T.C.
BOZOK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
MEKATRONİK MÜHENDİSLİĞİ ANABİLİM DALI

Yüksek Lisans Tezi

3 EKSENLİ ROBOT KOLUNUN MVPL(MICROSOFT VISUAL PROGRAMMING LANGUAGE) İLE SİMÜLASYONU

Ali Kemal DURKAYA

Tez Danışmanı
Yrd. Doç. Dr. Orhan ER

Yozgat 2013

T.C.
BOZOK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
MEKATRONİK MÜHENDİSLİĞİ ANABİLİM DALI

Yüksek Lisans Tezi

3 EKSENLİ ROBOT KOLUNUN MVPL(MICROSOFT VISUAL PROGRAMMING LANGUAGE) İLE SİMÜLASYONU

Ali Kemal DURKAYA

Tez Danışmanı
Yrd. Doç. Dr. Orhan ER

Yozgat 2013

T.C.
BOZOK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

TEZ ONAYI

Enstitümüzün Mekatronik Mühendisliği Anabilim Dalı70111710011numaralı öğrencisi Ali Kemal DURKAYA'nınhazırladığı **"3 EKSENLİ ROBOT KOLUNUN MVPL(MICROSOFT VISUAL PROGRAMMING LANGUAGE) İLE SİMÜLASYONU"** başlıklı YÜKSEK LİSANS tezi ile ilgili TEZ SAVUNMA SINAVI, Lisansüstü Eğitim-Öğretim ve Sınav Yönetmeliği uyarınca 07/10/2013Pazartesi günü saat 15:00 yapılmış, tezin onayına OY BİRLİĞİYLE karar verilmiştir.

Başkan : Yrd.Doç.Dr. Mustafa YAZ

Üye : Yrd.Doç.Dr.Orhan ER (Danışman)

Üye : Yrd.Doç.Dr.Halit ÖZTEKİN

ONAY:

Bu tezin kabulü, Enstitü Yönetim Kurulu'nun ……../……../2013 tarih ve ……..… sayılı kararı ile onaylanmıştır.

...../...../2013

(Ünvanı, Adı Soyadı)
Müdür

İÇİNDEKİLER

Sayfa

ÖZET……... **iii**

ABSTRACT…… **iv**

TEŞEKKÜR…… **v**

TABLOLAR LİSTESİ……... **vi**

ŞEKİLLER LİSTESİ……. **vii**

1. GİRİŞ…….. **1**

2.LİTERATÜR ÇALIŞMASI……. **3**

3. SİSTEM MODELLEME……... **4**

4. ROBOT KOLU SİSTEMİNİN MODELLENMESİ VE KONTROLÜ…… **6**

4.1. İleri Kinematik Analiz…….. **6**

4.1.1. Koordinat Sistemlerinin Eklemelere Yerleştirilmesi……... **9**

4.1.2. İleri Kinematik Problemlerinin Çözümünde Kullanılan Yaklaşımlar……. **10**

4.1.2.1. Geometrik Yaklaşım…… **10**

4.1.2.2. Cebirsel Yaklaşım…… **10**

4.2. Ters Kinematik Analiz……. **15**

4.2.1. Ters Kinematik Problemlerine Analitik Çözüm Yaklaşımı……. **18**

5. KUKA ROBOT KOLU MODELİ…… **26**

5.1. Denavit-Hartenberg Parametreleri……... **26**

5.2. İleri Kinematik Hesaplamaları……. **28**

6. KOD PLATFORMU……. **31**

6.1. Servis Kavramı…… **32**

6.2. Windows Form Yapısı ve Servis İletişimi……... **37**

6.3. Robot Kolu Modelini Kodlama…… **41**

7. LABORATUAR UYGULAMASI…….. **51**

SONUÇ ……... **68**

KAYNAKLAR…… **69**

ÖZGEÇMİŞ…… **70**

3 EKSENLİ ROBOT KOLUNUN MVPL(MICROSOFT VISUAL PROGRAMMING LANGUAGE) İLE SİMÜLASYONU

Ali Kemal DURKAYA

Bozok Üniversitesi
Fen Bilimleri Enstitüsü
Mekatronik Mühendisliği Anabilim Dalı
Yüksek Lisans Tezi

2013; Sayfa: 71

Tez Danışmanı: Yrd. Doç. Dr. Orhan ER

ÖZET

Bu çalışmada, 3 eksenli bir robot kolunun yeni nesil bir programlama dili kullanılarak modellenmesi ve web üzerinde uygulamaya sunulması hedeflenmiştir. Robot kolunun programlama dili içerisinde modelinin oluşturulması ve MVPL ile modelin simülasyonu gerçekleştirilmektedir.

Simülasyon ortamı Microsoft Robotic Developer Studio 2008 R3 olarak seçilmiş ve MVPL dili kullanılarak modelin simülasyonunun oluşturulması sağlanmıştır. Oluşturulan simülasyon, seçilen ortamın web desteği sayesinde internet ortamında sunulmaktadır. 3 eksenli robot kolu matematiksel olarak modellendikten sonra seçilen ortam üzerinde kodlanarak simüle edilmiştir.

Anahtar Kelimeler: 3 eksenli robot kolu, simülasyon, MVPL, Robotic Developer Studio.

SIMULATION OF 3 AXIS ROBOT ARM USING MVPL(MICROSOFT VISUAL PROGRAMMING LANGUAGE)

Ali Kemal DURKAYA

Bozok University
Graduate School of Natural andAppliedSciences
Department of MechatronicsEngineering
Master of Science Thesis

2011; Sayfa:71

Thesis Supervisor: Assist. Prof. Orhan ER

ABSTRACT

In this study, simulating of 3 axis robot arm using by new generation programming language and publishing simulation to web is aimed. Creating model of Robot arm with programming language and simülating model using by MVPL (Microsoft Visual Programming Language) is actualised.

Microsoft Robotic Developer 2008 R3 is selected as simulating platform and creating simülating of model is provided by using MVPL(Microsoft Visual Programming Language). After mathematical modelling of 3 axis robot arm, model is simülated by coding on selecting platform.

Keywords:3 axis robot arm, simülation, MVPL, Robotic Developer Studio

TEŞEKKÜR

Bu araştırma için beni yönlendiren, karşılaştığım zorlukları bilgi ve tecrübesi ile aşmamda yardımcı olan, gerekli desteği hiçbir zaman benden esirgemeyen değerli Danışman Hocam Sayın Yrd. Doç. Dr. Orhan ER'e teşekkürlerimi sunarım.

TABLOLAR LİSTESİ

			Sayfa
Tablo 4.1	**:**	D-H Değişkenleri……..	**9**
Tablo 4.2	**:**	D-H Değişkenleri……..	**13**
Tablo 5.1	**:**	Denavit-Hartenberg Parametreleri……	**26**
Tablo 6.1	**:**	MoveTo Parametreleri……..	**47**
Tablo 6.2	**:**	MoveToPosition Parametreleri…….	**49**

ŞEKİLLER LİSTESİ

			Sayfa
Şekil 3.1	**:**	Sistem Modeli……..	**5**
Şekil 4.1	**:**	Dönme Eksenlerin Koordinat Sisteminin Yerleştirilmesi	**7**
Şekil 4.2	**:**	a_{i-1} Uzuv Uzaklığı……	**7**
Şekil 4.3	**:**	d_i eklem kayması……..	**8**
Şekil 4.4	**:**	α_{i-1} eksen açısı……..	**8**
Şekil 4.5	**:**	θ_i eklem açısı……	**8**
Şekil 4.6	**:**	İki boyutlu düzlemde hareket eden robot kolu…….	**10**
Şekil 4.7	**:**	İki boyutlu düzlemde hareket eden P noktasının konumu	**11**
Şekil 4.8	**:**	İki boyutlu düzlemde hareket eden kolun başlangıç değerleri……	**11**
Şekil 4.9	**:**	Dönme Eksenlerine Z Eksenlerinin Yerleştirilmesi…….	**11**
Şekil 4.10	**:**	X Eksenlerinin Yerleştirilmesi…….	**12**
Şekil 4.11	**:**	Sağ El kuralına göre Y Eksenlerinin Yerleştirilmesi……	**12**
Şekil 4.12	**:**	Eklem Değişkenleri ile Kartezyen Uzay Dönüşümü……	**16**
Şekil 4.13	**:**	Fiziksel Çözüm…….	**16**
Şekil 4.14	**:**	Matematiksel Çözüm……	**18**
Şekil 4.15	**:**	Robotun Aynı Noktaya Dört Farklı Şekilde Ulaşması….	**18**
Şekil 4.16	**:**	İki Boyutlu Düzlemde Hareket Eden Robot Kolu……	**19**
Şekil 4.17	**:**	İki boyutlu düzlemde hareket eden P noktasının konumu	**24**
Şekil 4.18	**:**	Ters Kinematik Probleminin Gerçek Çözümlerine Ait Konumları…….	**25**
Şekil 4.19	**:**	Ters Kinematik Probleminin Sanal Çözümlerine Ait Konumları…….	**25**
Şekil 5.1	**:**	1.Eklem Açısı……	**28**

Şekil 5.2 **:** 3.Eklem Açısı…… **29**

Şekil 5.3 **:** 2. Eklem Açısı……... **30**

Şekil 6.1 **:** Dssservice1 hizmetinin yürütülmesi……. **34**

Şekil 6.2 **:** localhost:50000……. **35**

Şekil 6.3 **:** Yürütülen Servis Dizini…… **35**

Şekil 6.4 **:** dssservice1…… **36**

Şekil 6.5 **:** dssservice1verileri……. **36**

Şekil 6.6 **:** L5 ve L6 Geometrisi……. **41**

Şekil 7.1 **:** Sistemin Çalışma Ortamı…….. **51**

Şekil 7.2 **:** Run DSS Node komut penceresi…….. **52**

Şekil 7.3 **:** Servis Web Hizmeti…….. **52**

Şekil 7.4 **:** Servisi Başlatma…… **53**

Şekil 7.5 **:** Simülasyon ortamı…… **54**

Şekil 7.6 **:** Tuş kontrol…… **54**

Şekil 7.7 **:** Kontrol Arayüzü……... **55**

Şekil 7.8 **:** Grafik Arayüzü……. **63**

GİRİŞ

Günümüzde robotlar çoğu endüstriyel sistemlerde insanın yerini almakta ve üretime büyük katkıda bulunmaktadır. Bu teknolojinin getirdiği avantajlar;

1. Çalışma saatlerinin insanlar kadar sınırlı olmaması
2. İnsanların çalışamayacağı kadar zor şartlar ve ortamlarda çalışabilmeleri
3. Herhangi bir problem veya yenileme durumunda onarılabilmeleri ve yenilenebilmeleri
4. Daha hassas, doğru ve hızlı sonuçlar üretebilmeleri

Gelişen teknoloji ile beraber endüstriyel sistemlerde robot kollarının kullanılmasının getirdiği avantajları yanında bir takım dezavantajları da bulunmaktadır. Karar verme yetenekleri oluşturuldukları modele göre sınırlı ve insanlar kadar gelişmiş değildir. Kullanılan robot kolu veya robotik teknolojinin çeşidine göre maliyeti oldukça artabilmektedir. Ancak seri üretim verimliliğinin yüksek olması bu dezavantajı ortadan kaldırabilmektedir.

Robot kolları çalışma şartları ve yapılacak işin niteliğine göre farklı yapılarda tasarlanmaktadır. Genel olarak 3 eksende hareket eden ve insan kolu örnek alınarak 6 serbestlik derecesine sahip robot kolu tasarımı göze çarpar. İnsan kolunun örnek alınması ve benzer eklemlerin oluşturulmaya çalışılması robot kolu modellerinde bir takım zorlukları da beraberinde getirmektedir. Eklem hareketindeki en büyük problem tork problemidir. Robot kolunun düşük hızda çalışmasına sebep olur. Aksine hızlı hareket edebilen bir robot kolu yapılan işin amacına daha çabuk ulaşmasını sağlar. Hızlı hareket yeteneğine sahip robot kollarında tork düşüktür veya eklem hareketini sağlayan motor kontrol devreleri oldukça karmaşıktır. Düşük tork problemi, düşük güçlü motorlarda dişli sistemleriyle ortadan kaldırılmaktadır.

