mathbf{y} = y_x\mathbf{i} + y_y\mathbf{j} + y_z\mathbf{k}$ yeni çerçevenin *y* ekseninin doğrultu vektörünü,

$\mathbf{z} = z_x\mathbf{i} + z_y\mathbf{j} + z_z\mathbf{k}$ yeni çerçevenin *z* ekseninin doğrultu vektörünü temsil eder.

Transformasyon matrisinin 4. kolonu *x, y,* ve *z* doğrultularındaki ötelemeye karşılık gelen 3 elemana sahiptir.

$$\mathbf{Trans}(p_x, p_y, p_z) = \begin{bmatrix} 1 & 0 & 0 & p_x \\ 0 & 1 & 0 & p_y \\ 0 & 0 & 1 & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

3 koordinat ekseninin herhangi birinde dönme mümkün olduğundan, *x, y* ve *z* eksenlerinde θ açısı kadar olan dönmelere karşılık gelen 3 dönme transformu vardır. *x* ekseni için aşağıdaki matris yazılabilir;

$$\mathbf{Rot}(x, \theta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) & 0 \\ 0 & \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

y ve *z* eksenleri etrafında sadece dönmeleri temsil eden matrisler benzer şekilde yazılabilir.

$$\mathbf{Rot}(y, \theta) = \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

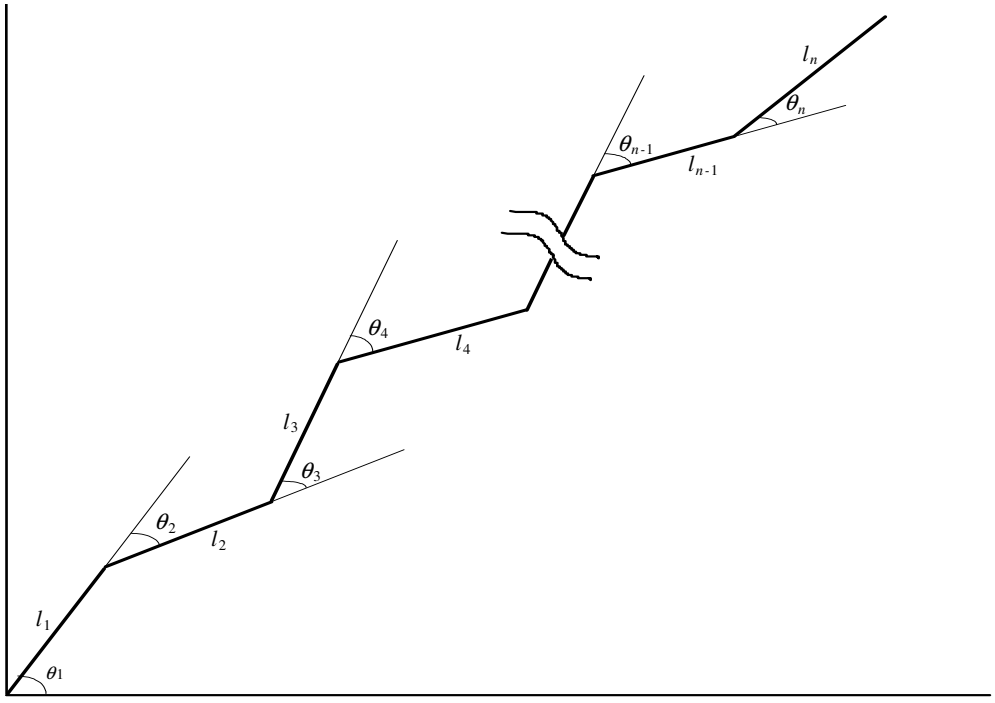
$$\mathbf{Rot}(z, \theta) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 & 0 \\ \sin(\theta) & \cos(\theta) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Transformasyon matrisinin elemanları saf dönme ve öteleme matrislerinin ard arda çarpımıyla bulunabilir. Kartezyen uzayda uç noktasının referans çerçevesine göre oryantasyonu istendiğinde, bu, sabit referans çerçevesinin eksenleri etrafındaki dönmelerin bir dizisi olarak elde edilebilir. Bunu yapmak için çok sayıda yol varsa da, en iyi bilinenlerde birisi "roll-pitch-yaw" transformasyonudur. 3 dönmenin belirlenmesiyle olur. Önce *x* ekseni etrafında dönme, sonra *y* ve daha sonra da *z* eksenli etrafında dönme.

$$\mathbf{RPY}(\phi,\theta,\Psi) = \mathbf{Rot}(z,\phi)\mathbf{Rot}(y,\theta)\mathbf{Rot}(x,\Psi)$$

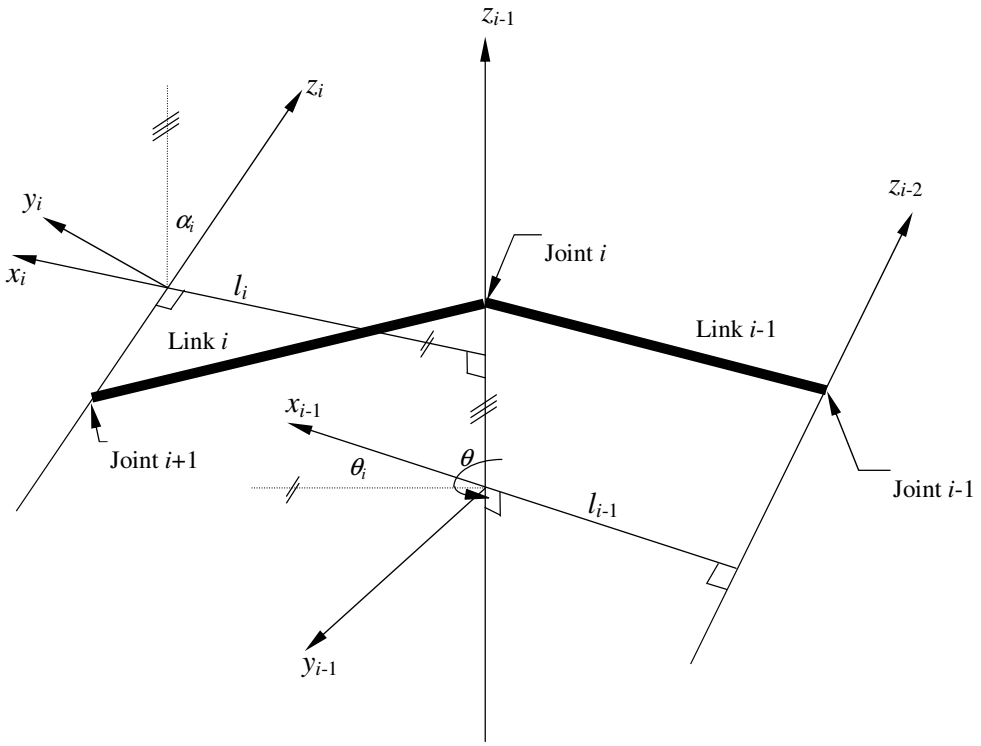
$$= \begin{bmatrix} C(\phi)C(\theta) & C(\phi)S(\theta)S(\Psi) - S(\phi)C(\Psi) & C(\phi)S(\theta)C(\Psi) + S(\phi)S(\Psi) & 0 \\ S(\phi)C(\theta) & S(\phi)S(\theta)S(\Psi) + C(\phi)C(\Psi) & S(\phi)S(\theta)C(\Psi) - C(\phi)S(\Psi) & 0 \\ -S(\theta) & C(\theta)S(\Psi) & C(\theta)C(\Psi) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Robotikte kinematik, hareket inceleme bilimidir. Robot kolu uzuvlarının pozisyonları, hızları ve ivmeleri arasındaki ilişkileri, kuvvetleri ve hareketi etkileyen diğer faktörleri dikkate almadan inceler.



Şekil 3.1 Bağıl açıları gösterilen *n* uzuvlu seri robot kolu

Şekil 3.1'de verilen seri uzuv robot kolunu incelendiğinde uzuvlardan birinin pozisyonu ve oryantasyonu veya her ikisi değiştiğinde, bu uzuv ve uç noktası arasında olan uzuvların pozisyonları ve oryantasyonları da değişecektir. Uzuvların pozisyon ve oryantasyonları değişirken, genellikle uç noktasının pozisyonu ve oryantasyonunun temel referans çerçevesine göre belirlenmesi arzu edilir.



Şekil 3.2 D & H parametreleri

Her hareketli uzuva bir koordinat çerçevesi bağlandığında, her iki uzuv arasındaki transformasyon, **A** matrisi olarak isimlendirilen bir homojen transformasyon matrisi ile tanımlanabilir. Örneğin 1. uzuv temel çerçeveye 1. **A** matrisi olan ${}^{0}\mathbf{A}_{1}$ ile bağlanır. Uç noktasının transformasyon matrisi ${}^{R}\mathbf{T}_{H}$ ise **A** matrislerinin 1. **A** matrisinden başlayarak uç noktasının **A** matrisine kadar olan **A** matrislerinin ard arda çarpımıyla referans çerçevesine göre ifade edilebilir.

$$ {}^{R}\mathbf{T}_{H} = {}^{R}\mathbf{T}_{1}.{}^{1}\mathbf{T}_{2} \cdot\cdot {}^{n-2}\mathbf{T}_{n-1}.{}^{n-1}\mathbf{T}_{H} = {}^{0}\mathbf{A}_{1}.{}^{1}\mathbf{A}_{2} \cdot\cdot {}^{n-2}\mathbf{A}_{n-1}.{}^{n-1}\mathbf{A}_{n} $$

Bu denklem kapalı formda olduğundan, herhangi bir terim bir diğerine göre ifade edilebilir ve robot kolunun kinematik analizinde çok önemlidir. Çünkü robot kolunun ileri ve ters kinematiğinin çözümlemesinde kullanılır.

3.3. D & H (Denavit ve Hartenberg) Koordinat Çerçeveleri

Uzuvlar arasındaki açılar ve yer değiştirmeler uzuv değişkenleri tarafından tanımlanan mafsal koordinatları olarak adlandırılır. Her uzuvu için bir DOF'a sahip olan *n* mafsallı bir seri uzuv robot kolu *n+1* tane uzuva sahiptir. D & H parametrelerini kullanarak Şekil 3.2'de görüldüğü üzere doğrusal olması şart olmayan herhangi bir uzuv, yapısal kinematik uzuv parametreleriyle karakterize edilebilir (Koivo, 1989);

l_n: uzuv uzunluğu, mafsal eksenleri arasındaki ortak normal boyunca olan mesafedir. z ve z_{i-1} eksenleri arasındaki uzunluğa diktir.

α_i: positif z_{i-1} ekseninden positif z_i eksenine olan pozitif x ekseni etrafındaki dönme açısıdır.

θ_i: pozitif x_{i-1} ekseninden positif z_i eksenine olan pozitif z_{i-1} ekseni etrafındaki dönme açısıdır.

d_i 1. koordinat çerçevesinin orijininden x_{i-1} ekseni boyunca olan z_{i-1} ve x_i eksenlerinin kesişimine olan uzaklıkdır.

Yapısal parametrelerin bazıları robot kolu hareketi sırasında zamanla değişebilir. Bu değerleri değişen parametreler, örneğin dönel mafsaldaki θ_1 açısı, mafsal değişkenleri olarak isimlendirilir.