Bir robot kolunun istenilen şartlarda istenilen işleri yapıp yapamayacağının anlaşılması için onun fiziksel olarak oluşturulması hem mali hem de zaman fizibilitesi açısından bir takım olumsuzlukları içerebilmektedir. Bu nedenle oluşturulmak istenen robot kolunun testi için matematiksel modelini oluşturmak

vemodeli bilgisayar sistemleri üzerinde simüle etmek fiziksel gerçekliğe göre daha ucuz ve daha kısa zamanda sonuç vereceğinden tercih edilmektedir.

Bu çalışmada da 3 eksenli 6 serbestlik derecesine sahip bir robot kolu modelinin oluşturulması ve oluşturulan modelin bilgisayar ortamında simüle edilerek uygun bir ağ üzerinden hizmet vermesi amaçlanmıştır. Çalışma ile simülasyonu oluşturulan robot kolu uygulamasının ağ üzerinden kullanılması da amaçlanarak hem lisans hem de yüksek lisans düzeylerinde verilmekte olan konu ile ilgili çalışmalara destek verilmesi amaçlanmıştır. Laboratuar uygulaması olarak kullanabilecek olan sistem robot kolu konusunda eğitim almakta olan öğrencilerin istedikleri uygulamaları deneysel olarak test etmelerini ve sonuçlarını birebir görmelerini hedeflemektedir. Robot kolu ve kolun kullanılacağı simülasyon ortamı C# programlama dili modellenmiş olup Microsoft Visual Programming Language kullanılarak kolaylıkla kullanılabilir hale getirilmiştir.

2. LİTERATÜR ÇALIŞMASI

Robot kolu uygulamaları genellikle nesne tabanlı programlama dilleri tercih edilerek masaüstü uygulamalar şeklinde yapılmaktadır. Masaüstü uygulamalar kişisel makinelerde işletim sistemi üzerinden yürütülen programlardır ve sadece o makine kullanıcısına hizmet verebilirler. Bu tür çalışmalar bir yöntemin uygulanması ve sonuçlarının alınması gibi verilere ışık tutmak amacıyla gerçekleştirilir. Bir çalışmada beş eksenli bir robot kolu modeli matlab ortamında oluşturulmuş ve yörünge analizi çalışması yapılmıştır [1]. Yapılan bir başka çalışmada engel takibi amacıyla Delphi7 programlama dili kullanılarak robot kolu simnülasyonu gerçekleştirilmiştir [2].

Bazı uygulamalar ise java tabanlı olup, web üzerinden bir sunucu yardımıyla robot kolunun görsel olarak hareketlerini ifade edebilmek amacıyla gerçekleştirilmiştir.
Yapılan web tabanlı bir modelde java programlama dili ve applet yapısı kullanılarak herhangi bir internet browser üzerinden robot kolu modeline erişim sağlanmıştır [3]. Ancak hizmet üzerinden farklı uygulamalar geliştirmek ve bunu laboratuar ortamına açabilmek mümkün değildir.

3. SİSTEM MODELLEME

Fiziksel dünyadaki bir olayın, sürecin veya birimler ile beraber birimler arasındaki iç ve dış ilişkilerini inceleyen bir sistemin belli bir anlatımına model denir. Model anlatımı sözle, çizimle, belli bir oranda fiziki benzer oluşturmakla veya bilimin ortak dili olan matematik ile yapılabilir.

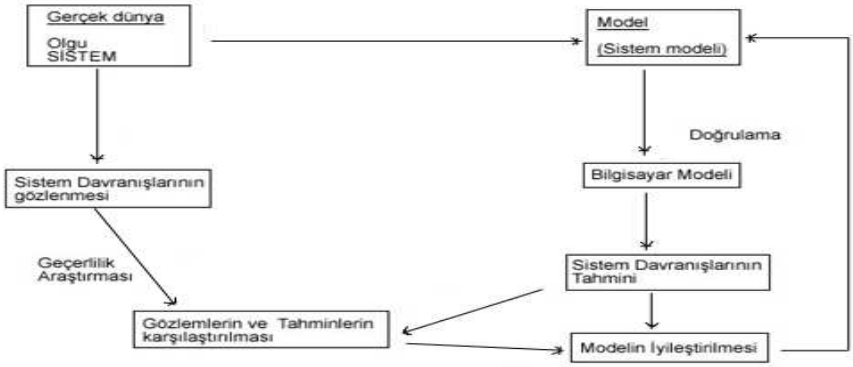
Modellenecek olgu ilgili olduğu bilim sahasının kavram ve kanunlarına bağlı olarak ifade edilir. İfade edilmek istenen olgu ve sistemler çok karmaşık olması nedeniyle basitleştirme yöntemlerine gidilerek belli varsayımlar ile modellenir. Oluşturulan model sistemin gerçeği değil model kurucunun anlayışı ürünüdür.

Her model oluşturma işi bir soyutlama süreci gerektirir. Soyutlama süreci, gerçek dünyadaki olguların ayrıntılarından arındırılmış veya istenen detayları alınıp sonuca etki etmeyen ayrıntılardan soyutlanmış örüntülerin insan düşüncesine aktarımıdır.Bir modeli oluşturabilmek için söz konusu sistemin temel özelliklerini, durumları arasındaki iç ve çevre ile olan ilişkilerini bilmek gerekir. Bir modelin başarısı, pratik ve bilimsel yararlılığı sistemin esasını soyutlamadaki doğruluğun derecesine ve dikkate alınan özelliklerin temel niteliklere uyumluluğuna bağlıdır.

Matematiksel yöntemlerle oluşturulan modeller anlatım gücü en fazla ve en geçerli olan modellerdir. Bu model türü değişkenlerin durumu, lineerlik, süreklilik veya ayrık zamanlı gibi farklı model sınıflarında olabilir. Değişkenlerin durumuna göre rastgele değişkenler içeren matematiksel modellere stokastik, içermeyenlere deterministik matematiksel model adı verilir. Matematiksel modeller lineer ve non-lineer matematiksel model olarak da sınıflandırılabilirler. Bir matematiksel model diferansiyel denklemler kullanılarak ifade ediliyorsa sürekli, fark denklemleriyle ifade ediliyorsa ayrık zamanlı matematiksel model olarak ifade edilir.

Belli bir olgunun matematiksel olarak modellenmesi girdi ve ölçümler sonucu elde edilen sayısal değerlerden oluşur. Sayısal veriler, başlangıç değerleri gibi bilgiler, oluşturulan matematiksel model yani matematiksel ifadelerden geçerek sistemin sonucunu yani davranışını ortaya koyar. Model kurucunun gerçek dünyadaki sisteme bakışına bağlı olarak matematiksel model yukarıda bahsi geçen sınıflara ayrılabilir.

Bir modelin faydalı olup olmadığını sistemin girişlerine karşılık verdiği çıkışların doğruluğu ve bu çıkışların uygulanabilirliği belirler. Uygulanabilirlik analitik veya sayısal olabildiği gibi belli veriler altında gerçek dünyadakine benzer şekilde modeli çalıştırıp gelen sonuçları incelemek yoluyla da olabilir.



Şekil 3.1 Sistem Modeli

Sistemler, gerek birimleri arasındaki ilişkiler gerekse de çevre ile olan ilişkileri bakımından genellikle çok karmaşık yapıdadır. Bunlar, bazı basitleştirmeler ve varsayımlar altında modellenmektedir. Modelin, sistemi ne kadar iyi temsil ettiğinin araştırılması olan geçerlilik araştırması sırasında olası iki tür hata ile karşılaşılır. Birincisi, gerçekte geçerli olan bir modelin geçerlilik testleri sonucu reddedilmesi, ikincisi ise geçersiz olan bir modelin geçerlilik testleri sonucu kabul edilmesidir. Birinci olası durum model kurucunun, ikinci durum ise model kullanıcının riski olarak ortaya çıkar.

Bir modelin geçerliliğini sınamadan önce, model yapısındaki algoritma ve bilgisayar programlarında hata olup olmadığının araştırılması gerekmektedir. Bu işleme doğrulama denir.

Matematiksel modellerin faydalılığı, kullanılan matematiğin düzeyine, ölçümlerin niteliğine ve değerlendirilmesine bağlıdır. İyi modeller kurabilmek için ileri düzeyde matematik bilgisine ihtiyaç duyulur. Matematiksel ifadeler geliştikçe gerçek dünyayı anlamak ve anlatmak daha doğru gerçekleştirilir.

4. ROBOT KOLU SİSTEMİNİN MODELLENMESİ VE KONTROLÜ

Robot kolu, dönel veya kayar eklemler ile birbirine bağlanmış “uzuv” adı verilen cisimlerden meydana gelen açık çevrimli bir zincir sistemdir. Çevrimin bir ucu hareketli veya hareketsiz bir desteğe veya temele bağlanmış, diğer ucu ise serbesttir. Eklemler birbirlerine bağlı uzuvların izafi hareketine izin vermektedirler. Bu hareketlerin çözümü için robot kinematiğinden yararlanılır.

Kinematik kavramı robot biliminin temelini oluşturmaktadır. Robot kol kinematiği hesabında farklı yollar izlenmektedir. İzlenen yöntemlerden biri, sabit ağırlıktan başlayarak en uç noktaya ulaşmaya çalışmaktır. Bu işleme ileri kinematik hesaplama yöntemi adı verilir. Diğer bir yöntemde is en uç noktadan başlayarak sabite ulaşmaya çalışılır. Bu yönteme ise ters kinematik hesaplama yöntemi adı verilir.

4.1 İleri Kinematik Analiz

Bir robot kolu ana merkezinden serbest uca doğru birbirine prizmatik veya döner eklemlerle bağlanmış seri uzuvlardan oluşur. İki uzuv arasındaki ilişki bir homojen dönüşüm matrisi ile açıklanır. Eklem dönüşüm matrislerinin ard arda çarpılmasıyla ana merkez ile uç nokta arasındaki ilişki tanımlanır. Kısaca ileri yön kinematiği, eklem değişkenleri ile uç işlevcisinin konumunu ve yönelimini ana merkeze göre hesaplar denilebilir.

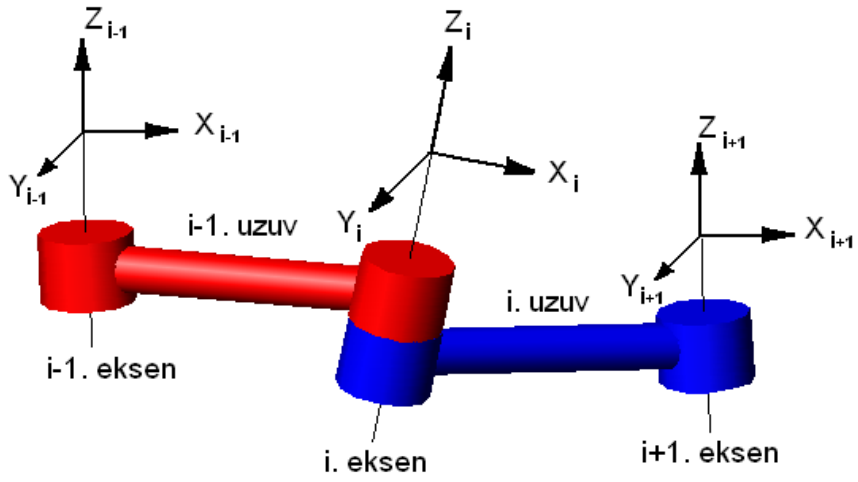
Her bir ekleme bir koordinat sistemi yerleştirilirse komşu iki eklem arasındaki ilişki bir ${}^{i-1}_{i}T$ dönüşüm matrisi ile ifade edilir. İlk ekleme ait dönüşüm matrisi ilk eklem ile ana merkez arasındaki ilişkiyi tanımlarken, son ekleme ait dönüşüm matrisi uç işlevcisi ile son eklem arasındaki ilişkiyi ifade eder. Arka arkaya sıralanan bu eklem dönüşüm matrisleriyle ana merkeze kadar olan ilişkiler tanımlanır. Bu ilişkiye de ileri kinematik denir. Ana merkez ile uç işlevci arasındaki ilişki

$$ {}^{0}_{N}T = {}^{0}_{1}T\,{}^{1}_{2}T \ldots\ldots {}^{N-1}_{N}T \qquad (4.1.1) $$

şeklinde tanımlanır.

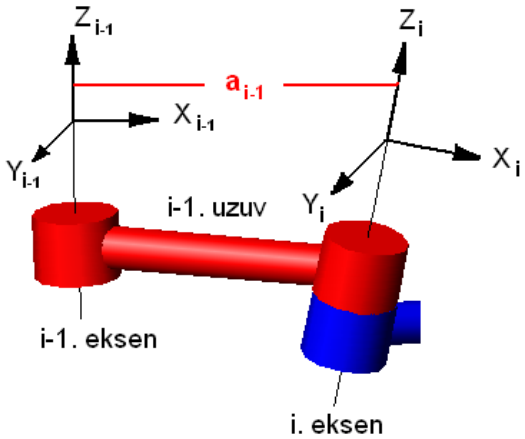
Robot kolundaki eklemlere ait değişkenlerin belirlenmesi için birçok yöntem belirlenmiştir. Ancak en fazla tercih edilen yöntem Denavit-Hartenberg yöntemidir.

Bu yöntemde dört ana değişken kullanılarak robot kinematiği çıkartılır. Bu değişkenler iki eksen arasındaki uzuv uzunluğu a_{i-1}, iki komşu eksen arasındaki eksen açısı α_{i-1}, üst üste çıkan bağlar arasındaki eklem kayması d_i ve iki komşu uzuv arasındaki eklem açısı θ_{i-1}'dir. Bu dört değişken de D-H değişkenidir. Daha sonra bu eksenlerin her birine koordinat sistemi yerleştirilir ve uzuv dönme ekseni koordinat sisteminin Z ekseni olarak kabul edilir.



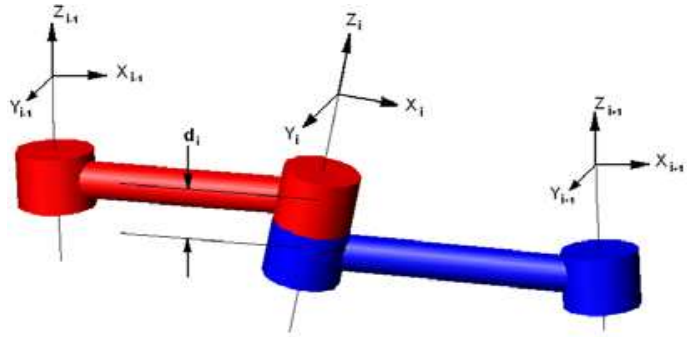
Şekil 4.1Dönme eksenlerin koordinat sisteminin yerleştirilmesi

Aşağıdaki şekilde X_{i-1} yönünde uzanan Z_{i-1} ile Z_i eksenleri arasındaki dik uzaklık yani a_{i-1} uzuv uzaklığı gösterilmiştir.



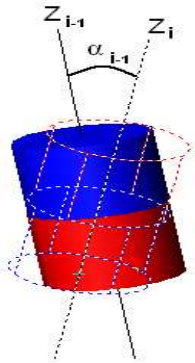
Şekil 4.2 a_{i-1} uzuv uzaklığı

Aşağıdaki şekilde Z_i yönünde uzanan X_{i-1} ile X_i eksenleri arasındaki dik uzaklık yani d_i eklem kayması gösterilmiştir.



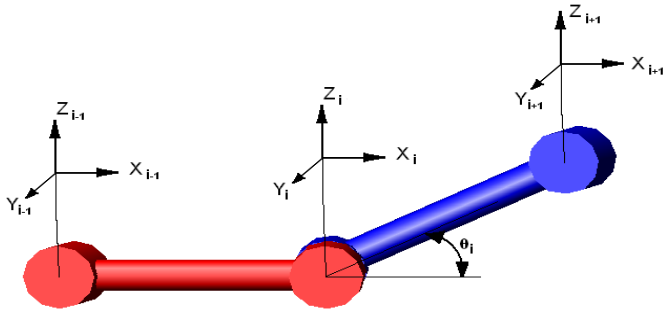
Şekil 4.3 d_i eklem kayması

Aşağıdaki şekilde Z_{i-1} ekseni ile Z_i ekseni arasındaki X_i boyunca ölçülen açı yani α_{i-1} eksen açısı gösterilmiştir.



Şekil 4.4 α_{i-1} eksen açısı

Aşağıdaki şekilde X_{i-1} ekseni ile X_i ekseni arasındaki Z_i boyunca ölçülen açı yani θ_i eklem açısı gösterilmiştir.



Şekil 4.5 θ_i eklem açısı

4.1.1 Koordinat Sistemlerinin Eklemlere Yerleştirilmesi

Koordinat sistemleri eklemlere yerleştirilirken sırasıyla aşağıdaki işlemler gerçekleştirilir. Öncelikle eklem eksenlerinin dönme ve kayma yönleri belirlenerek bu eksene paralel bir doğru çizilir. Eklem eksenleri döner eklemler için dönme yönü Z, prizmatik eklerler için kayma yönü Z ekseni olarak belirlenir. Z eksenine dik ve uzuv boyunca uzanan eksen X ekseni olarak belirlenir. Z ve X eksenleri belirlendikten sonra Y ekseni sağ el kuralına göre belirlenir. Arka arkaya gelen iki eklemin dönme ve kayma yönleri aynı ise Z ekseni belirlendikten sonra kol boyunca X ekseni belirlenir ve sağ el kuralına göre Y ekseni belirlenir. Sıfır ve birinci eksenler üst üste kabul edilir.

Bir robotun eklemlerine koordinat sistemleri yerleştirilirken birinci eksenin dönme yönü Z ekseni olarak belirlendikten sonra bu ekleme X ekseni döndürüldüğünde komşu iki Z ekseni üst üste çakışacak şekilde bir X ekseni yerleştirilir. Koordinat sistemleri eklemlere yerleştirildikten sonra D-H değişkenleri bulunur ve aşağıdaki tabloya yazılır. Robotun hareket etmesiyle değişmeyen parametreler a_{i-1} uzuv uzunlukları ve α_{i-1} eksen açılarıdır. Değişen parametreler ise eklem döner ise θ_{i-1} eklem açısı eğer eklem prizmatik ise d_{i-1} eklem kaçıklığıdır.

Tablo 4.1 D-H Değişkenleri

Eksen no	D-H Değişkenleri				Eklem Değişkeni
i	α_{i-1}	a_{i-1}	d_i	θ_i	d_i veya θ_i
1	α_0	a_0	d_1	θ_1	d_1 veya θ_1
2	α_1	a_1	d_2	θ_2	d_2 veya θ_2
3	α_2	a_2	d_3	θ_3	d_3 veya θ_3

$$
{}^{i-1}_{1}T = R_X(\alpha_{i-1}) D_X(a_{i-1}) R_Z(\theta_i) D_Z(d_i)
$$

$$
= \begin{bmatrix}
\cos\theta_i & -\sin\theta_i & 0 & a_{i-1} \\
\sin\theta_i \cos\alpha_{i-1} & \cos\theta_i \cos\alpha_{i-1} & -\sin\alpha_{i-1} & -\sin\alpha_{i-1} d_i \\
\sin\theta_i \sin\alpha_{i-1} & \cos\theta_i \sin\alpha_{i-1} & \cos\alpha_{i-1} & \cos\alpha_{i-1} d_i \\
0 & 0 & 0 & 1
\end{bmatrix} \qquad (4.1.2)
$$

Her bir ekleme ait genel dönüşüm matrisi elde edilir.

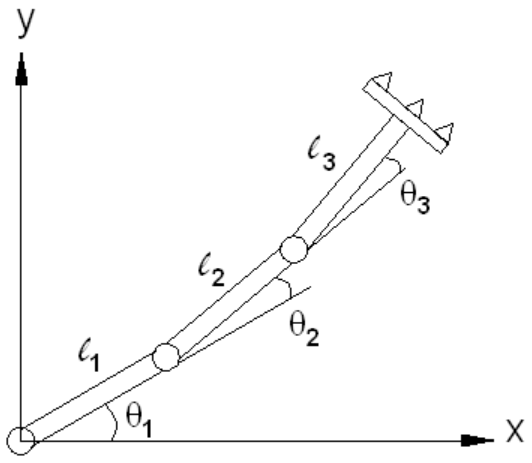
4.1.2 İleri Kinematik Problemlerinin Çözümünde Kullanılan Yaklaşımlar

4.1.2.1 Geometrik Yaklaşım

Bu yaklaşım manipülatör duruşuna bağlı olarak oluşan geometrik şekilden yararlanır.

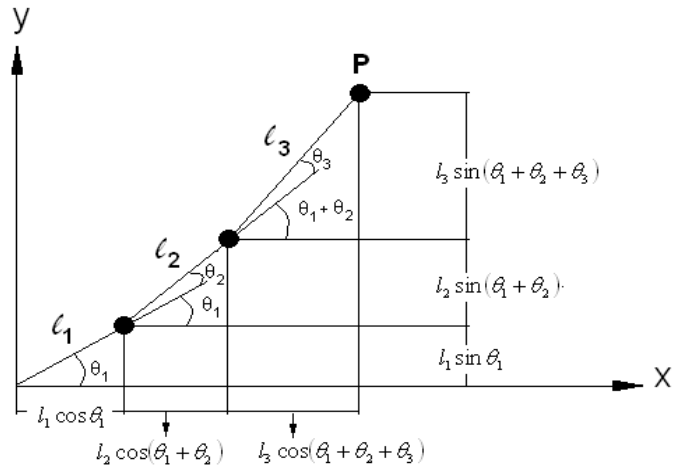
4.1.2.2 Cebirsel Yaklaşım

Bu yaklaşım manipülatörün parametreleri ve eklem değişkenleri arasındaki cebirsel ilişkilerden yararlanır. İki boyutlu düzlemde hareket eden robot kolunun geometrik yaklaşım kullanılarak ileri yön kinematiğinin bulunması şeklinde gerçekleştirilir.



Şekil 4.6 İki boyutlu düzlemde hareket eden robot kolu

Öncelikle robotun iki boyutlu düzlemde aldığı şekil çizilerek uç işlevcisini ifade eden P noktasının konumu bulunur.



Şekil 4.7İki boyutlu düzlemde hareket eden P noktasının konumu

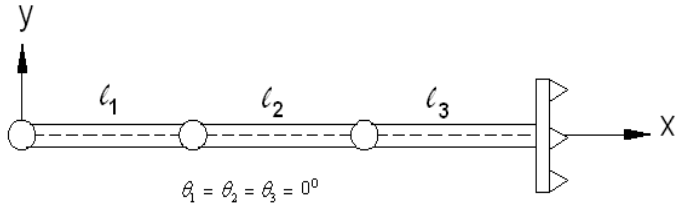
$$P_X = l_1\cos\theta_1 + l_2\cos(\theta_1+\theta_2) + l3\cos(\theta_1+\theta_2+\theta_3) \quad (4.3)$$

P noktasının X Eksenindeki izdüşümü

$$P_Y = l_1\sin\theta_1 + l_2\sin(\theta_1+\theta_2) + l_3\sin(\theta_1+\theta_2+\theta_3) \quad (4.4)$$

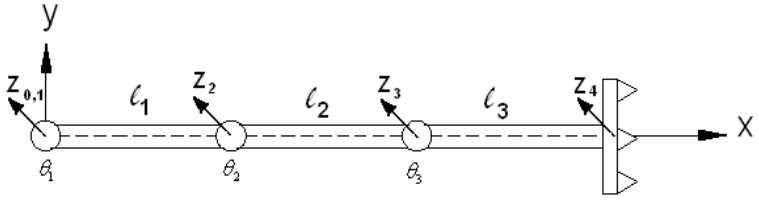
P noktasının Y Eksenindeki izdüşümü

Aynı kolun ileri yön kinematiğini D-H yöntemi ile bulalım. Öncelikle kolun başlangıç değerlerine göre eklemlere koordinat sistemleri yerleştirilir.



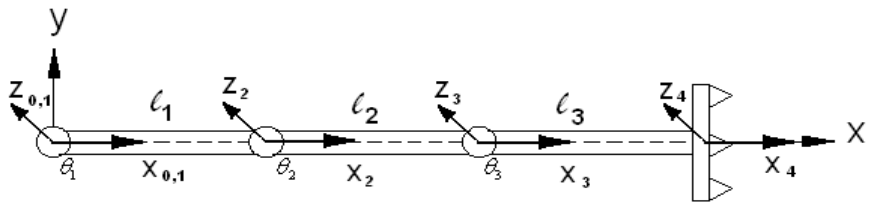
Şekil 4.8 İki boyutlu düzlemde hareket eden kolun başlangıç değerleri

İkinci adımda dönme eklemlerine Z eksenleri yerleştirilir.