3.4. İleri Kinematik

Bir robot kolunun mafsal uzayı tanımı, kartezyen uzayı tanımıyla ilişkilendirilebilir. Yani, verilen bir mafsal değişkenleri takımı için uç noktasının pozisyonu ve oryantasyonu kartezyen koordinatlarda belirlenebilir. Bu işlem ileri kinematik olarak bilinir. İleri kinematik, önceden bahsettiğimiz **A** matrislerini kullanarak uç noktasının pozisyon ve oryantasyonunu veren ${}^{R}\mathbf{T}_{H}$ transformasyon matrisini bulma olayıdır. ${}^{R}\mathbf{T}_{H}$ transformasyon matrisi, genel öteleme matrisi ile RPY açılarının oryantasyon transformu matrisiyle çarpılarak da elde edilebilir.

$$ {}^{R}\mathbf{T}_{H} = \begin{bmatrix} x_x & y_x & z_x & p_x \\ x_y & y_y & z_y & p_y \\ x_z & y_z & z_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & p_x \\ 0 & 1 & 0 & p_y \\ 0 & 0 & 1 & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}. $$

3.5. Ters Kinematik

Kartezyen koordinatlarda verilen bir uç noktası pozisyonu ve oryantasyonu için, mafsal değişkenlerinin alması gereken değerleri bulma işlemine ters kinematik denir. Ters kinematik ileri kinematiğe göre çözümlenmesi oldukça zor olabilmektedir. Birçok durumda çözümü garanti etmeyen ve deneme yanılma yöntemini de içeren teknikler kullanmak gerekmektedir

4. HAREKET KONTROL ALGORİTMASI VE İKİ KOLLU ROBOT KOLU TASARIMI

4.1. Hareket Algoritması Genel Özellikleri

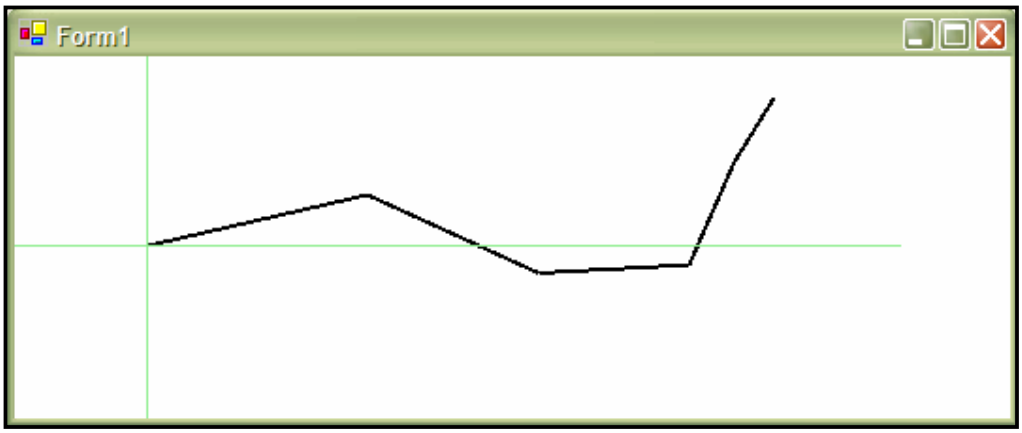
Hareket algoritması bir robot kolu için verilen hedefe ulaşmak için yapılacak hareketlerin uygun hız, uygun zaman, uygun pozisyon gibi kriterleri göz önünde bulundurarak planlanması, yazılımın oluşturulması, eksen kontrol kartı aracılığı ile servo sürücü ve servo motorlara aktarılmasıdır.

Yazılım ve algoritma geliştirme aşamaları dört ana gruba ayrılarak incelenecektir.

4.1.1. Nesneler ve Nesne Dizileri (Objects and Object Arrays)

Nesne temelli yaklaşım, geliştirdiğimiz programının her kısmında kullanılmaktadır. Microsoft Visual Studio.NET dillerinden olan Visual C++.NET ile geliştirilmiş "nesne temelli programlamayı" esas alan bir yapıya sahiptir. Robot uzuvları, robotun kendisi, engeller, robotun uç noktasının çizimindeki çizgi parçacıkları gibi birçok işlem nesneleri tanımlayan sınıflarla başarılır. Her nesne de kendisiyle ilgili *ArrayList* isimli bir nesne dizisinde saklanarak kullanılırlar ve birçok manipülasyona maruz bırakılabilirler.

Nesnelerin ve nesne dizilerinin programımızdaki öneminden dolayı geliştirilen basit bir programla nesneler ve nesne dizileri kısaca şu şekilde anlatılabilir: Bu program, çok sayıda uzva sahip bir seri robot kolu oluşturup, robot kolunun uzuv açılarına keyfi değerler vererek ekranda robot kolunun hareketini sağlamaktadır (Şekil 4.1).



Şekil 4.1 Beş uzuvlu robot kolu

Öncelikle aşağıdaki gibi bir uzuv (link) sınıfı yazılır:

```
public __gc class  link
{
      public:
      double l, teta;
};
```

Görüldüğü gibi "link" sınıfı kullanımı son derece basittir, sadece iki tane üyesi vardır. "l" uzuv uzunluğu değişkeni ve "teta" uzuv açısı değişkenidir. Bir nesne dizisi (ArrayList) oluşturulur ve bu dizinin adresini "ar" isimli ArrayList tipinde bir işaretçiye atanır.

```
ArrayList *ar;
ar = new Array List();
```

Daha sonra bir *link* nesnesi oluşturulur ve bu nesnenin adresini "le" isimli *link* tipindeki işaretçiye atayalım. Daha sonra, uzuv değişkenlerine keyfi değerler verelim:

```
link *le=new link();
le->l = 200;
le->teta = 0.2;
```

Sonra, *link* nesnesini nesne dizisine ekleyelim.

```
ar->Add(le);
```

Bu şekilde istenilen kadar *link* nesnesi oluşturulup *ArrayList'e* eklenebilir. *Link* ekleme bittiğinde artık robot kolu ekranda devamlı hareket edecek şekilde çizilmeye hazırdır.

Bunun için bir *timer* olayı kullanılabilir. *Timer'ın* içinde bir *link* nesnesi oluşturulur. *ArrayList*'te önceden kaydedilmiş *link* nesnesi bu *link*'e atanır. Böylece bu *link'in* üyelerine erişim mümkün hale gelir. Aşağıda gösterildiği gibi her *link* açıları *timer'*ın her *tick* olayında belli bir miktar arttırıldığında ve *paint* olayında çizimler yapıldığında robot ekranda hareket eder.

```
link *le=new link();
le = dynamic_cast < link* >(ar->Item[0]);
le->teta+=-0.01;
le= dynamic_cast <link* >(ar->Item[1]);
le->teta+=0.02;
```

Link açılarına farklı arttırım değerleri verilerek değişik hareketler elde edilmektedir. Ayrıca robotun uç noktasının hareketi çizildiğinde çok ilginç eğriler elde edilmektedir. Belli bir zaman sonra robotun ekranda çizdiği eğrilerin üzerinden giderek, hareketlerini tekrarladığı görülmektedir.

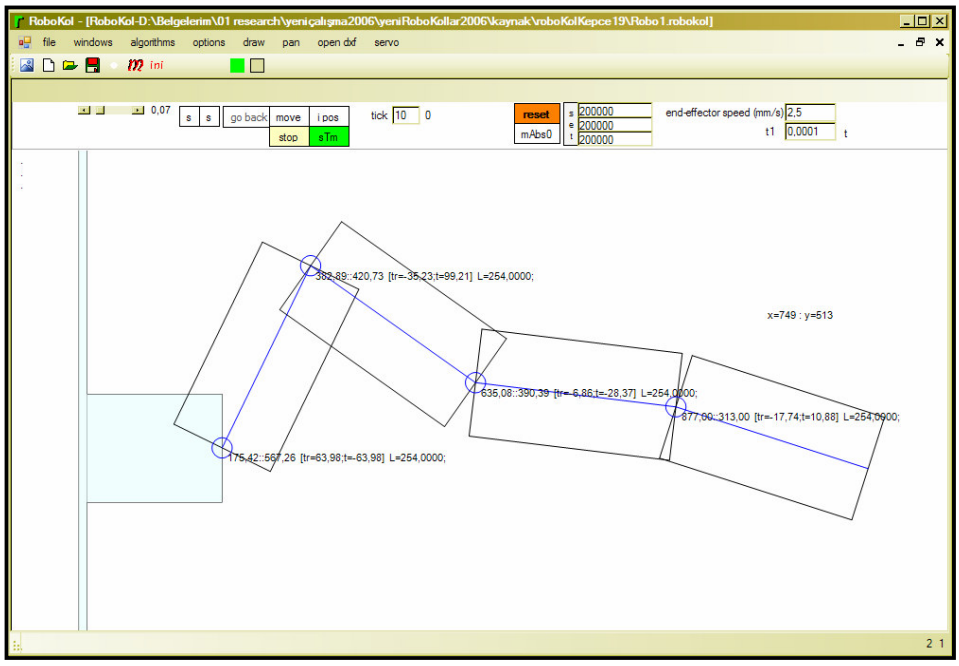
Eğer yukarıdaki sınıfları kullanmadan bu programı yazmaya çalışsaydık birçok ciddi zorlukla karşılaşacaktık. Örneğin her uzvun sadece iki değişkeni olmasına rağmen 5 uzuvlu robotumuza 10 tane değişken tanımlamamız gerekecekti. Uzuvlar için yeni bir değişken düşündüğümüzde her uzuv için ayrı ayrı değişiklik yapmamız gerekecekti. Robot kol sayısını arttırmaya kalktığımızda da kodun hemen her yanında zor olan değişiklikler yapmamız gerekecekti. Ayrıca döngüleri kullanarak kodu kısaltamayacaktık.

4.1.2. RoboKol Programının Temel Yapısı ve Sınıfları

RoboKol programı İngilizce dilinde yazılmıştır. Ayrıca fonksiyon ve değişken isimleri mümkün olduğunca yaptığı işe uygun olarak ve yeteri kadar uzunlukta isimlendirilmiştir.

Algoritma geliştirebilmek için yazılması gereken RoboKol programdaki temel kısımlar

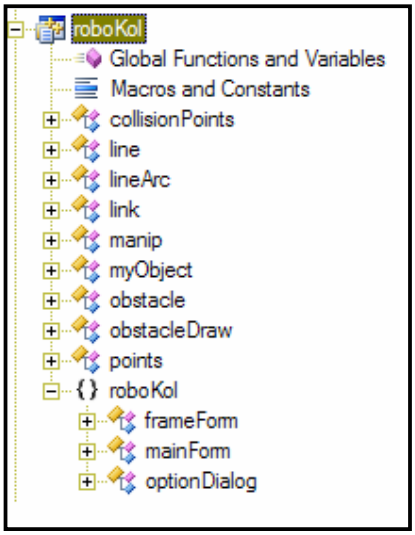
- Robotu temsil eden sınıfı,
- Engelleri temsil eden sınıfı,
- Engelleri çizecek kodu,
- Potansiyel alanı hesaplayıp ekranda görüntüleyecek kodu,
- Menüler, ikonlar, dosya açma vb. kısımları,
- Robotları ve engelleri içeren çalışma alanını kaydedecek kodu yazmaktır.



Şekil 4.2 *RoboKol* programının arayüzü

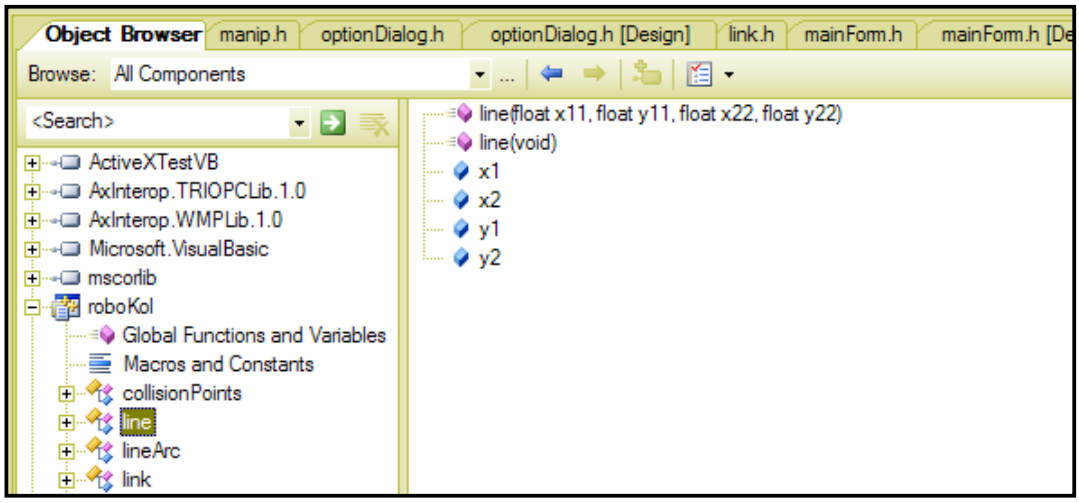
Şekil 4.2'de görülen *RoboKol* programının arayüzünde dört uzuvlu, uzuv genişlikleri ve uzuvların bazı parametreleri de görülen bir robot çizilmiştir. Algoritma geliştirirken uzuv genişlikleri ve parametre değerleri bazen gerekli olmakta bazen de bunların

ekranda görüntülenmesi istenmemektedir. Bu yüzden uzuv genişlikleri ve parametreleri değerleri "options" diyalog kutusundan gerektiğinde iptal edilebilir.



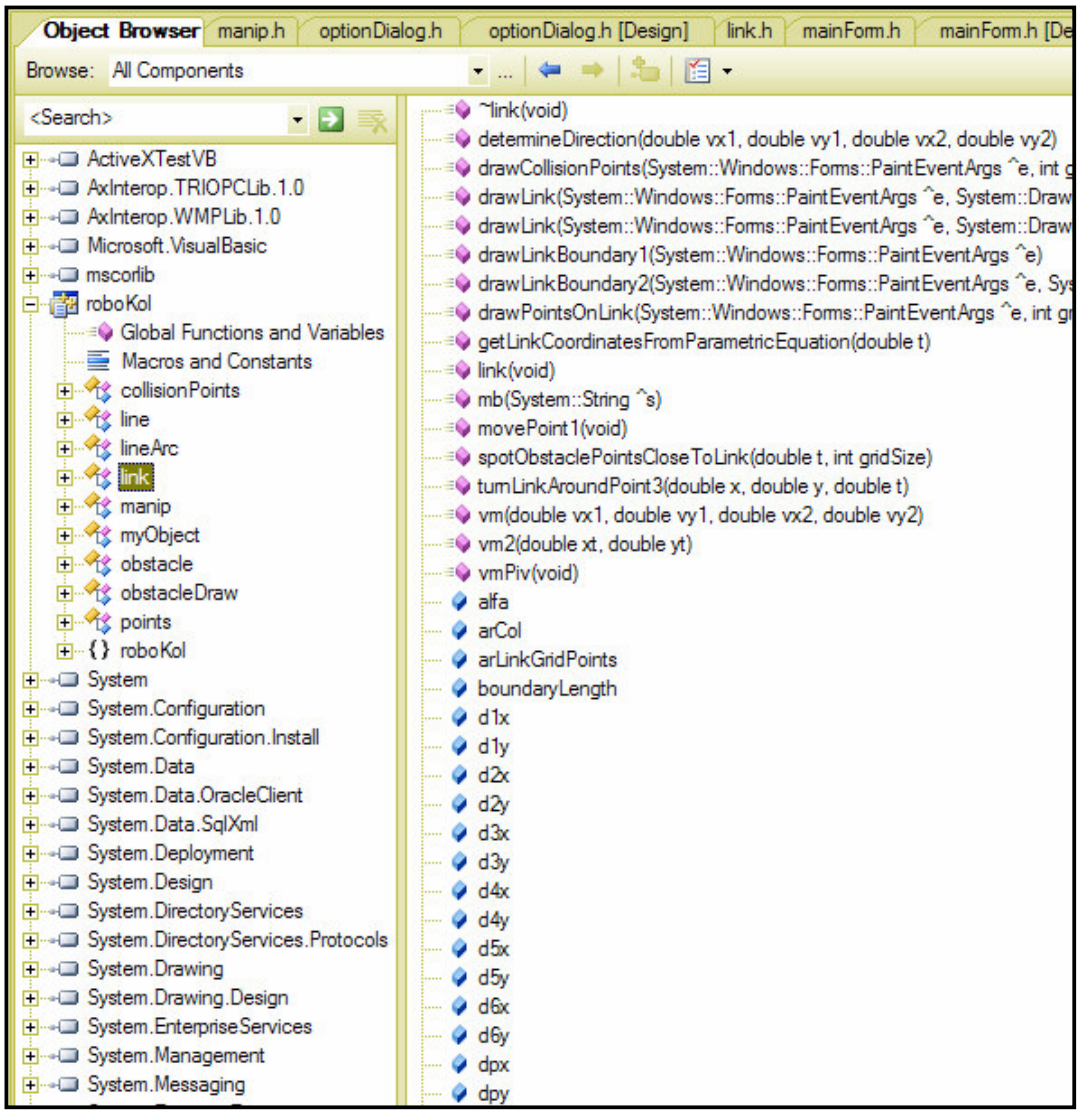
Şekil 4.3 *RoboKol* programının sınıfları

Şekil 4.3'te şu ana kadar yazılmış olan kodda bulunan sınıflar görülmektedir. Geliştirilmiş sınıflar aşağıda açıklanmıştır.



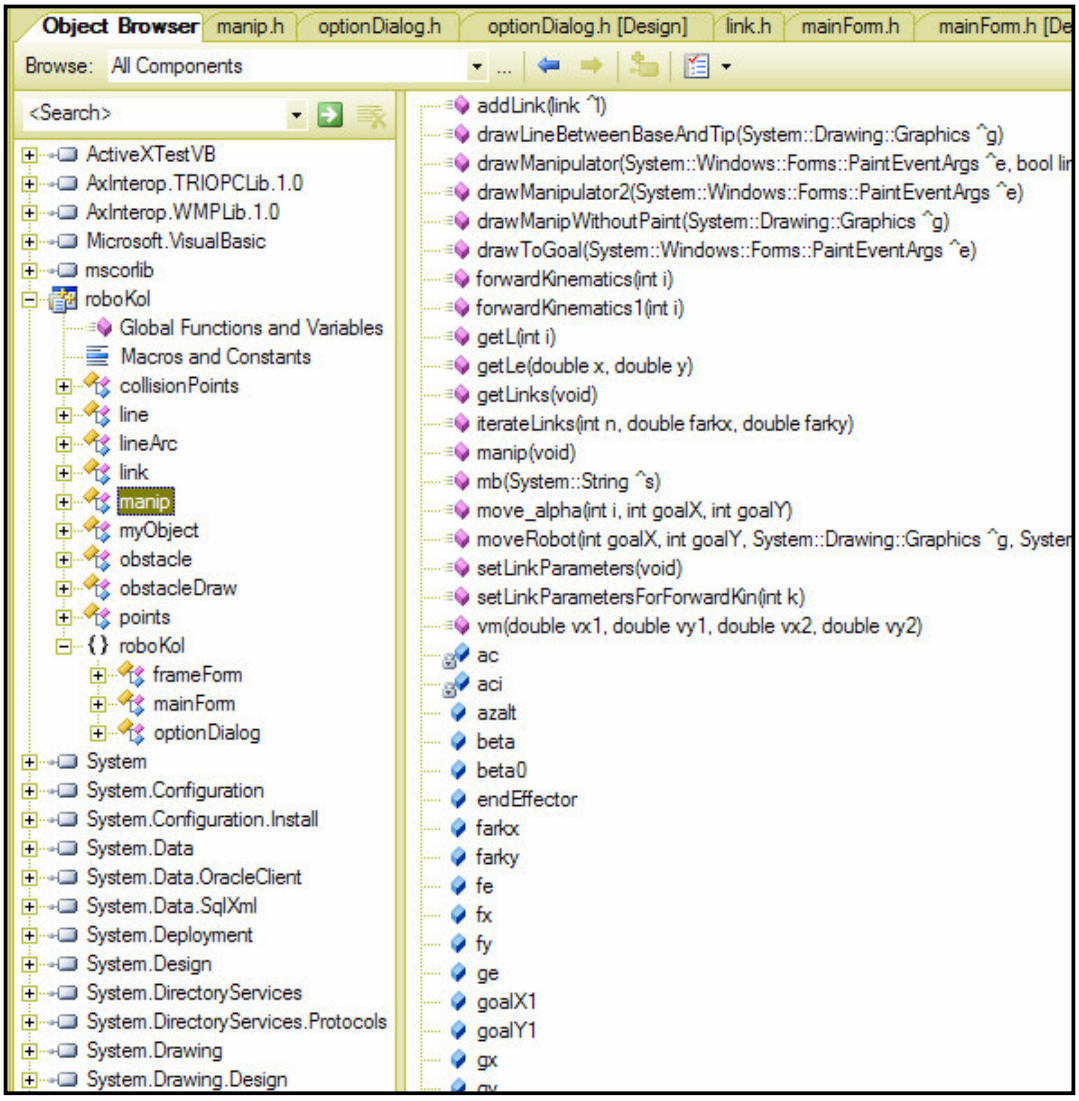
Şekil 4.4 *line* sınıfının değişkenleri ve fonksiyonları

Şekil 4.4'te görülen *line* sınıfı robotun uç noktasının gittiği yolu ekranda çizmek için kullanılır. Bu yol ard arda eklenmiş çizgilerden oluşur. Şekildeki dört değişken her çizgi için başlangıç ve bitiş koordinatlarını saklar. Bu koordinatlara erişmek için ise dört tane değişken alan "yapıcı" tasarlanmıştır.



Şekil 4.5 *link* sınıfının değişkenleri ve fonksiyonları

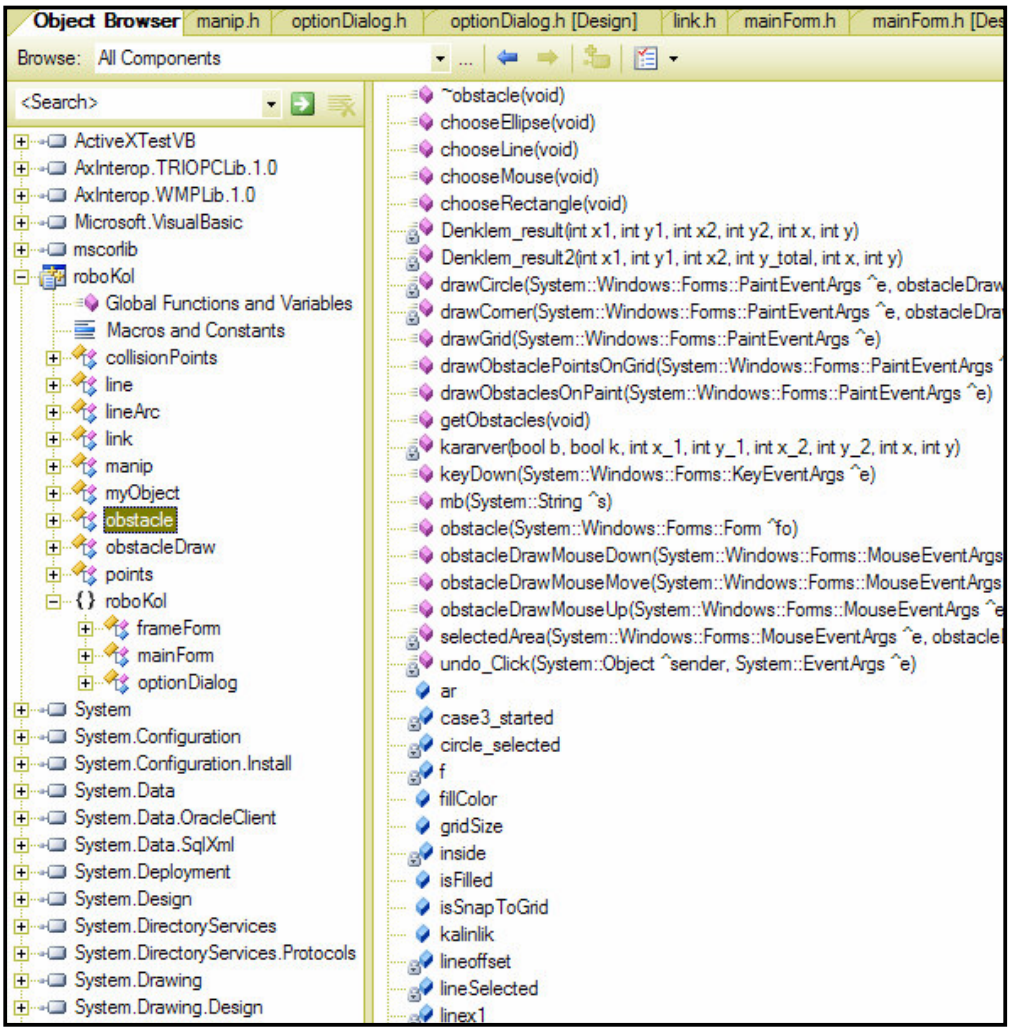
Şekil 4.5'te görülen *link* sınıfı uzuvlarla ilgili işlemleri bir araya getiren bir sınıftır. Her *link* ile ilgili başlangıç ve bitiş koordinatları, *link*'in rengi, *link*'in mutlak ve bağıl açıları gibi çok sayıda değişken bu sınıftan oluşturulan nesnelerce saklanır. Ayrıca *link*'in çizimini bu sınıfı kullanarak yapmak oldukça pratik olmaktadır. Bu sebepten Şekil 4.5'te görüldüğü gibi bu sınıf içinde *link* çizimiyle ilgili çok sayıda fonksiyon geliştirilmiştir.