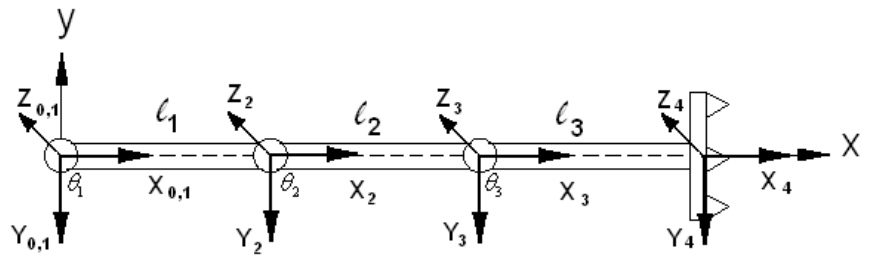
Şekil 4.9 Dönme Eksenlerine Z Eksenlerinin Yerleştirilmesi

Üçüncü adımda Z eksenine dik ve uzuv boyunca uzanan X eksenleri yerleştirilir.



Şekil 4.10 X Eksenlerinin Yerleştirilmesi

Dördüncü adımda Sağ El kuralına göre Y eksenleri yerleştirilir.



Şekil 4.11 Sağ El kuralına göre Y Eksenlerinin Yerleştirilmesi

Son adımda da Koordinat sistemleri eklemlere yerleştirildikten sonra D-H değişkenleri belirlenir ve tabloya eklenir. (Z Ekseni bizden kâğıda doğrudur.) Öncelikle D-H değişkenlerinden sabit olan parametreler belirlenir. $Z_{0,1}, Z_2, Z_3 ve\ Z_4$ eksenlerinin dönme yönleri aynı olduğundan $\alpha_0, \alpha_1, \alpha_2$ ve α_3 açıları 0 ‘dır. Aynı şekilde 0. ve 1. eklemler üst üste olduğundan,

X_0 yönünde uzanan Z_0 ve Z_1 eksenleri arasında $a_0 = 0$

X_1 yönünde uzanan Z_1 ve Z_2 eksenleri arasında $a_1 = l_1$

X_2 yönünde uzanan Z_2 ve Z_3 eksenleri arasında $a_2 = l_2$

X_3 yönünde uzanan Z_3 ve Z_4 eksenleri arasında $a_3 = l_3$

Şimdi ise D-H değişkenlerinden değişen parametrelerini bulalım ancak her bir eklem için yalnızca bir parametre değişken olabilir kısacası döner eklem için eklem açısı, kayar eklem içinse eklem kaçıklığıdır. Örneğimizdeki tüm eklemler döner tip olduğundan d_1, d_2, d_3 ve d_4 eklem kaçıklıkları $0, \theta_1, \theta_2, \theta_3$ ve θ_4 açıları değişkendir.

Tablo 4.2 D-H Değişkenleri

Eksen No	D-H Değişkenleri				Eklem Değişkeni
i	α_{i-1}	a_{i-1}	d_i	θ_i	d_i veya θ_i
1	0	0	0	θ_1	θ_1
2	0	l_1	0	θ_2	θ_2
3	0	l_2	0	θ_3	θ_3
4	0	l_3	0	0	0

Elde edilen tabloya göre her ekleme ait değişkenleri aşağıdaki genel matriste yerine koyarak dönüşüm matrislerini bulalım.

$$ {}^{i-1}_{1}T = \begin{bmatrix} \cos\theta_i & -\sin\theta_i & 0 & a_{i-1} \\ \sin\theta_i \cos\alpha_{i-1} & \cos\theta_i \cos\alpha_{i-1} & -\sin\alpha_{i-1} & -\sin\alpha_{i-1} d_i \\ \sin\theta_i \sin\alpha_{i-1} & \cos\theta_i \sin\alpha_{i-1} & \cos\alpha_{i-1} & \cos\alpha_{i-1} d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (4.5) $$

Birinci eklem için dönüşüm matrisi

$$ {}^{0}_{1}\mathrm{T} = \begin{bmatrix} \cos\theta_1 & -\sin\theta_1 & 0 & a_0 \\ \sin\theta_1 \cos\alpha_0 & \cos\theta_1 \cos\alpha_0 & -\sin\alpha_0 & -\sin\alpha_0 d_1 \\ \sin\theta_1 \sin\alpha_0 & \cos\theta_1 \sin\alpha_0 & \cos\alpha_0 & \cos\alpha_0 d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ = \begin{bmatrix} \cos\theta_1 & -\sin\theta_1 & 0 & 0 \\ \sin\theta_1 & \cos\theta_1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (4.6) $$

İkinci eklem için dönüşüm matrisi

$$
{}^{1}_{2}\mathrm{T} = \begin{bmatrix} \cos\theta_2 & -\sin\theta_2 & 0 & a_1 \\ \sin\theta_2\cos\alpha_1 & \cos\theta_2\cos\alpha_1 & -\sin\alpha_1 & -\sin\alpha_1 d_2 \\ \sin\theta_2\sin\alpha_1 & \cos\theta_2\sin\alpha_1 & \cos\alpha_1 & \cos\alpha_1 d_2 \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

$$
= \begin{bmatrix} \cos\theta_2 & \sin\theta_2 & 0 & l_1 \\ \sin\theta_2 & \cos\theta_2 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (4.7)
$$

Üçüncü eklem için dönüşüm matrisi

$$
{}^{2}_{3}T = \begin{bmatrix} \cos\theta_3 & -\sin\theta_3 & 0 & a_2 \\ \sin\theta_3\cos\alpha_2 & \cos\theta_3\cos\alpha_2 & -\sin\alpha_2 & -\sin\alpha_2 d_3 \\ \sin\theta_3\sin\alpha_2 & \cos\theta_3\sin\alpha_2 & \cos\alpha_2 & \cos\alpha_2 d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

$$
= \begin{bmatrix} \cos\theta_3 & -\sin\theta_3 & 0 & l_2 \\ \sin\theta_3 & \cos\theta_3 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (4.8)
$$

Dördüncü eklem için dönüşüm matrisi

$$
{}^{3}_{4}T = \begin{bmatrix} \cos\theta_4 & -\sin\theta_4 & 0 & a_3 \\ \sin\theta_4\cos\alpha_2 & \cos\theta_4\cos\alpha_3 & -\sin\alpha_3 & -\sin\alpha_3 d_4 \\ \sin\theta_4\sin\alpha_2 & \cos\theta_4\sin\alpha_3 & \cos\alpha_3 & \cos\alpha_3 d_4 \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

$$
= \begin{bmatrix} \cos\theta_4 & -\sin\theta_4 & 0 & l_3 \\ \sin\theta_4 & \cos\theta_4 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & l_3 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} (4.9)
$$

Elde edilen dört ekleme ait dönüşüm matrisleri birbirleriyle çarpılarak manipülatöre ait dönüşüm matrisi elde edilir.

$$ {}^{0}_{4}T = {}^{0}_{1}T\,{}^{1}_{2}T\,{}^{2}_{3}T\,{}^{3}_{4}T \qquad (4.10) $$

$$ = \begin{bmatrix} \cos\theta_1 & -\sin\theta_1 & 0 & 0 \\ \sin\theta_1 & \cos\theta_1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta_2 & -\sin\theta_2 & 0 & l_1 \\ \sin\theta_2 & \cos\theta_2 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta_3 & -\sin\theta_3 & 0 & l_2 \\ \sin\theta_3 & \cos\theta_3 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & l_3 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (4.11) $$

elde edilen dönüşüm matrisinin konum vektöründen,

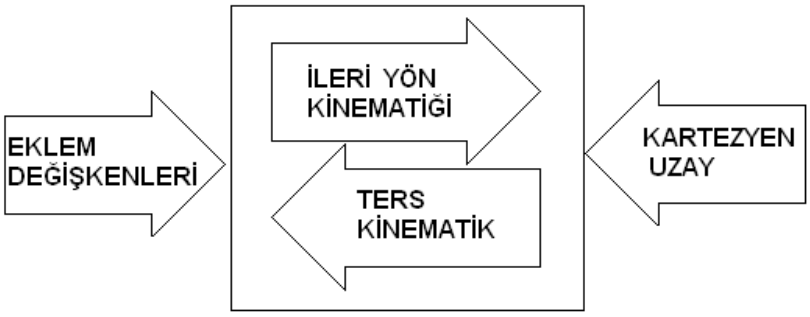
$$ P_x = l_3\cos(\theta_1+\theta_2+\theta_3) + l_2\cos(\theta_1+\theta_2) + l_1\cos\theta_1 \qquad (4.12) $$

$$ P_y = l_3\sin(\theta_1+\theta_2+\theta_3) + l_2\sin(\theta_1+\theta_2) + l_1\sin\theta_1 \qquad (4.13) $$

Denklemleri elde edilir.

4.2 Ters Kinematik Analiz

Robotun uç işlevcisinin ana çerçeveye göre konumu ve yönelimi verildiğinde manipülatörün bu konuma ve yönelime gelebilmesi için gerekli eklem değişkenlerinin bulunmasıdır. Kısacası uç işlevcisinin konum ve yönelim verileriyle eklem değişkenlerinin bulunması olarak da tanımlanabilir. Başka bir değişle de uç işlevcisinin konum ve yönelimini Kartezyen koordinat sisteminden eklem koordinat sistemine dönüştürme işlemi olarak da tanımlayabiliriz.

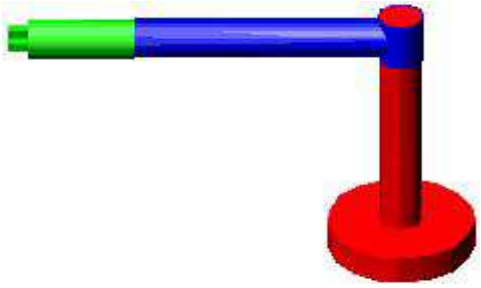


Şekil 4.12 Eklem Değişkenleri ile Kartezyen Uzay Dönüşümü

Ters kinematik aşağıdaki nedenlerden dolayı çözülmesi oldukça zor olan problemler içerir.

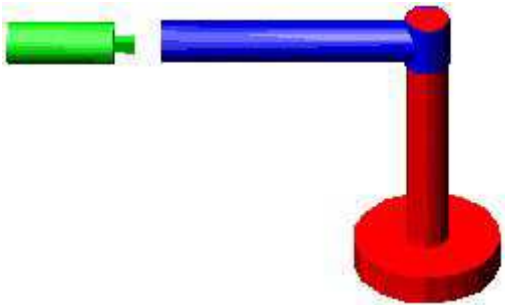
1. Analitik olarak karmaşık, doğrusal olmayan denklemler içerir.
2. Eklemlerin yapısına bağıdır. Eğer robot prizmatik eklemlerden oluşuyorsa ters kinematik problemin çözümü kolaylaşırken, robottaki döner eklem sayısı arttıkça problemin çözümü de o derece zorlaşmaktadır.
3. Her zaman matematiksel çözüm fiziksel çözümü temsil etmez.

Şekil 4.13 ‘de matematiksel çözümle fiziksel çözüm örtüşürken ikinci Şekil 4.14’de örtüşmez.



θ =arctan2(−k, p_z)

Şekil 4.13 Fiziksel Çözüm



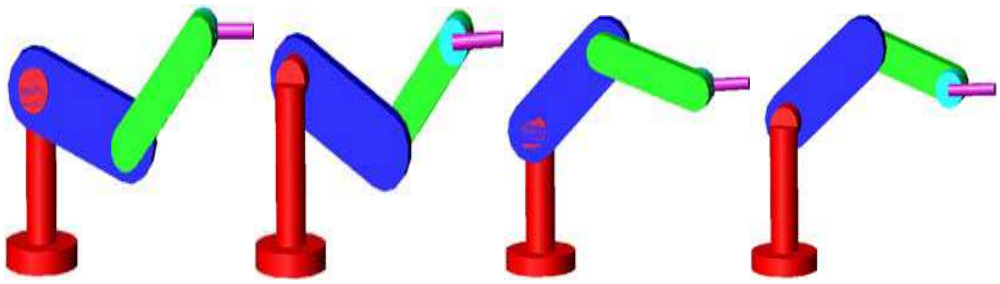
θ = arctan 2(k,−p)

Şekil 4.14 Matematiksel çözüm

Aynı uç işlevci düzenleşimi için birden fazla çözüm olabilir. Ters kinematik çözüm sayısı robotun serbestlik derecesinin yanında aynı zamanda eklem değişkenlerine de

bağlıdır. Her bir eklemde uzuv uzunluğu ve eklem kaçıklığının olması çözüm sayısının artmasına neden olur. Örneğin 6R robotta her bir eklem için en azından bir uzuv uzunluğu ve eklem kaçaklığı olduğundan ters kinematik çözüm sayısı $2^6 = 64$ 'dür. Yalnız bu çözümlerin bir kısmı gerçek bir kısmı ise sanaldır.