Şekil 4.6 *manip* sınıfının değişkenleri ve fonksiyonları

Şekil 4.6'da görülen *manip* sınıfı robot kolu ile ilgili işlemleri gerçekleştirir. Çok sayıda fonksiyon ve değişken içerir. Bu sınıfı kullanarak aynı anda birden çok robot kolu oluşturulabilir. Böylece program daha esnek bir yapıya sahip olmaktadır. *Manip* sınıfı yukarıda bahsedilen *link* sınıfını kullanarak uzuvlarını oluşturur ve uzuvlar arasındaki ilişkileri kurar. Örneğin bir *link* hareket ettirildiğinde diğer linklerin koordinatları değişecektir. Bu yüzden *link* nesnelerinin üyelerine yeni uygun değerler atanmalıdır. Bunu *manip* sınıfı "forwardKinematics" fonksiyonuyla yapar. Ekranda robot kolunu çizmek için *manip* sınıfında "drawManipulator" fonksiyonu geliştirilmiştir. Esasında bu fonksiyonda *link*'leri çizecek kodun kendisi yoktur. Bu

fonksiyon sadece her *link* için *link* sınıfında bulunan ve esas çizimi yapan "drawLink" fonksiyonunu çağırır.



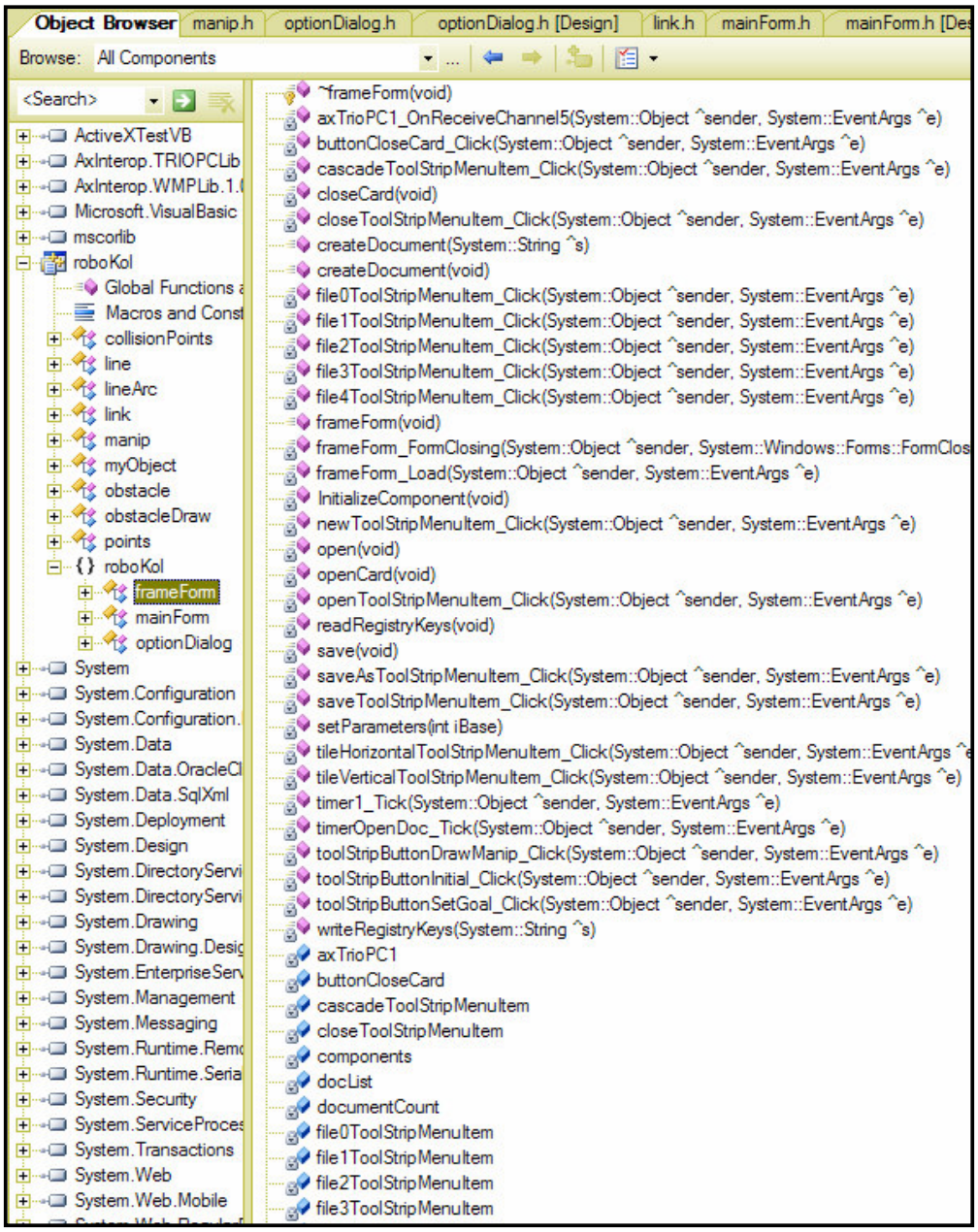
Şekil 4.7 *obstacle* sınıfının değişkenleri ve fonksiyonları

Şekil 4.7'de görülen *obstacle* sınıfı ayrı bir program olarak *RoboKol* programına entegre edilmiştir. Burada ortaya çıkan ilginç bir nokta, "mousemove" gibi olaylara yazılmış kodu adapte etmekte ortaya çıktı. Bir programda tek bir "mousemove olayı olduğundan diğer programda olan "mousemove" olayına yazılmış kodu kopyalayıp, bu program içine yazmak yerine, "mousemove" olayının ismini değiştirerek basit bir fonksiyon haline getirip esas programın "mousemove"ında kullanıldı. Böylece karışıklık önlenmiş oldu.

Obstacle sınıfı, dikdörtgen, daire, çizgi gibi nesneleri ekrana çizmekte ve gerektiğinde büyültme, küçültme, uzatma, kısaltma ve silme gibi edit işlemlerine izin

vermektedir. Burada ekrana bir dikdörtgen çizmekle onu fare ile değiştirmek için gerekli kodun oldukça farklı ve zor olduğunu vurgulamak gerekir.

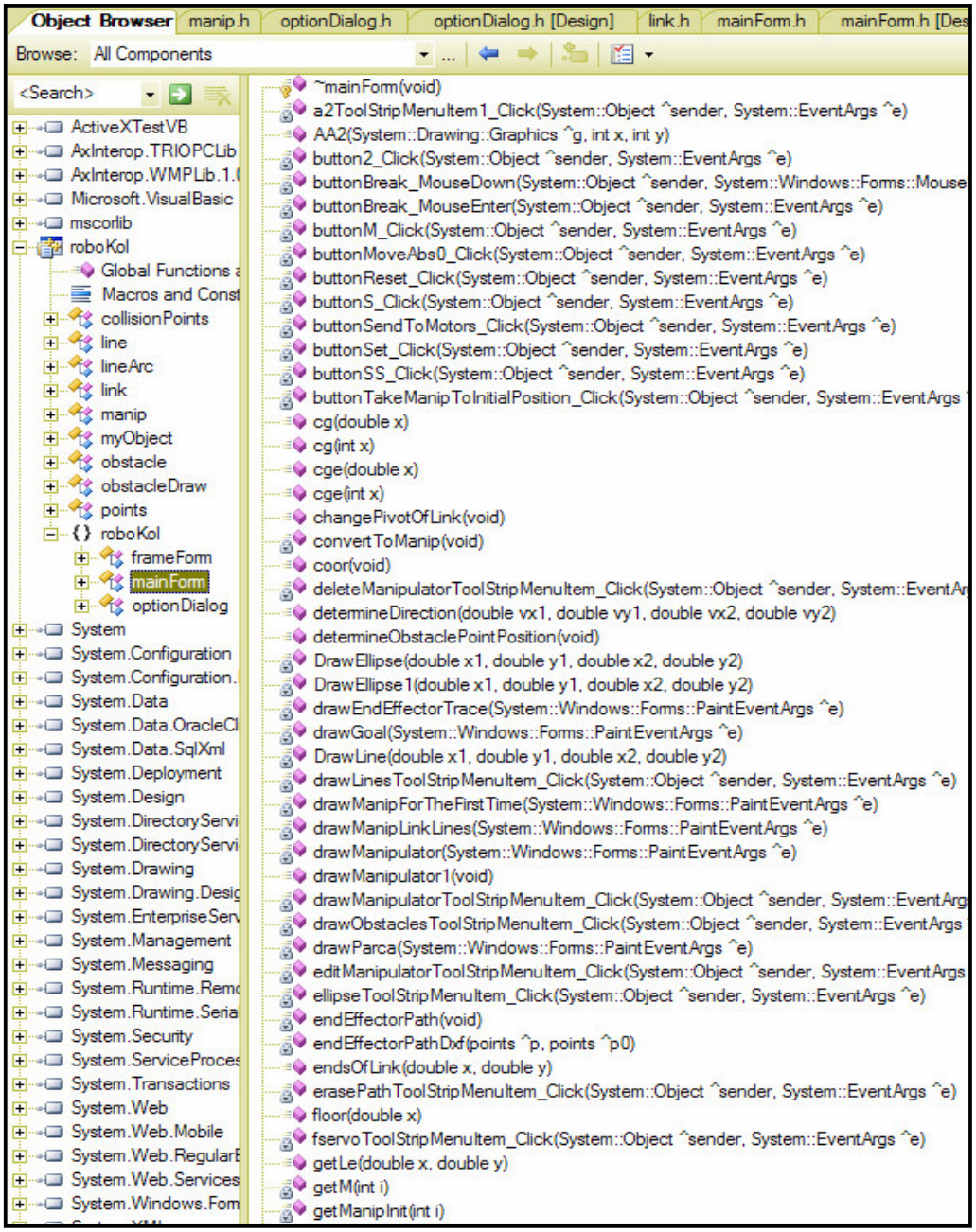
Bu sınıfın esas görevi ise ekrana çizdiği temel geometrik şekilleri, genişliği ve yüksekliği kullanıcı tarafından ızgarayı kullanarak "engel noktalarına" dönüştürmesidir. İstendiğinde ekrana bir ızgara çizebilir ve çizimleri bu ızgaraya uyumlu hale getirebilir.



Şekil 4.8 *frameForm* sınıfının değişkenleri ve fonksiyonları

RoboKol sınıfı MDI (Multiple Documet Interface) özelliğine sahip bir programdır. Yani program içinde aynı anda birden fazla doküman açılabilir (Şekil 4.11'de bu özellik görülebilir).

Şekil 4.8'de görülen *frameForm* sınıfının "parent" özelliği aktif hale getirilerek MDI oluşturulur. Bu sınıf kayıt yapmak, menüleri kullanmak, ikonları oluşturmak ve Windows kayıt defterine kayıt yapmak gibi işlemlerden sorumludur.



Şekil 4.9 *mainForm* sınıfının değişkenleri ve fonksiyonları

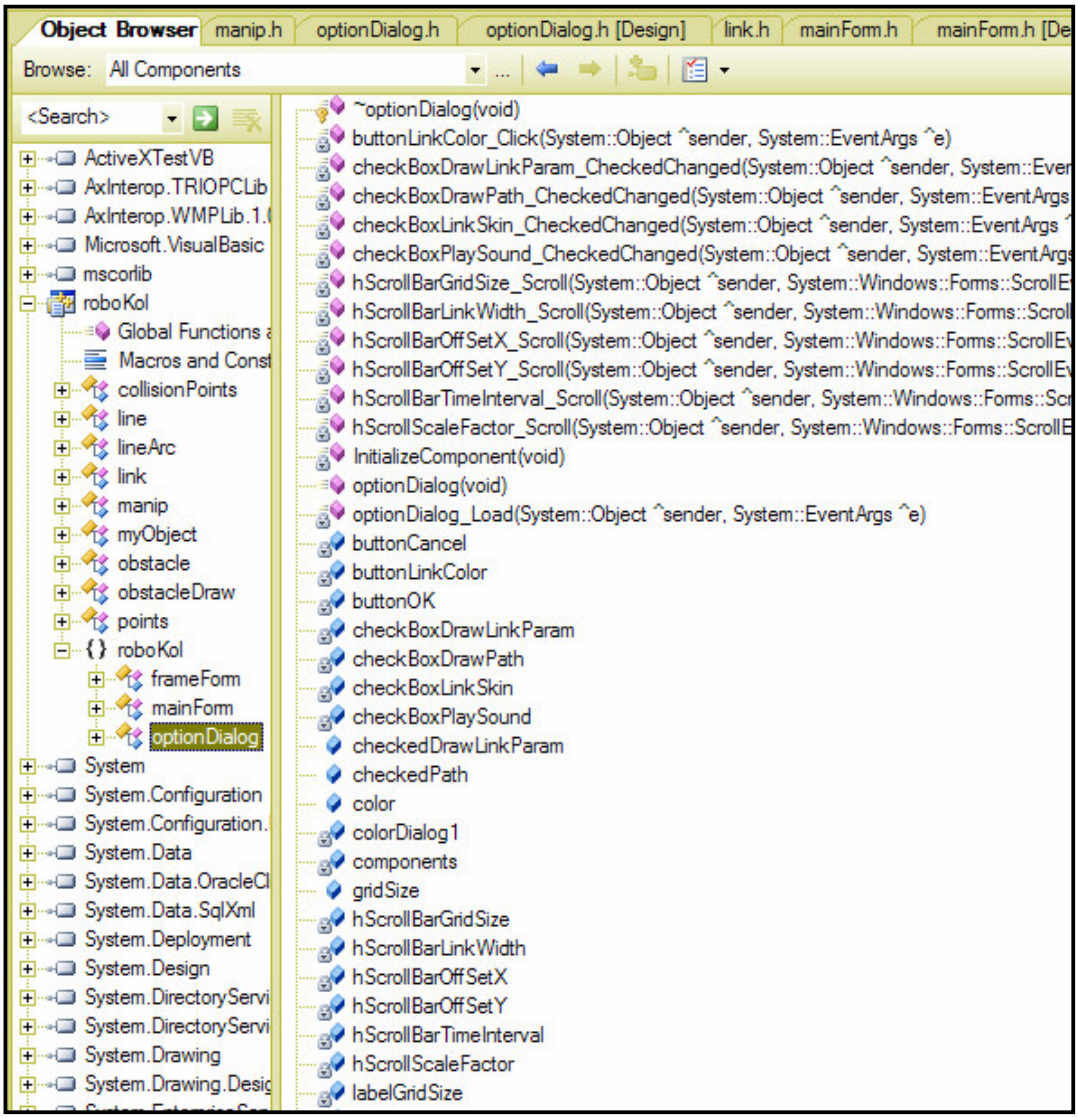
Şekil 4.9'da görülen *mainForm* sınıfı asıl işi gören sınıftır. Diğer sınıflarda tanımlanmış fonksiyonlar, o sınıflar vasıtasıyla çağırılır ve bu sınıf içinde kullanılır. Şekil 4.9'da da görüldüğü gibi çok sayıda fonksiyon ve değişken içermektedir.

Öncelikle ekrana bir robot kolu çizilir. Fakat çizim tamamlanmadan hemen önce bu çizim sadece uç uca eklenmiş doğrulardan başka bir şey değildir. Çizim tamamlanıp farenin ortadaki tuşuna basıldığında "convertToManip" fonksiyonu bu çizgileri robot koluna çevirir. Ayrıca, gerektiğinde robot kolunun ilk haline dönebilmek için robot kolunun ilk durumu ayrı bir *ArrayList*'de saklanır.

Program mümkün olduğunca değişik denemelere izin verecek şekilde esnek tasarlanmıştır. Birden çok robot kolu aynı çalışma alanında oluşturulabilir. Bu robot kolları temel (ilk) uzuvlarındaki mafsala tıklanarak istenilen bir yere sürüklenebilir. Diğer mafsallara fare ile tıklanarak uzuvlar elle oynatılarak düz kinematik denenebilir ve uzuv uzunlukları fare ile değiştirilebilir.

Bunlara ek olarak, bütün çalışma alanı fare ile sağa-sola ve yukarı-aşağı serbestçe kaydırılabilir, küçültülebilir ve büyültülebilir.

Diğer bir yazması zor olan kısım da koordinat sistemini değiştirmek oldu. Bilindiği gibi Windows'un varsayılan koordinat sisteminin orijini formun sol üst köşesidir. *x* ekseni sağa doğru, *y* ekseni ise aşağı doğru pozitiftir. Bu sistem her zaman kullanılan orijini sol alt köşede istenilen bir yerde olan, *x* ekseni sağa doğru, *y* ekseni ise yukarı doğru pozitif olan sisteme dönüştürüldü. Şekil 4.9'daki fonksiyonlar incelendiğinde, bu fonksiyonlar arasında servo motorlarla ilgili fonksiyonların olduğu da görülecektir.

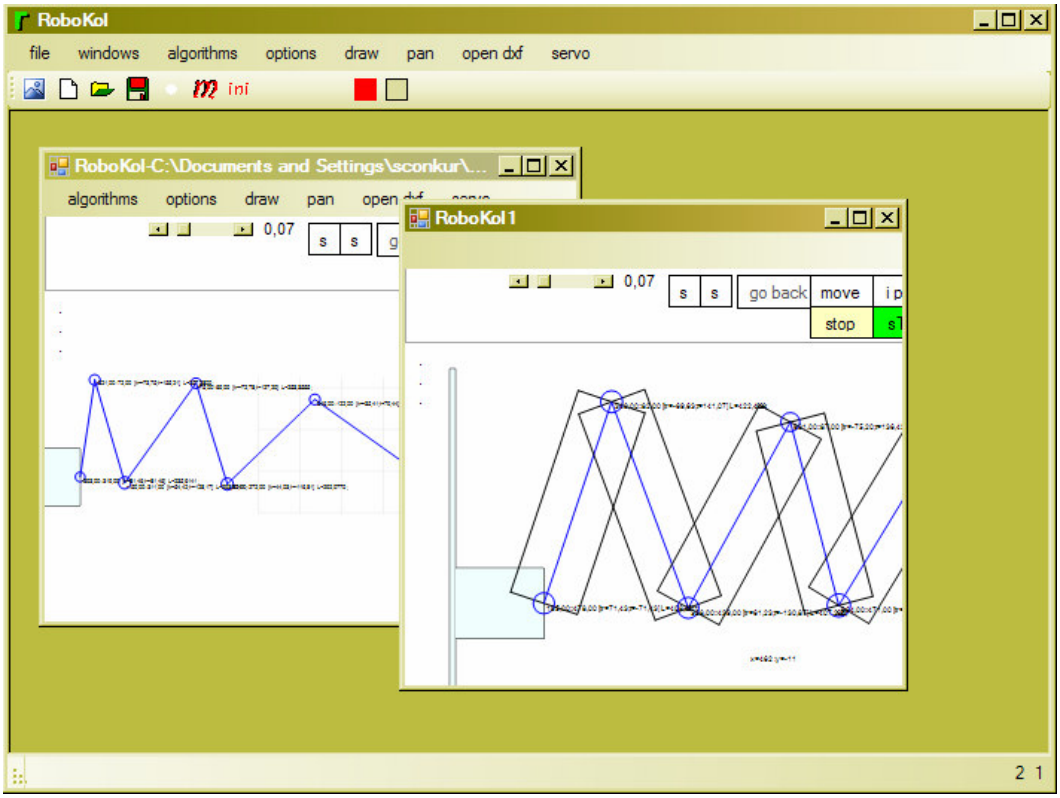


Şekil 4.10 *optionDialog* sınıfının değişkenleri ve fonksiyonları

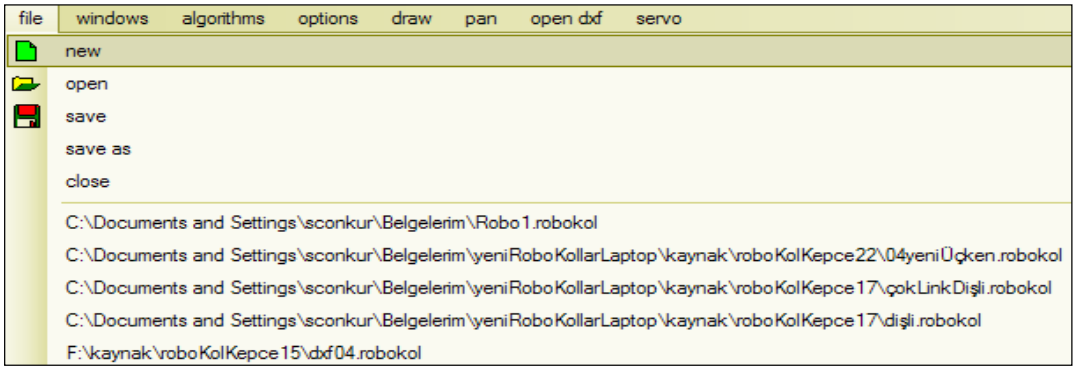
Şekil 4.10'da görülen *optionDialog* sınıfı (Şekil 4.15'te bu sınıfın arayüzü görülmektedir) *RoboKol* programında kullanıcının bazı çok kullanılan değerleri ayarlayabilmesini sağlar. Ekrandaki robot veya robotlar, engeller ve bu sınıf değişkenlerine verilen değerler kullanıcı dokümanı kaydettiğinde, doküman ile saklanır. Kullanıcı dokümanı açtığında değiştirdiği değerlerin saklandığını görecektir.