Dönel eklemlerden oluşan robotlarda fiziksel çözüm sayısın fazla olması, üç boyutlu uzayda bir noktaya birkaç şekilde ulaşma imkânı sağlar. Örneğim KUKA robotunun aynı noktaya farklı eklem açıları ile erişebildiğini gösterir.



Şekil 4.15 Robotun Aynı Noktaya Dört Farklı Şekilde Ulaşması

Ters kinematik problem verilen bir robot düzenleşimi için tamamen analitik olarak çözülebileceği gibi, analitik çözüm mümkün olmadığı durumlarda da sayısal yöntemler kullanılarak da çözülebilir. Tamamıyla kesin sonuç üreten analitik çözüme ait denklemler bilgisayara ortamında çok hızlı çalışırken, eklem açılarının iteratif olarak çözüldüğü sayısal çözüm ise bilgisayar ortamında analitik yönteme göre yavaş çalışır. Bu nedenle robot tasarımcıları tasarımı basit analitik çözümü mümkün olana robot tasarımı üzerinde durmuşlardır.

Sayısal çözüm için kinematik eşitliklerin farklarının toplamını alan tahmin edici ve düzeltici tip algoritmalar kullanılır. Ters kinematik problemin sayısal yöntemlerle çözümlenmesinde karşılaşılan en büyük sorun, Jakobleyn matrisinin tekil olduğu noktalarda yazılan algoritmanın çözüm üretmemesidir. Ayrıca başlangıç çözüm vektörü çözüme yönelik elemanlardan oluşmadığı zaman sayısal çözüm kararlı bir çözüm üretmez. Tamamıyla sayısal veya analitik çözüm gerçekleşmediği durumlarda eklem değişkenlerinden bir kaçı analitik olarak bulunup diğerleri sayısal olarak çözülür.

4.2.1 Ters Kinematik Problemlerine Analitik Çözüm Yaklaşımı

Craig[9] tarafından tanımlanan altı serbestlik derecesine sahip bir robotun ileri yön kinematiği aşağıdaki gibi yazılır.

$$ {}^{0}_{6}T = {}^{0}_{1}T\,{}^{1}_{2}T\,{}^{2}_{3}T\,{}^{3}_{4}T\,{}^{4}_{5}T\,{}^{5}_{6}T \qquad (4.13) $$

${}^{0}_{6}T$ İleri yön kinematiği matrisi, konum ve yönelim verilerini içeren matris elemanları cinsinden Denklem 4.27'deki gibi yazılıp Denklem 4.28 teki eşitlik sağlanır.

Denklem 4.27'nin 1. kolonu uç işlevcisinin normal vektörünü, 2. Kolonu kayma ve 3. kolonu ise yalaşım vektörünü göstermektedir.

$$ {}^{0}_{6}T = \begin{bmatrix} r_{11} & r_{12} & r_{13} & \mathrm{p}_X \\ r_{21} & r_{22} & r_{23} & p_Y \\ r_{31} & r_{32} & r_{33} & p_Z \\ 0 & 0 & 0 & 0 \end{bmatrix} \qquad (4.14) $$

$$ \left[{}^{0}_{6}T\right]^{-1}\,{}^{0}_{6}T = \left[{}^{0}_{1}T\right]^{-1}\,{}^{0}_{1}T\,{}^{1}_{2}T\,{}^{2}_{3}T\,{}^{3}_{4}T\,{}^{4}_{5}T\,{}^{5}_{6}T \qquad (4.15) $$

${}^{0}_{1}T^{-1}\,{}^{0}_{1}T = I$ Olduğundan yukarıdaki denklem daha aşağıdaki gibi basit bir ifadeyle elde edilebilir.

$$ \left[{}^{0}_{1}T\right]^{-1}\,{}^{0}_{6}T = {}^{1}_{2}T\,{}^{2}_{3}T\,{}^{3}_{4}T\,{}^{4}_{5}T\,{}^{5}_{6}T \qquad (4.16) $$

$$ \left[{}^{0}_{1}T\,{}^{1}_{2}T\right]^{-1}\,{}^{0}_{6}T = {}^{2}_{3}T\,{}^{3}_{4}T\,{}^{4}_{5}T\,{}^{5}_{6}T \qquad (4.17) $$

$$ \left[{}^{0}_{1}T\,{}^{1}_{2}T\,{}^{2}_{3}T\right]^{-1}\,{}^{0}_{6}T = {}^{3}_{4}T\,{}^{4}_{5}T\,{}^{5}_{6}T \qquad (4.18) $$

$$ \left[{}^{0}_{1}T\,{}^{1}_{2}T\,{}^{2}_{3}T\,{}^{3}_{4}T\right]^{-1}\,{}^{0}_{6}T = {}^{4}_{5}T\,{}^{5}_{6}T \qquad (4.19) $$

$$\left[{}^{0}_{1}T\,{}^{1}_{2}T\,{}^{2}_{3}T\,{}^{3}_{4}T\,{}^{4}_{5}T \right]^{-1}\,{}^{0}_{6}T = {}^{5}_{6}T \tag{4.20}$$

Ters kinematik çözüm gerçekleştirilirken kullanılan bazı trigonometrik eşitlikler.

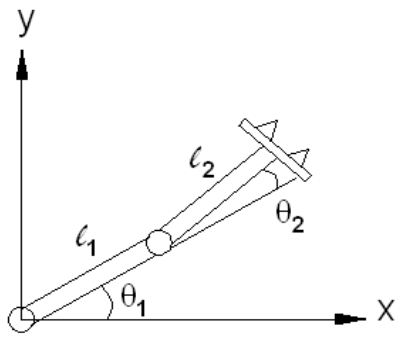
$\cos\theta = a$ ise $\theta = \arctan2\left(\pm\sqrt{1-a^2}, a\right)$ (4.21)

$\sin\theta = a$ ise $\theta = \arctan2\left(a, \pm\sqrt{1-a^2}\right)$ (4.22)

$\cos\theta = a$ ve $\sin\theta = b$ ise $\theta = \arctan2(b, a)$ (4.23)

$\sin\theta + b\cos\theta = 0$ ise (4.24)

$\theta = \arctan2(a,b) + \theta = \arctan2\left(\pm\sqrt{a^2+b^2-c^2}, c\right)$ (4.25)



Şekil 4.16İki Boyutlu Düzlemde Hareket Eden Robot Kolu

$$ {}^{0}_{1}T = \begin{bmatrix} \cos\theta_1 & -\sin\theta_1 & 0 & 0 \\ \sin\theta_1 & \cos\theta_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.26}$$

$$ {}^{1}_{2}T = \begin{bmatrix} \cos\theta_2 & -\sin\theta_2 & 0 & I_2 \\ \sin\theta_2 & \cos\theta_2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.27}$$

$$ {}^{2}_{3}T = \begin{bmatrix} 1 & 0 & 0 & I_2 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.28} $$

$$ {}^{0}_{3}T = {}^{0}_{1}T\,{}^{1}_{2}T\,{}^{2}_{3}T \tag{4.29} $$

İleri yön kinematiğine ait dönüşüm matrisinin her iki tarafını $\left[{}^{0}_{1}T\right]^{-1}$ ile çarpalım.

$$ \left[{}^{0}_{3}T\right]^{-1}\,{}^{0}_{3}T = \left[{}^{0}_{1}T\right]^{-1}\,{}^{0}_{1}T\,{}^{1}_{2}T\,{}^{2}_{3}T \tag{4.30} $$

Bilindiği gibi $\left[{}^{0}_{1}T\right]^{-1}\,{}^{0}_{1}T = I$ olduğundan denklem aşağıdaki gibi olur.

$$ \left[{}^{0}_{1}T\right]^{-1}\,{}^{0}_{3}T = {}^{1}_{2}T\,{}^{2}_{3}T \tag{4.31} $$

$$ {}^{0}_{1}T^{-1} = \begin{bmatrix} {}^{0}_{1}R^{T} & -{}^{0}_{1}R^{T} & {}^{0}P_1 \\ 0 \quad 0 & 0 \quad 1 & \end{bmatrix} \tag{4.32} $$

$$ {}^{0}_{1}R = \begin{bmatrix} \cos\theta_1 & -\sin\theta_1 & 0 \\ \sin\theta_1 & \cos\theta_1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad {}^{0}_{1}R^{T} = \begin{bmatrix} \cos\theta_1 & \sin\theta_1 & 0 \\ -\sin\theta_1 & \cos\theta_1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.33} $$

${}^{0}_{1}T$ Dönüşüm matrisinin konum vektörü sıfır olduğundan

$$ -{}^{0}_{1}R^{T}\,{}^{0}P_1 = \begin{bmatrix} \cos\theta_1 & \sin\theta_1 & 0 \\ -\sin\theta_1 & \cos\theta_1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{4.34} $$

elde edilir.

Bu matrisi ${}^{0}_{3}T$ ileri kinematiği temsil eden matrisle çarpalım

$$ {}^{0}_{3}T = \begin{bmatrix} r_{11} & r_{12} & r_{13} & p_x \\ r_{21} & r_{22} & r_{23} & p_y \\ r_{31} & r_{32} & r_{33} & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.35} $$

$$ {}^{0}_{1}T^{-1}\,{}^{0}_{3}T = \begin{bmatrix} \cos\theta_1 & \sin\theta_1 & 0 & 0 \\ -\sin\theta_1 & \cos\theta_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & p_x \\ r_{21} & r_{22} & r_{23} & p_y \\ r_{31} & r_{32} & r_{33} & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ = \begin{bmatrix} \cos\theta_1 r_{11} + \sin\theta_1 r_{21} & \cos\theta_1 r_{12} + \sin\theta_1 r_{22} & \cos\theta_1 r_{13} + \sin\theta_1 r_{23} & \cos\theta_1 p_x + \sin\theta_1 r_y \\ -\sin\theta_1 r_{11} + \cos\theta_1 r_{21} & -\sin\theta_1 r_{22} + \cos\theta_1 r_{22} & -\sin\theta_1 r_{13} + \cos\theta_1 r_{23} & -\sin\theta_1 p_x + \cos\theta_1 p_x \\ r_{31} & r_{32} & r_{33} & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.36} $$

$\left[{}^{0}_{1}T\right]^{-1}\,{}^{0}_{3}T$ Çarpımını bulduk. Şimdide ${}^{1}_{2}T\,{}^{2}_{3}T$ çarpımını bulalım ve iki sonucu birbirine eşitleyelim.

$$ {}^{1}_{2}T\,{}^{2}_{3}T = \begin{bmatrix} \cos\theta_2 & -\sin\theta_2 & 0 & I_1 \\ \sin\theta_2 & \cos\theta_2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & I_2 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \cos\theta_2 & -\sin\theta_2 & 0 & I_2\cos\theta_2 + I_1 \\ \sin\theta_2 & \cos\theta_2 & 0 & I_2\cos\theta_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.37} $$

Bulunur.

$$ \begin{bmatrix} \cos\theta_1 r_1 + \sin\theta_1 r_{21} & \cos\theta_1 r_{12} + \sin\theta_1 r_{22} & \cos\theta_1 r_{13} + \sin\theta_1 r_{23} & \cos\theta_1 p_x + \sin\theta_1 p_y \\ -\sin\theta_1 r_{11} + \cos\theta_1 r_{21} & -\sin\theta_1 r_{12} + \cos\theta_1 r_{22} & -\sin\theta_1 r_{13} + \cos\theta_1 r_{23} & -\sin\theta_1 p_x + \cos\theta_1 p_x \\ r_{31} & r_{32} & r_{33} & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.38} $$

$$ = \begin{bmatrix} \cos\theta & -\sin\theta_2 & 0 & I_2\cos\theta_2 + I_1 \\ \sin\theta_2 & \cos\theta_2 & 0 & I_2\cos\theta_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.39} $$

1. $\cos\theta_1 r_{11} + \sin\theta_1 r_{21} = \cos\theta_2$ (4.40)

2. $-\sin\theta_1 r_{11} + \cos\theta_1 r_{21} = \sin\theta_2$ (4.41)

3. $r_{31} = 0$ (4.42)

4. $\cos\theta_1 r_{12} + \sin\theta_1 r_{22} = -\sin\theta_2$ (4.43)

5. $-\sin\theta_1 r_{12} + \cos\theta_1 r_{22} = \cos\theta_2$ (4.44)

6. $r_{32} = 0$ (4.45)

7. $\cos\theta_1 r_{13} + \sin\theta_1 r_{23} = 0$ (4.46)

8. $-\sin\theta_1 r3 + \cos\theta_1 r_{23} = 0$ (4.47)

9. $r_{33} = 0$ (4.48)

10. $\cos\theta_1 P_x + \sin\theta_1 P_y = I_2\cos\theta_2 + I_1$ (4.49)

11. $-\sin\theta_1 P_x + \cos\theta_1 P_y = I_2\sin\theta_2$ (4.50)

12. $P_z = 0$ (4.51)

Eşitlikleri elde edilir.