4.1.3. RoboKol Programının Menüleri

RoboKol sınıfının MDI özelliğine sahip bir program olduğu Kısım 4.1.2'de bahsedilmişti. Şekil 4.11'de *RoboKol* programının ana çerçevesi içinde iki tane açık doküman görülmektedir. Bu dokümanlardan istenilen dokümanın başlık kısmına çift tıklanarak, bu dokümanın tüm formu kaplaması sağlanabilir. Ayrıca yine Şekil 4.11'de de görülen "Windows" menüsü bu dokümanlara "tile horizontal", "tile vertical" ve "cascade" düzenlerinin verilmesini sağlar. Geliştirilmiş olan diğer menüler aşağıda açıklanmıştır.

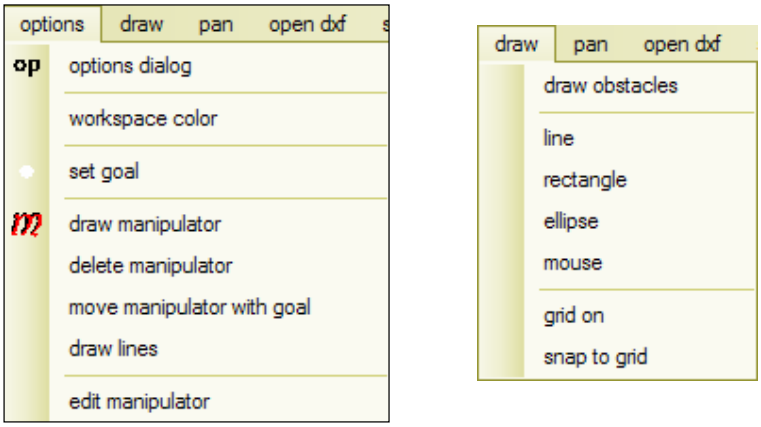


Şekil 4.11 *RoboKol* programının MDI özelliğini gösteren açık iki dokümanlı arayüzü

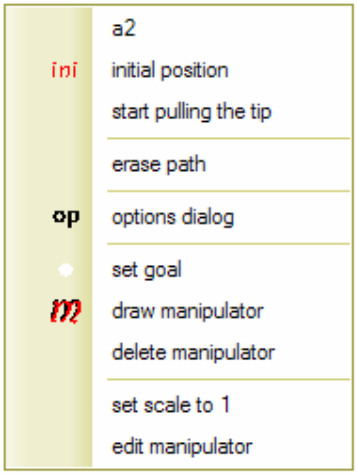


Şekil 4.12 Program *file* menüsü

Şekil 4.12'de görülen *file* menüsü standart doküman işlemlerini gerçekleştirir. Şekil 4.12'de görülen son açılan dokümanlar listesi MFC'de standart olarak gelmekteydi. Fakat yeni versiyonda hazır olmadığından bunun yazılması gerekti. Bu listeyi dokümanla kaydetmek mümkün olmadığından "Windows kayıt defterine" yazılması gerekti. En büyük zorluk da sıralamada ortaya çıktı. En son açılan dokümanın adının en üstte olması zorunluluğu probleminin çözümü üzerinde çok çalışılması gerekti. Fakat problem istenilen şekilde çözüldü. Bu listenin önemi doküman açmayı çok kolaylaştırmasıdır. Ayrıca, biraz daha ileri gidip program açılırken en son dokümanın da otomatik olarak açılması sağlandı. Böylece bir dosya üzerinde çalışırken devamlı olarak program kapatılıp açıldığında bu kullanıcıya çok zaman kazandırmaktadır. Şekil 4.13'te option ve draw menüleri görülmektedir.

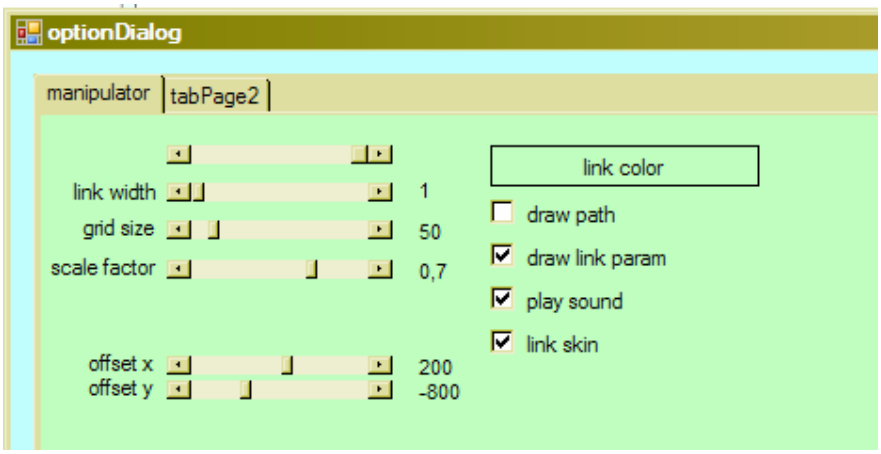


Şekil 4.13 Program *option* ve *draw* menüleri



Şekil 4.14 Program *context* menüsü

Şekil 4.14'te fare form üzerinde hareket ederken sağ tuşa basıldığında farenin olduğu yerde ortaya çıkan ve en çok kullanılan konutları içeren "context" menüsü görülmektedir. Şekil 4.15'te, *options* sınıfı incelenirken bahsedilen *options* dialoğu görülmektedir.



Şekil 4.15 Program *options* dialoğu

4.1.4. RoboKol Programında Kontrol Algoritmaları Geliştirilmesi

Yukarıda kısaca anlatılan program geliştirme aşamasının bilindiği gibi hedefi gereğinden çok serbestlik dereceli robot kolları için kinematik kontrol algoritmaları geliştirmektir. Program belli bir olgunluğa eriştikten sonra algoritma geliştirilmeye başlandı. Bundan sonraki aşamalarda hem program gerektikçe modifiye edildi hem de onun içinde algoritmaları geliştirildi.

Engellerin olmadığı alanda robot uç noktasının verilen yörüngesini, robotun kaç tane uzvu olursa olsun, takip edebilecek algoritma geliştirildi. Bu yörünge sadece ileri doğru değil geriye doğru da olabilmektedir. Bu algoritmanın avantajları şöyle sıralanabilir:

- Basitlik
- Gerçek zamanlı çalışabilme
- Eklenebilirlik: Algoritmaya daha sonradan istenen bazı ek özellikler kolayca monte edilebilir. Bu bir anlamda lineer sistemlerdeki süperpozisyon prensibine benzetilebilir.
- Engeller olma durumunda engellerden kaçınmaya yardımcı olma

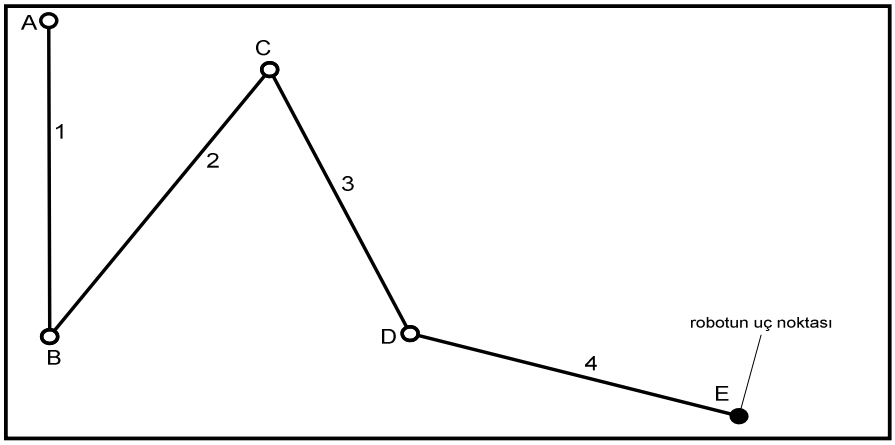
Şekil 4.16'da dört uzuvlu seri bir robot kolu görülmektedir. Buradaki dört uzuv, anlatımda basitlik için seçilmiştir. Algoritmada uzuv sınırlaması yoktur. Bilindiği gibi, robotun uç noktasının ulaşması gereken koordinatları verildiğinde bu koordinatlara ulaşmak için gereken uzuv açılarını hesaplamaya "ters kinematik" denmektedir. Bizim yapmaya çalıştığımız da gereğinden çok serbestlik dereceli robotlar için sayısal ters kinematik çözümleri geliştirmektir.

Şekil 4.17'de aynı robot kolunun iki konfigürasyonu çizilmiştir. Robot kolu birinci konfigürasyonda iken ikinci konfigürasyona ulaşması istenmektedir. Başka bir deyişle, robot kolunun uç noktası olan E noktasının E' noktasına gelmesi istenmektedir. Bu işlem gerçekleşirken de uzuvların mümkün olduğunca az hareket etmesi tercih edilmektedir.

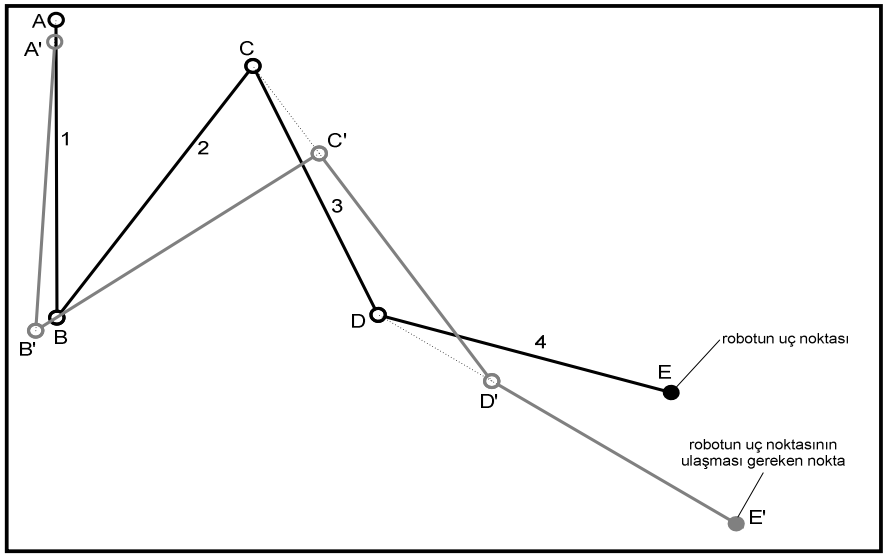
Şekil 4.17'ye tekrar bakılarak, E' noktasından D noktasına bir E'D vektörü çizilir. Bu vektör E' noktasında başlayıp D noktasında bitmektedir. Bu vektör kendi büyüklüğüne bölünürse E' noktasında başlayan ve E'D doğrultusunda olan bir birim vektör elde edilir. Şimdi bu birim vektör 4 nolu uzvun uzunluğuyla yani DE ile çarpılırsa D' noktası elde edilir. Böylece 4 nolu uzuv DE konumundan D'E' olan yeni konumuna gelmiş olur.

Benzer şekilde D'C arası çizilip D'C birim vektörü kullanılarak C' noktası, C'B arası çizilip C'B birim vektörü kullanılarak B' noktası ve B'A arası çizilip B'A birim vektörü kullanılarak A' noktası belirlendiğinde robot kolunun yeni konfigürasyonu ortaya çıkmış olur.

Uzuvlar birbirine dönel mafsallarla bağlı rijit cisimler olduğundan ve robot kolunun temeli yani A noktası sabit olduğundan henüz iş bitmiş değildir. Çünkü Şekil 4.17'de rahatlıkla görülebileceği gibi A noktası A' noktasına kaymıştır. Bu kayma geçici bir süre için dikkate alınmaz ve robot kolu fare ile hareket ettirilirse çok yumuşak hareketler elde edildiği görülür.



Şekil 4.16 Dört uzuvlu seri robot kolu

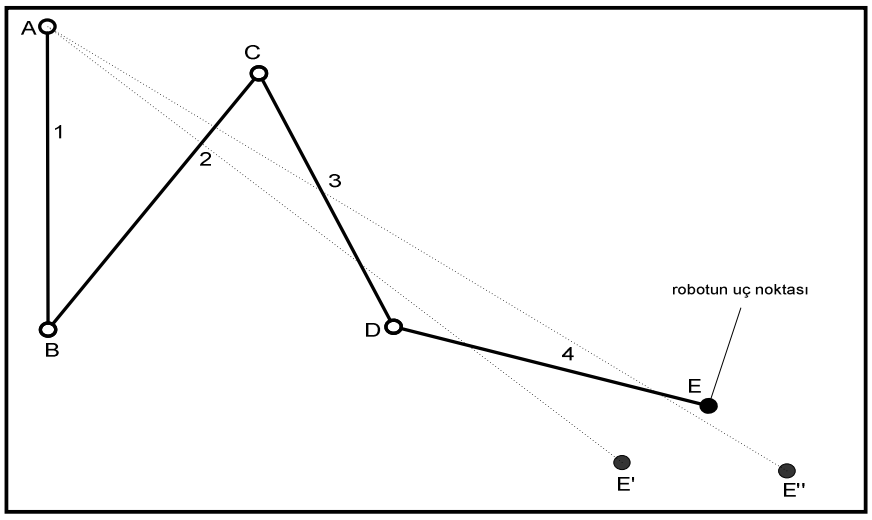


Şekil 4.17 Dört uzuvlu seri robot kolunun iki ayrı konfigürasyonu

Program yazılırken ve denenirken uzun bir zaman robot kolu bu halde kullanıldı. Bulunan çözüm şu şekilde açıklanabilir: Robot kolunun uç noktasıyla temel noktasının yeri değiştiriliyor. Yani A' noktasını robot kolunun uç noktası ve E' noktasını robot kolunun temel noktası kabul ediliyor. Şimdiki amaç bu ters dönmüş robot kolu ile A noktasına ulaşmak. Biraz önce açıklanan metot kullanılarak A noktasına ulaşılıyor. Bu sefer de robot kolunun şimdiki temel noktası olan E' noktası kayıyor. Robot kolunu eski haline getirip algoritma tekrar uygulanıyor.

Robot kolunun uç ve temel noktalarının sırayla değiştirilmesi hata sıfır olana kadar, yani robotun temeli A noktasına ucu da E' noktasına gelene kadar devam ediliyor. Buradaki en önemli soru şu: Acaba kaç iterasyondan sonra bu yerleşme gerçekleşiyor? Yapılan denemelerin her defasında hata gittikçe azalarak genelde 3-5 en çok da 10-15 defada yerleşme gerçekleşiyor.

Yukarıda her ne kadar yumuşak hareketler elde edildiği söylense de önemli bir sorun henüz çözüm beklemektedir. Robot kolunun uzuvları bazı durumlarda iç içe geçerek robot kolu uzuvları karmakarışık bir hal almaktadır. Hedef noktası robot kolunun temelinden uzaklaşırken hiçbir problem ortaya çıkmamaktadır. Gözlemlerimiz sonucu bu problemin, robot kolunun ucu robot kolunun temeline doğru giderken ortaya çıktığı fark edildi. Bu durumda başka bir strateji belirlenmesi gerekti.



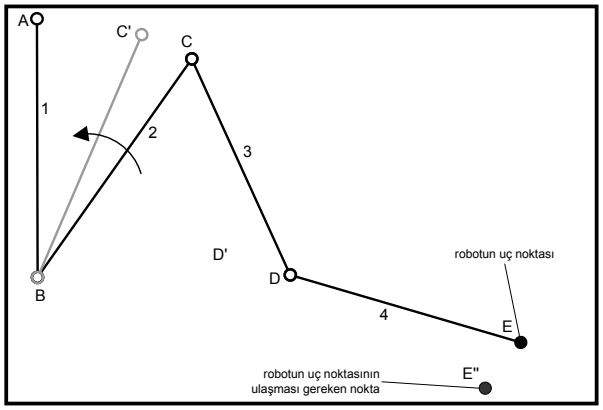
Şekil 4.18 "İleri" ve "geri" kavramlarının belirlenmesi

Şekil 4.18'e bakarak robotun uç noktasının E' ve E'' noktalarına gelmesinin istendiği farz edilir. "İleri" ve "geri" kavramlarının tanımlanmasında E'A ve EA doğrularını kullanıyoruz. E'A doğrusunun uzunluğu, EA doğrusunun uzunluğundan daha kısa olduğu için robotun E' noktasına olan hareketini "geriye doğru" olarak tanımlanır. E''A doğrusunun uzunluğu, EA doğrusunun uzunluğundan daha uzun olduğu için robotun E'' noktasına olan hareketini "ileriye doğru" olarak tanımlanır.

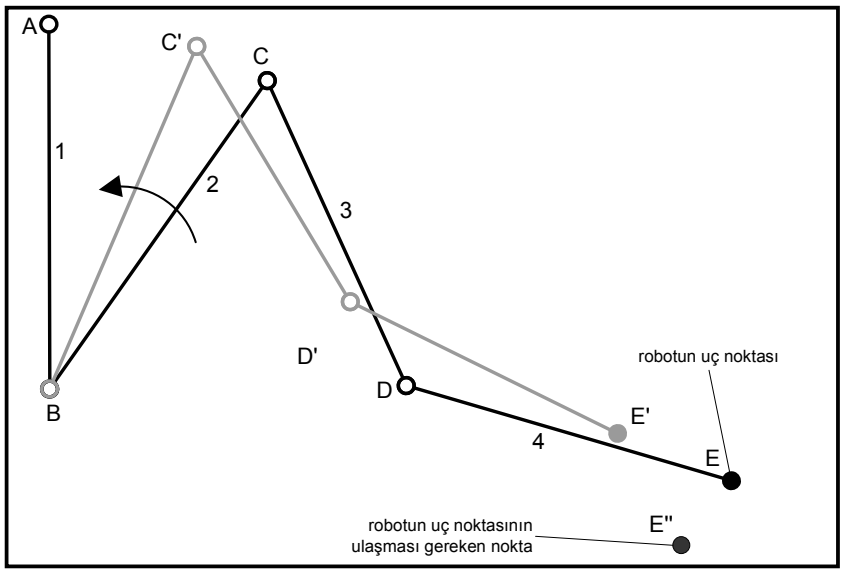
Şekil 4.19'da görülen robot kolunun "geriye doğru" hareketini incelenirse 1 nolu uzvun hareket sınırlarına eriştiğini ve artık harekete dahil edilmediğini düşünürsek bu durumdaki bir sonraki uzuv, yani 2 nolu uzuv, saat tersi yönde önceden belirlenen bir miktar döndürülür ve 2 nolu uzuv üzerindeki C noktası C' noktasına gelir. Robot kolu 3 nolu uzvu hala C noktası üzerindedir.

Şimdi bu C noktasını robotun uç noktası, E noktasını da robotun temeli olarak alıp "ileri doğru hareket" algoritması "bir" defa uygulanır. Bu durumda Şekil 4.20'de görüldüğü gibi 3 nolu uzvun C noktası da C' noktasına gelir. Fakat önceden de bilindiği gibi bir defa uygulanan "ileri doğru hareket" algoritmasında temel noktası kayacaktır ve E noktası E' noktasına gelecektir.

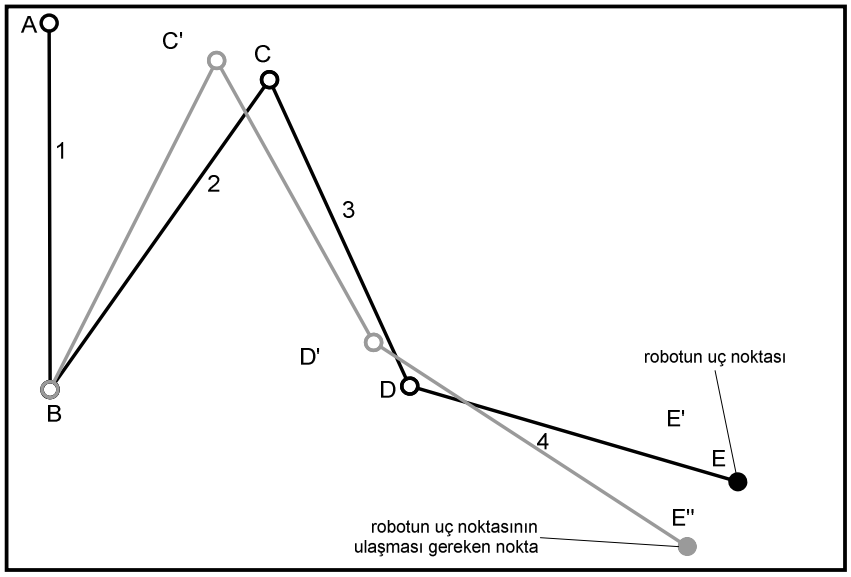
Şekil 4.20'ye tekrar bakıldığında robot kolu artık öyle bir pozisyona gelmiştir ki bu pozisyon çok iyi bilinen "ileri gitme" posizyonudur. "İleri gitme" algoritması uygulanarak robot kolu Şekil 4.21'de görülen E'' konumuna gelir. Böylece "geriye gitmede" 3 ve 4 nolu uzuvlar aşırı derecede birbirine yakınlaşmaz ve mafsal limitleri ihlal edilmez.



Şekil 4.19 "2" nolu uzvun geriye alınması

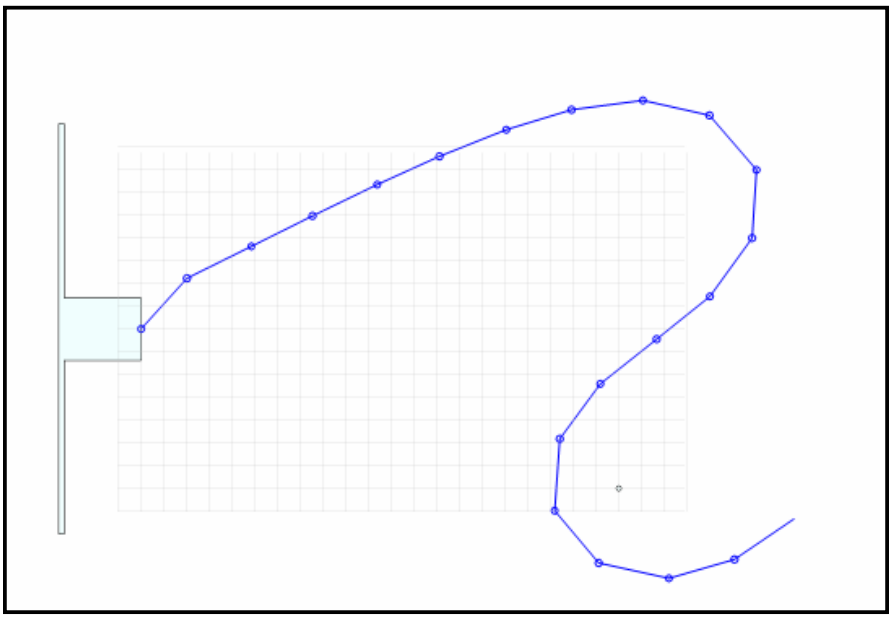


Şekil 4.20 "2" nolu uzvun ve diğerlerinin geriye alınması



Şekil 4.21 "Geriye doğru" hareketin tamamlanması

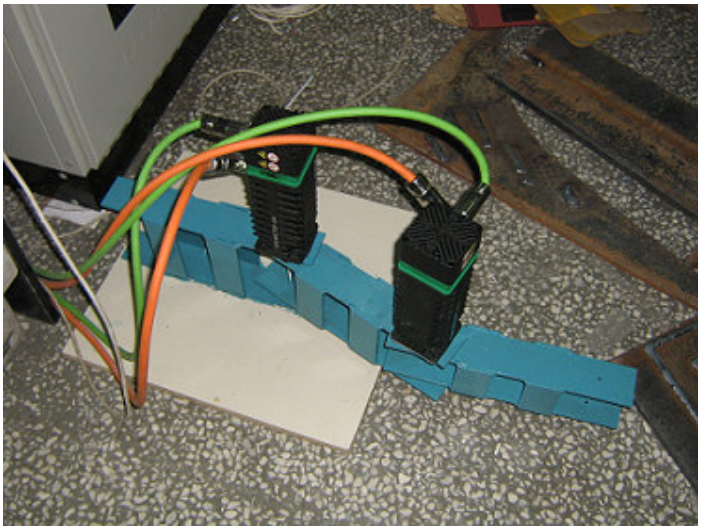
Bu metotta hareketler ileriye ve geriye doğru yumuşak bir şekilde gerçekleşmekte, Şekil 4.22'de görülen robot kolunun tabii kıvrımları korunmakta ve ani değişmelere izin verilmemektedir.