Ters kinematik çözüm tamamı mümkünse kol uzunlukları cinsinden elde edilmelidir. Bu nedenle 10. Ve 11. Denklemlerin her iki tarafının karesini alıp alt alta toplayalım.

$$\cos^2\theta_1 P_x^2 + \sin^2\theta_1 P_y^2 = I_2^2\cos^2\theta_2 + 2I_1^2 I_2^2\cos^2\theta_2 + I_1^2$$

$$+ \quad (-\sin\theta_1)^2 P_x^2 + \cos^2\theta_1 P_y^2 = I_2^2\sin^2\theta_2$$

$$\cos^2\theta_1 P_x^2 + \sin^2\theta_1 P_y^2 + \sin\theta_1^2 P_x^2 + \cos^2\theta_1 P_y^2 = I_2^2\cos^2\theta_2 + 2I_1 I_2\cos\theta_2 + I_1^2 + I_2^2\sin^2\theta_2 \quad (4.52)$$

Eşitliğin sol tarafını P_x^2 ve P_y^2, sağ tarafını da I_2^2 parantezini alalım.

$$\left(\cos^2\theta_1 + \sin\theta_1^2\right)P_x^2 + \left(\sin^2\theta_1 + \cos^2\theta_1\right)P_y^2 = I_2^2\left(\cos^2\theta_2 + \sin^2\theta_2\right) + 2I_1 I_2\cos\theta_2 + I_1^2 \quad (4.53)$$

olur.

$$\left(\cos^2\theta_1 + \sin\theta_1^2\right) = 1 \quad (4.54)$$

'dir.

Bu durumda denklemi yeniden yazarsak

$$P_x^2 + P_y^2 = I_2^2 + 2I_1 I_2\cos\theta_2 + I_1^2 \quad (4.55)$$

olur.

Buradan

$$\cos\theta_2 = \left(\frac{P_x^2 + P_Y^2 - I_2^2 + I_1^2}{2I_1 I_2} \right) \tag{4.56}$$

bulunur.

Bunu $\cos\theta_2 = a$ ise $\theta = \arctan 2\left(\pm\sqrt{1-a^2}, a\right)$ denklemine uyarlarsak

$$\theta_2 = \arctan 2\left(\pm\sqrt{1-\left(\frac{P_x^2 + P_y^2 - I_2^2 + I_1^2}{2I_1 I_2} \right)^2} \cdot \left(\frac{P_X^2 + P_Y^2 - I_2^2 + I_1^2}{2I_1 I_2} \right) \right) \tag{4.57}$$

bulunur.

$$\cos\theta_1 P_x + \sin\theta_1 P_y = I_2 \cos\theta_2 + I_1 \tag{4.58}$$

Bu ifadeyi de $a\sin\theta + b\cos = c$ ‘e uyarlasak.

$$P_x = a., P_y = b, I_2 \cos\theta_2 + I_1 = c \tag{4.59}$$

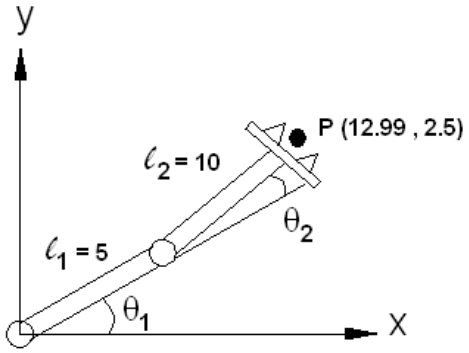
olur.

$$\theta = \arctan 2(a,b) \pm \theta = \arctan 2\left(\sqrt{a^2 + b^2 - c^2}, c\right) \tag{4.60}$$

$$\theta_1 = \arctan 2\left(P_y, P_x\right) \pm \theta = \arctan 2\left(\sqrt{P_y^2 + P_x^2 - \left(I_2\cos\theta_2 + I_1\right)^2}, \left(I_2\cos\theta_2 + I_1\right) \right) \tag{4.61}$$

elde edilir.

Bir robotun kol uzunlukları $I_1 = 10$ ve $I_2 = 5$ verilsin. Robotun uç işlevcisinin kartezyen uzayındaki konumu $P_x = 12{,}99, P_y = 2{,}5$ olması için birinci eklem θ_1 ve ikinci eklem θ_2 değişkenlerini hesaplayalım.



Şekil 4.17İki boyutlu düzlemde hareket eden P noktasının konumu

$$\theta_2 = \arctan 2\left(\pm\sqrt{1-\left(\frac{P_x^2+P_y^2-I_2^2+I_1^2}{2I_1I_2}\right)^2} \;,\left(\frac{P_x^2+P_y^2-I_2^2+I_1^2}{2I_1I_2}\right)\right)$$

$$= \arctan 2\left(\pm\sqrt{1-\left(\frac{12{,}99^2+2{,}5^2-5^2+10}{2.5.10}\right)^2} \;,\left(\frac{12{,}99^2+2{,}5^2-5^2+10^2}{2.5.10}\right)\right) \qquad (4.62)$$

$$= \arctan 2\left(\pm\sqrt{1-(0{,}4999)^2}\,,(0{,}4999)\right) = \pm 60^\circ \qquad (4.63)$$

$\theta_2 = \pm 60^\circ$ Bulunur. Kosinüs çift özelliğe sahip olduğu için negatif ve pozitif değeri eşittir.

$$\theta_1 = \arctan 2\left(P_{y,}P_x\right) \pm \theta = \arctan 2\left(\sqrt{P_y^2+P_x-\left(I_2\cos\theta_2+I_1\right)^2} \;,\left(I_2\cos\theta_2+I_1\right)\right)$$

$$= \arctan 2(2.5{,}12.99) \pm \theta = \arctan 2\left(\sqrt{2{,}5^2+12{,}99-\left(5\cos 60+10\right)^2}\,,\left(5\cos 60+10\right)\right)$$

$$= 10{,}9 \pm 19{,}1 \qquad (4.64)$$

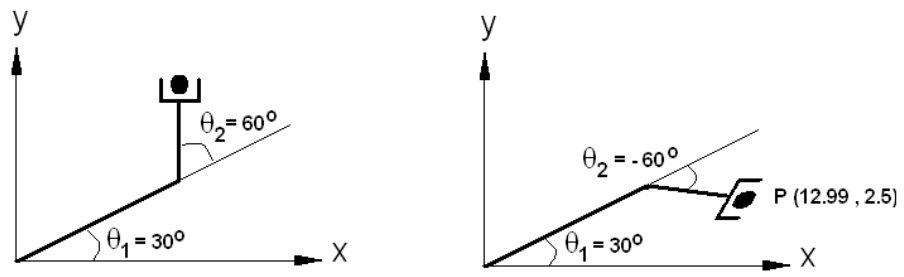
$$Ç_1 = \{\theta_1 = 10{,}9+19{,}1 = 30^\circ \quad \text{ve}\,\theta_2 = +60\} \qquad (4.65)$$

$$Ç_2 = \{\theta_1 = 10{,}9+19{,}1 = 30^\circ \,\text{ve}\,\theta_2 = -60^\circ\} \qquad (4.66)$$

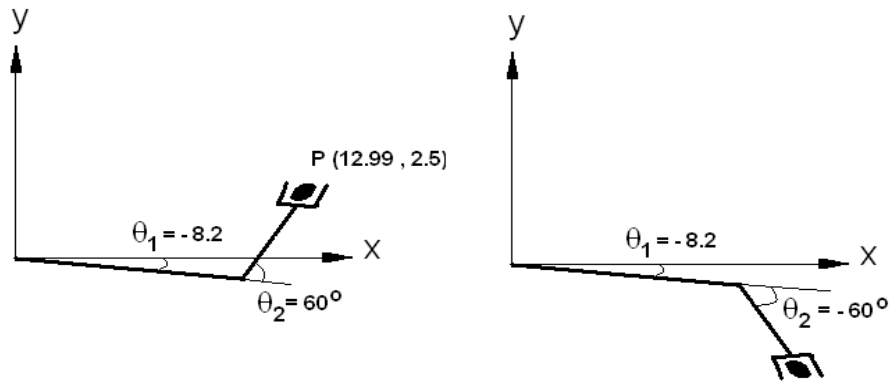
$$Ç_3 = \{\theta_1 = 10{,}9-19{,}1 = -8{,}2^\circ \,\text{ve}\,\theta_2 = +60^\circ\} \qquad (4.67)$$

$$Ç_4 = \{\theta_1 = 10{,}9-19{,}1 = -8{,}2^\circ \,\text{ve}\,\theta_2 = -60^\circ\} \qquad (4.68)$$

Görüldüğü gibi ters kinematik probleminin 4 tane çözüm kümesi bulundu. 1. ve 4.çözüm kümeleri uç işlevcisini hedefe ulaştıramazlar o halde 4 çözümün sadece iki tanesi gerçek çözümdür.



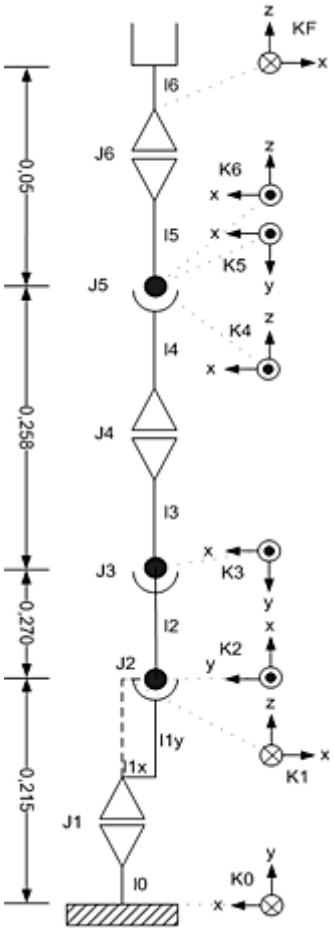
Şekil 4.18 Ters Kinematik Probleminin Gerçek Çözümlerine Ait Konumları



Şekil 4.19 Ters Kinematik Probleminin Sanal Çözümlerine Ait Konumları

5. KUKA ROBOT KOLU MODELİ

5.1Denavit-Hartenberg Parametreleri



Tablo 5.1 Denavit-Hartenberg Parametreleri

Eklemler	Dönüş Açısı α_{i-1}	Bağlantı Uzunluğu a_{i-1}	Eklem Açısı ϕ_i	Bağlantı Ofseti d_i
1	-90^o	**0**	$\Theta_1 + 180^0$	$l_0 + l_{1,y}$
2	90^o	$l_{1,x}$	$\Theta_2 + 90^0$	**0**
3	0^o	l_2	$\Theta_3 + 180^0$	**0**
4	90^o	**0**	Θ_4	$l_3 + l_4$
5	-90^o	**0**	Θ_5	**0**
6	90^o	**0**	Θ_6	**0**
7	0^o	**0**	180^0	l_5

Transformasyon yani dönüşüm matrisimiz,

$$ {}^{i}\mathbf{T}_{j} = \prod_{n=i+1}^{j} {}^{n-1}\mathbf{D}_{n}\left(\alpha_{n-1}, a_{n-1}, \theta_{n}, d_{n}\right), \quad i < j \tag{5.1} $$

Matrisimizi oluşturduğumuz kodlarımız aşağıda verilmiştir,

```
PublicstaticMatrixDonüsüm_Matrisi_Al(intlow_index, inthigh_indeks,
float[] teta, float[] alfa, float[] a, float[] d){

for(int i=low_indeks+1;i<high_indeks+1;i++){
values= newfloat[4,4];
values[0,0]=(float)(Math.Cos(teta(i)));
values[0,1]=(float)(Math.Sin(teta(i)));
values[0,2]=0;
values[0,3]=a[i-1];
values[1,0]=(float)(Math.Cos(alfa[i-1])*Math.Sin(teta[i]));
values[1,1]=(float)(Math.Cos(alfa[i-1])*Math.Cos(teta[i]));
values[1,2]= -(float)(Math.Sin(alfa(i-1)));
values[1,3]= -(float)(d[i]*Math.Sin(alfa[i-1]));
values[2,0]=(float)(Math.Sin(alfa[i-1])*Math.Sin(teta[i]));
values[2,1]=(float)(Math.Sin(alfa[i-1])*Math.Cos(teta[i]));
values[2,2]= (float)(Math.Cos(alfa(i-1)));
values[2,3]= (float)(d[i]*Math.Cos(alfa[i-1]));
values[3,0]=0;
values[3,1]=0;
values[3,2]=0;
values[3,3]=1;
matrisler[i-(low_indeks+1)]=newMatris(values) ;
}
```

5.2 İleri Kinematik Hesaplamaları

Kuka robot koluna ait geometrik yaklaşım kullanılarak yapılan ileri kinematik hesaplamalar verilmiştir.

$^{0}T_{6}$:

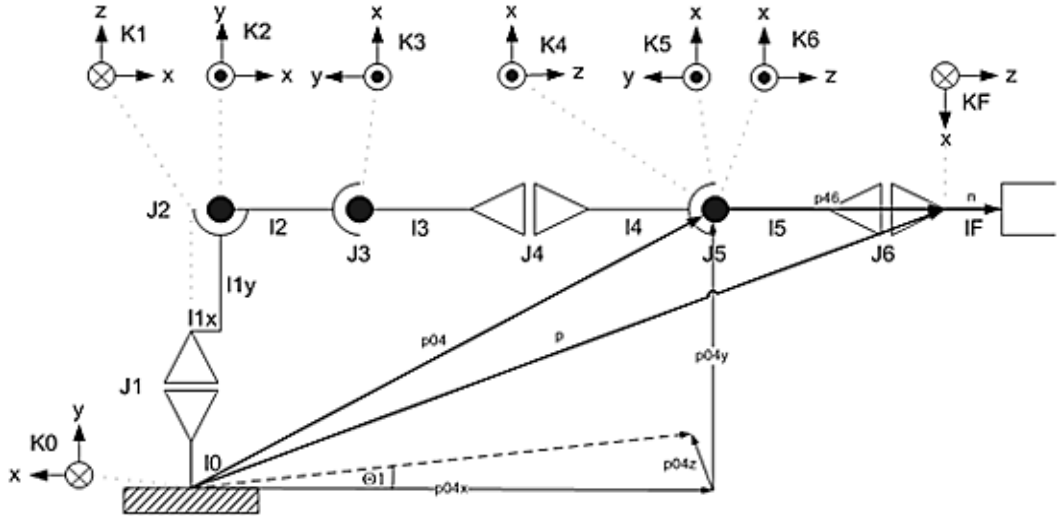
$$\vec{P}_{04} = \vec{P} - \vec{P}_{46} = \vec{P} - (l_5 + l_F) * \vec{n} \quad (5.2)$$

p04KO[0]=(float)(Math.Round(x-state.D[7]*n[0],5));

p04KO[1]=(float)(Math.Round(y-state.D[7]*n[1],5));

p04KO[2]=(float)(Math.Round(z-state.D[7]*n[2],5));

p04KO[3]=1;



Şekil 5.1 1.Eklem Açısı

$^{0}T_{1}$:

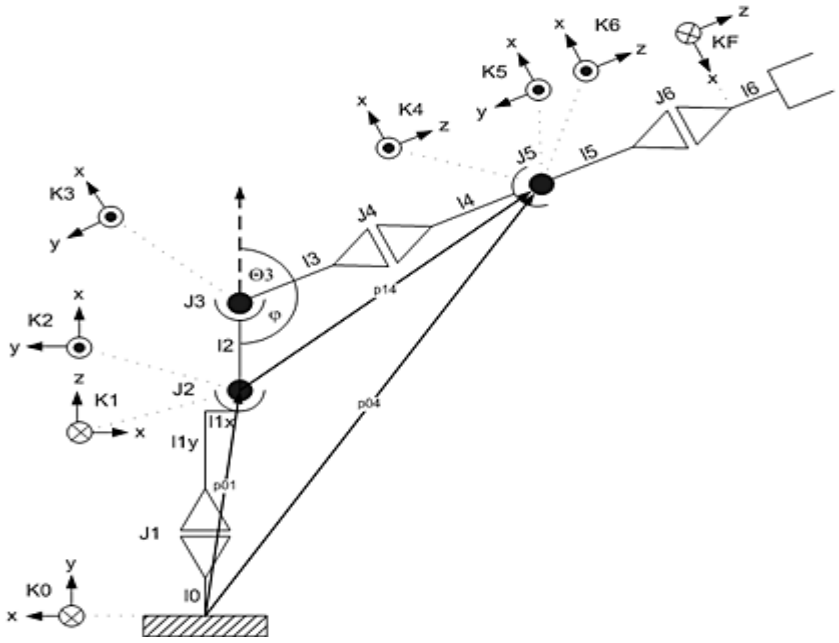
$$\vec{P}_{14} = \vec{P}_{04} - \vec{P}_{01} \quad (5.3)$$

p14KO[0]=(float)(p04KO[0]-T01.Values[0,3]);

p14KO[1]=(float)(p04KO[1]-T01.Values[1,3]);

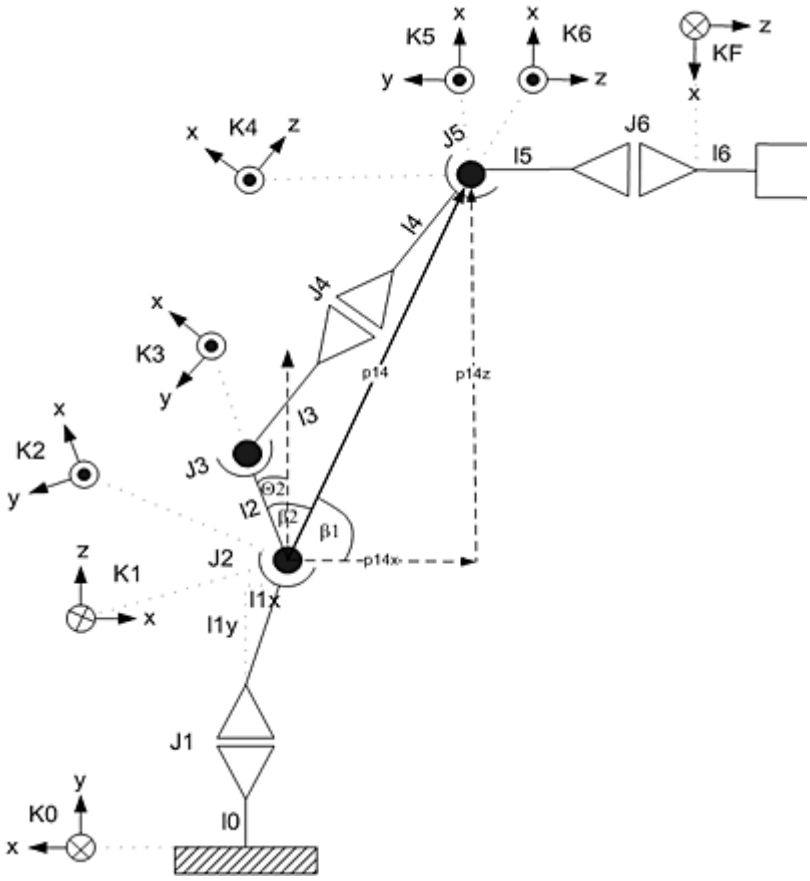
p14KO[2]=(float)(p04KO[2]-T01.Values[2,3]);

p14KO[3]=1;



Şekil 5.23.Eklem Açısı

```
T01.Values[0,3]=0;
T01.Values[1,3]=0;
T01.Values[2,3]=0;
float[] p14K1=T01.transpose().times(p14KO);
```



Şekil 5.32. Eklem Açısı

$$^{0}T_{4} = \begin{pmatrix} ^{0}R_{4} & \vec{P} \\ \vec{0}^{T} & 1 \end{pmatrix} \quad (5.4)$$

$$^{0}R_{4} = (\vec{x}_4 \vec{y}_4 \vec{z}_4) = {}^{0}R_{1}(\Theta_1) * {}^{1}R_{2}(\Theta_2) * {}^{3}R_{3}(\Theta_3) * {}^{4}R_{4}(\Theta_4) \quad (5.5)$$

6. KOD PLATFORMU

Simülasyon bir veya daha fazla fiziksel nesneyi ifade etmek için kullanılan bir matematiksel bilgisayar modelidir. Bir simülasyonun en büyük faydası nesnenin kendisine ihtiyaç duymadan modellenen bilgisayar simülasyonu üzerinden istenilen sonuçların elde edilebilir olmasıdır. Sonuçların doğruluğu oluşturulan simülasyonun doğru modellenmesine bağlı olduğu gibi aynı zamanda gerçek zamanda sonuçları elde etmek için harcanan zamandan daha kısa sürede sonuç verebilmesi de diğer bir getirisidir.

Endüstriyel bir robot kolu tasarımında robot kolunun fiziksel olarak hareket ve davranışları için oluşturulan yazılımın sonuçlarının elde edilmesi sistemin aslına uygulanarak üretilebileceği gibi, tasarlanan sistemin simülasyonu üzerinden de elde edilebilir. Bu sayede maliyet azalmış, istenmeyen sonuçların doğuracağı problemlerden uzak bir geliştirme ortamı hazırlanmış ve sonuçlara daha hızlı gözlem ve analiz yapılabilir olmaktadır.

Simülasyon bilgisayar ortamında gerçeklendiği üzere bir yazılım sonucu oluşturulmuş olması gerekmektedir. Yazılım ortamı programcının tercihi olmakla beraber kolay simüle edilebilirliği ve uygulamanın kullanabilirliği tercih edilmektedir. Çalışmada Microsoft Robotic Developer Studio (MSRS) simülasyon ortamı ve Microsoft Visual Programming Language (MVPL) yazılımı kullanılacaktır.

MSRS ve MVPL basit ve karmaşık birçok robot uygulaması geliştirebileceğimiz grafiksel uygulama tasarım araçları sunan bir platform ve dildir. Sürükle bırak programlama tekniğini kullanarak ara yüze ait bir çok araçtan faydalanabilir ve giriş/çıkış elemanları arası bağlantıları oluşturabiliriz. Elemanlar oldukça basit veri değerleri veya karmaşık servisler olarak kullanılabilir.

MSRS, akademik, hobi amaçlı ve ticari geliştiricilerin çok çeşitli donanım platformlarında kolay bir şekilde robotik tabanlı uygulamalar geliştirmeleri için Windows tabanlı geliştirme ortamıdır.

Robot uygulamaları kodlama işine oldukça büyük kolaylıklar getirmektedir. Fiziksel donanımdan bağımsız olarak simülasyon geliştirilebilmekte ve bu simülasyonları fiziksel donanıma uygulayabilmektedir. Kodlama kısmında eşzamanlılık ve çoklu iş yürütebilme yetenekleriyle programcı üzerinden büyük bir yükü kaldırmaktadır. Bu yeteneği kendi içerisinde barındırdığı ve programcıya bağımsız olarak yeniden kodlama imkânı verdiği servislerinden almaktadır.

6.1. Servis Kavramı

Microsoft Robotic Developer Studio çalışma anı Eşzamanlılık ve Koordinasyon Çalışma Anı servisi (CCR) ve Dağıtılmış Yazılım Servisi (DSS) adında robotik servisleri, uygulama kütüphanelerini ve kodları yöneten iki ana bileşen ile Ortak Dil Çalışma Platformu (CLR) ve Çalışma Anı Servisinden oluşur.