Şekil 4.22 Robot kolunun ideal hareketi

4.1.5. İki Uzuvlu Robot Kolu Dizaynı ve Çalıştırılması

İki uzuvlu robot kolu mekanik ve elektrik tasarımı yapılarak, yazılan hareket kontrol algoritması çalıştırılmıştır. Şekil 4.23'te gösterilen robot kolu iki eksenlidir, ve servo motorlar yüke doğrudan akuple edilmiştir. Yazılan programdan servo motorların konum kontrolleri ve hız kontrolleri istenilen düzeyde gerçekleştirilememiştir. Projede kullanılan servo motorların tork değerleri, redüktörsüz montaj için yetersiz gelmiştir.



Şekil 4.23 Tasarlanan iki uzuvlu robot kolu

4.1.6. Lazer Mesafe Ölçme Sensörü

Tasarlanan prototip robot kolunun, sanayi ortamında çalışabilmesi için, robot kolunun son uzuvu üzerine herhangi bir yörüngeyi takip edebilmeği sağlayacak hassas bir algılayıcı olan mikron bazında ölçüm yapabilen lazer sensör takılmıştır (Şekil 4.24).

Lazer sensorden gelen 4-20 mA mesafe bilgisi, eksen kontrol kartında fazladan anolog giriş olmadığı için RS-485 haberleşme protokolü ile robot kolunu kontrol eden bilgisayar programına aktarılmıştır.

Şekil 4.24 Lazer sensör

4.1.7. Robotun Mafsalları Üzerine Ek Bir Enkoder Yerleştirilmesi

Prototip robot kolunda kullanılan redüktörler standart redüktör olduğundan dolayı dişli boşlukları bulunmaktaydı. Bilgisayardan gönderilen hedef pozisyon değerine ulaşıp ulaşamadığımızı kontrol edebilmek ve sistemin güvenilirliğini arttırmak için mafsallar üzerine ek bir enkoder yerleştirilmesi çözüm olarak düşünülmüştür. Bu amaçla ek bir enkoder tek bir mafsal üzerine yerleştirilerek servo sürücüye bağlanıp, servo sürücünün RS-485 haberleşme portundan bilgisayar programına enkoder bilgisi aktarılmıştır. Bu alınan enkoder bilgisi robot kolu hareket kontrol algoritması içine entegre edilecektir.

5. SONUÇ VE ÖNERİLER

Bu tezde, motorların kurulması, ayarlanması ve kontrol kartıyla bilgisayardan kontrolü gerçekleştirilmiştir. Buna ek olarak, bu prototipte kullanılan kontrol algoritmaları incelenmiştir. Robotun mafsalları üzerinde ek bir enkoder yerleştirilebilmesi için gerekli çalışmalar yapılmış, bu enkoderlerden alınan bilgi bilgisayar programına aktarılmıştır. Benzer şekilde robotun uç noktasına takılan bir lazer sensörüyle seri port vasıtasıyla iletişim kurulmuş ve sensör verisi bilgisayar programına aktarılmıştır.

Tasarlanan ve kurulan elektrik panosu ve eksen kontrol kartı donanımları çalıştırıldı. Hareket kontrol algoritması Visual C++.NET ile yazılarak çalıştırıldı. Eksen kontrol kartında hazır bulunan seri haberleşme portuna bağlanabilen çevresel giriş ve çıkışların bağlanacağı ek giriş, çıkış modüllerinin kullanılmasının sistemin fonksiyonelliğini arttıracağı gözlendi.

Sistemde kullanılabilecek redüktörlerin boşluksuz olması ve daha küçük gövde boyutlarında olması tasarlanan robot kolunun fonksiyonelliğini arttıracaktır.

Hareket kontrol algoritmasının yazılmasında kullanılan Visual C++.NET programlama dilinin nesneye yönelik olması ve sınıf yapısı, programlamayı kolaylaştırmış ve hızlandırmıştır. Sistemde kullanılan bilgisayarın sanayi tip bilgisayar olması sistemin güvenilirliğini arttıracaktır.

İki uzuvlu robot kolu mekanik ve elektrik tasarımı yapılarak, yazılan hareket kontrol algoritması çalıştırılmıştır. Servo motorlar yüke doğrudan akuple edilmiştir. Yazılan programdan servo motorların konum kontrolleri ve hız kontrolleri istenilen düzeyde gerçekleştirilememiştir. Projede kullanılan servo motorların tork değerleri, redüktörsüz montaj için yetersiz gelmiştir. Çözüm olarak motorların tork değerlerinin yükseltilmesi ya da boşluksuz redüktör kullanılması düşünülmüştür.

KAYNAKLAR

E. S. Conkur and R. Buckingham, “Clarifying the definition of redundancy as used in robotics”, Robotica 15 (5), 583-586 (1997).

Ma S., Hirose S., Yoshinada H. 1995. Development of a hyper-redundant multijoint manipulator for maintenance of nuclear reactors, Advanced Robotics 9 (3): 281-300.

Buckingham R. 1993. Towards safe active robotic devices for surgery, Industrial Robot 20 (2): 8-11.

A. A. Maciejewski and C. A. Klein, “Obstacle avoidance for kinematically redundant manipulators in dynamically varying environments”, *The International Journal of Robotics Research* 4 (3), 109-117 (1985).

J. L. Chen, J. S. Liu, W. C. Lee and T. C. Liang, “On-line multi-criteria based collision-free posture generation of redundant manipulator in constrained workspace”, *Robotica* 20 (6), 625–636 (2002).

D. N. Nenchev, “Redundancy resolution through local optimisation: a review”, *Journal of Robotic Systems* 6 (6), 769-798 (1989).

C. L. Boddy and J. D. Taylor, “Whole arm reactive collision avoidance control of kinematically redundant manipulators”, *IEEE International Conference on Robotics and Automation* (1993), pp. 382-387.

Y. Nakamura, *Advanced robotics*, *redundancy and optimisation* (Addison-Wesley Pub. Company, Reading, 1991).

J. Latombe, *Robot motion planning* (Kluwer Academic Publishers, USA, 1991).

D. Hsu, J. Latombe and R. Motwani, “Path planning in expensive C-spaces”, *IEEE International Conference on Robotics and Automation* (1997), pp. 2719-2726.

N. Amato and Y. Wu, “A randomised roadmap method for path and manipulation planning”, *IEEE International Conference on Robotics and Automation* (1996), pp. 113-120.

K. K. Gupta, “Fast collision avoidance for manipulator arms: a sequential search strategy”, *IEEE Transactions on Robotics and Automation* 6 (5), 522-532 (1990).

A. Hayashi, Geometric motion planning for highly redundant manipulators using a continuous model (PhD Thesis, The University of Texas at Austin, 1994).

O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots". *The International Journal of Robotics Research* 5 (1), 90-98 (1986).

F. Janabi-Sharifi and D. Vinke, "Robot path planning by integration the artificial potential field approach with simulated annealing", *IEEE International Conference on Robotics and Automation* (1993), pp. 282-287.

S. W. Kim and D. Boley, "Building and navigating a network of local minima", *Journal of Robotic Systems* 18 (8),405–419 (2001).

C. Connolly and R. Grupen, "The application of harmonic functions to robotics", *Journal of Robotic Systems* 10 (7), 931–946 (1993).

E. Rimon and D. E. Koditschek, "Exact robot navigation using artificial potential functions*", IEEE Transactions on Robotics and Automation* 8 (5), 501-517 (1992).

B. Faverjon and P. Tournassoud, "A local based approach for path planning of manipulators with a high number of degrees of freedom", *IEEE International Conference on Robotics and Automation* (1987), pp. 1152-1159.

D. Reznik and V. Lumelsky, "Sensor-based motion planning in three dimensions for a highly redundant snake robot", *Advanced Robotics* 9 (3), 255-280 (1995).

F. Fahimi, H. Ashrafiuon and C. Nataraj, "Obstacle avoidance for spatial hyper-redundant manipulators using harmonic potential functions and the mode shape technique", *Journal of Robotic Systems* 20 (1), 23-33 (2003).

H. Mochiyama, "Kinematics of the whole arm of a serial-chain manipulator", *Advanced Robotics* 15 (2), 255-275 (2001).

J. Z. Li and M. B. Trabia, "Adaptive path planning and obstacle avoidance for a robot with large number of redundancy", *Journal of Robotic Systems* 13 (3), 163-176 (1996).

P. J. Choi, J. A. Rice and J. C. Cesarone, "Kinematics of an indefinitely flexible robot arm", *Journal of Robotics Systems* 10 (4), 407-425 (1993).

M. W. Hannan and I. D. Walker, "Kinematics and the implementation of an elephant's trunk manipulator and other continuum style robots", *Journal of Robotic Systems* 20 (2), 45–63 (2003).

S. Ma and M. Konno, "An obstacle avoidance scheme for hyper-redundant manipulators-global motion planning in posture space", *IEEE International Conference on Robotics and Automation* (1997), pp. 161-166.

S. Ma, I. Kobayashi, S. Hirose and K. Yokoshima, "Control of a multijoint manipulator moray arm", *IEEE/ASME Transactions on Mechatronics*, **7** (3), 1-14 (2002).

S. Ma and I. Kobayashi, "An obstacle avoidance control scheme for the Moray arm on the basis of posture space analysis", *Robotics and Autonomous Systems* 32 (2-3), 163–172 (2000).

G.S. Chirikjian, J.W. Burdick, Parallel formulation of the inverse kinematics of modular hyper-redundant manipulators, in: Proceedings of the IEEE International Conference on Robotics and Automation, 1991, pp. 708–713.

WEB_1. (2006), www.controltechniques.com

WEB_2. (2006), www.controltechniques.com

WEB_3. (2006), www.controltechniques.com

WEB_4. (2006), www.controltechniques.com

WEB_5. (2006), www.controltechniques.com

WEB_6. (2006), www.controltechniques.com

WEB_7. (2006), www.controltechniques.com

WEB_8. (2006), www.controltechniques.com

WEB_9. (2006), www.triomotion.com

WEB_10. (2006), www.triomotion.com

WEB_11. (2006), www.triomotion.com

ÖZGEÇMİŞ

1977 yılında Afyon Sandıklı'da doğdu. İlk ve orta okul öğrenimini Afyon Dinar Atatürk İlkokulu ve Atatürk Ortaokulu'nda tamamladı. Ortaokul sonunda yapılan sınavla Isparta Gönen Anadolu Öğretmen Lisesini kazandı ve okul birincisi olarak lise öğrenimini tamamladı. 1999 yılında İstanbul Teknik Üniversitesi Elektronik ve Haberleşme Mühendisliği bölümünden mezun oldu. 2000 yılında askerlik görevini tamamladı. 2000 yılından itibaren Denizli'de bulunan özel bir şirkette proje mühendisi olarak çalışırken 2006 yılının başında kendi şirketini kurdu.