CCR, MSRS çalışma anının servisler için bir kullandığın önemli bir parçasıdır. Uygulama programlama için sınıfların kullanımı sağlar. Eşzamanlılık, koordinasyon ve hata yakalama metotlarını sunan kütüphaneleri yönetir. Bağımsız işleyebilen kod parçalarının yazılmasına imkân verir ve kod iletilerinin birbirleriyle iletişimi yönetir. İletiler alındığında sıraya koyar ve ilgili portları açar. Asenkron işlemlerin gerçekleşmesini sağlayan CCR robot simülasyonu uygulamaları için örneğin bir yandan robot kolu hareketi gerçekleşirken diğer yandan robot kolu üzerinde kullanılmış olan sensör verileri işlenmesi gibi asenkron prosesleri yürütür.
CCR tek proses parçası içinde paralel koşturulan ve ileti geçiren kod segmentlerinin çalışmasını sağlarken DSS aynı görevi karşılıklı prosesler ve hatta makineler dahilinde gerçekleştirir. Servislerin en büyük avantajlarından biri de web tabanlı hizmetler sunabilmeleridir.

Basit bir DSS servisi kod bloğu içinde ;

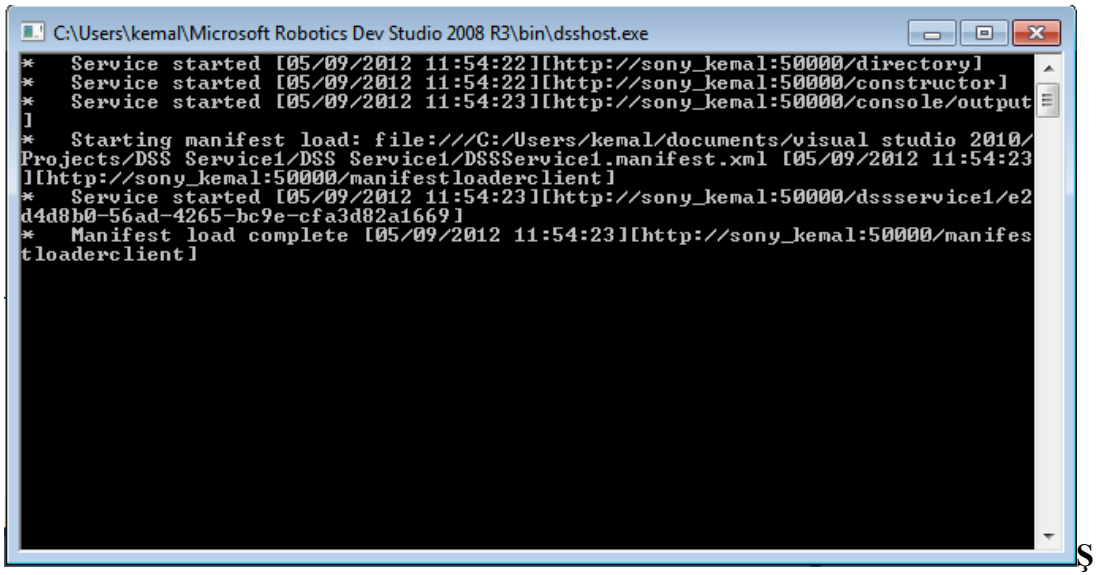
```
usingSystem;
usingSystem.Collections.Generic;
usingSystem.ComponentModel;
usingMicrosoft.Ccr.Core;
usingMicrosoft.Dss.Core.Attributes;
usingMicrosoft.Dss.ServiceModel.Dssp;
```

```
usingMicrosoft.Dss.ServiceModel.DsspServiceBase;
using W3C.Soap;
usingsubmgr = Microsoft.Dss.Services.SubscriptionManager;
usingSystem.Threading;

namespace DSSService1
{
   [Contract(Contract.Identifier)]
   [DisplayName("DSSService1")]
   [Description("DSSService1 service (nodescriptionprovided)")]
classDSSService1Service : DsspServiceBase
{
            /// Servis durumu için nesne tanımı
            [ServiceState]
      DSSService1State _state = new DSSService1State();

      /// Servis için portu tanımı
      [ServicePort("/DSSService1", AllowMultipleInstances = true)]
      DSSService1Operations _mainPort = new DSSService1Operations();

      [SubscriptionManagerPartner]
submgr.SubscriptionManagerPort _submgrPort =
newsubmgr.SubscriptionManagerPort();

  /// Servise ait yapıcı fonksiyon
public DSSService1Service(DsspServiceCreationPortcreationPort)
         : base(creationPort)
      {     }

      /// Service başlanıgıcı
protectedoverridevoid Start()
{
```

```
        //Servis için programcı tanımları kod bloğu
base.Start();


}
}
}
```

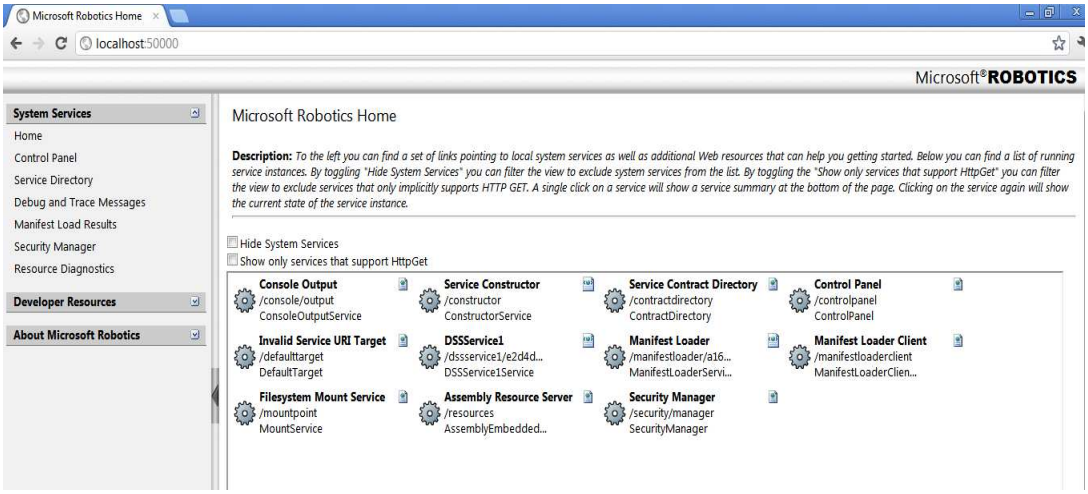
bulunur.

Servis hizmeti başlangıcı itibariyle http://localhost:50000 adresinden yayına başlar ve bu adres üzerinden servis web hizmetini verir. Web hizmeti sayesinde DSS servisini kullanan diğer servislere, servis durumlarına, güvenlik ayarlarına ve kaynak kullanımlarına erişilebilir.

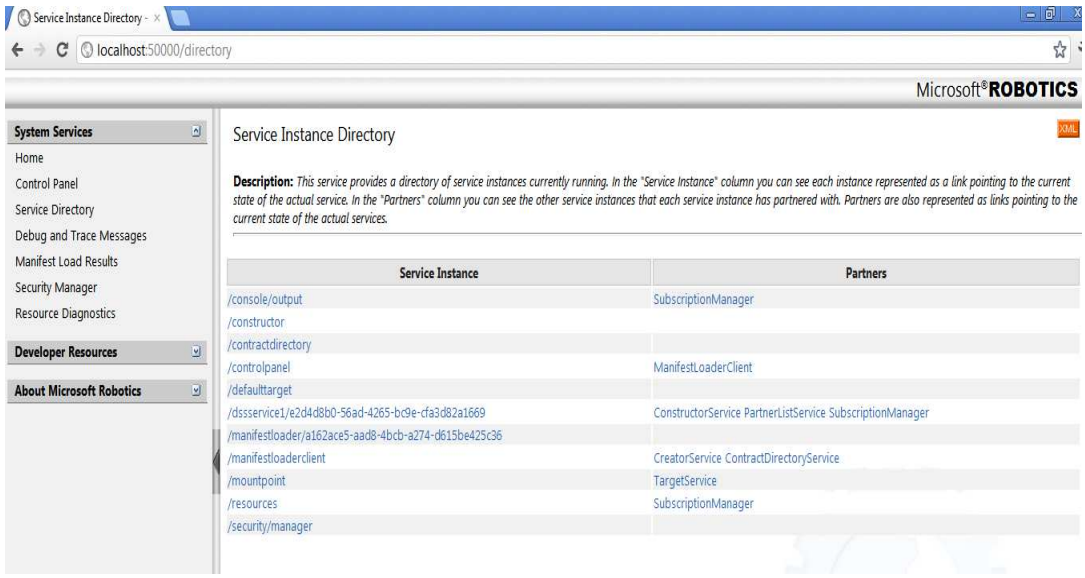


Şekil 6.1 Dssservice1 hizmetinin yürütülmesi.

Microsoft Visual Studio 2010 ortamında kodlanmış olan DSS servisimiz programlama dili platformunda yürütüldüğünde Şekil 6.1' deki gibi hizmete girer. Hizmet başlangıcı ile beraber web servisi hizmeti de başlar (Şekil 6.2).



Şekil 6.2localhost:50000



Şekil 6.3Yürütülen Servis Dizini

Web servisi hizmetiyle hizmet veren halihazırdaki servisler görülmektedir. Yürütülen Servis Dizini (Service Instance Directory) linki ile çalışan servislerden sonuç alınabilinir.

Service Instance
/console/output
/constructor
/contractdirectory
/controlpanel
/defaulttarget
/dssservice1/e2d4d8b0-56ad-4265-bc9e-cfa3d82a1669
/manifestloader/a162ace5-aad8-4bcb-a274-d615be425c36
/manifestloaderclient
/mountpoint
/resources
/security/manager

Şekil 6.4dssservice1

Şekil 4.3 deki sayfadan Şekil 4.4 deki liste göstermektedir ki dssservisimiz çalışmakta ve tıklandığında ilgili veriler XML formatında şekil 4.5 deki gibi verilmektedir.

Şekil 6.5dssservice1verileri

Robotik uygulamalar için kullanılacak olan her web tabanlı servisimiz tek bir sensörden veri alma veya aktuatörlere çıkış sinyali vermek gibi bir veya daha fazla fonksiyonu yerine getirmek için farklı kodlara ihtiyaç duyarlar. Ayrıca kodlar yardımıyla servislerin birbirleriyle iletişim kurmaları da sağlanabilir. Bir robot uygulamasında çoklu servisler robot operasyonları için eşzamanlı olarak çalışabilmelidir. Örneğin hareket halindeki bir robot için motor servisi çalışmaktayken sonar sensör servisi ve ışık sensör servisi de aynı anda veri kontrolü yapmaktadır. Bu eşzamanlılığın sağlanması için MSRS platformu kendi içindeki iki ana servisi kullanarak ve kullandırarak çözüm sağlar.

6.2 Windows Form Yapısı ve Servis İletişimi

Windows uygulamaları, Windows kontrollerinin tutulduğu pencereler olan formlardan oluşur. Programcı windows işletim sistemine ait bir uygulama yazarken bu form yapılarından faydalanır. Kullanıcı formlar üzerinden uygulamayı kullanmaktadır. MSRS platformunda servis hizmetleri ile Windows formları arasında iletişim kurabilmek windows tabanlı robotik form uygulamaları yazabilmek demektir. Bir veya birden fazla servis hizmeti ile form uygulaması arasında veri geçişleri sağlanabilir.

DSS hizmeti çalışmakta olan windows formu için bir port oluşturur. Bu port üzerinden forma ait buton tıklama, Mouse hareketi gibi olaylar mesajlar halinde DSS hizmetinde gönderilmektedir. Aşağıda genel bir form yapısı gösterilmektedir.

```
#region Genel Form Yapısı
publicclassFormEvent
{
private Form _yeniform;
public Form Form{
get {return _yeniform;}
set {_yeniform = value;}
}

publicFormEvent(Form form){
```

```
_yeniform=form;
}
}

publicclassOnLoad : FormEvent{

        publicOnLoad(Form form){

        : base(form);
}
}

publicclassOnClosed : FormEvent{
publicOnClosed(Form form){
        : base(form);
}
}
#endregion
```

FormEvent sınıfı windows form dan servise herhangibir olay bilgisinin geçmesini sağlar. Yukarıdaki örnek kod içerisinde OnLoad ve OnClosed adında iki tane windows formu oluşturulmuştur. Bunlar form açıldığında ve kapandığında tetiklenmektedir.

Form load olayı oluştuğunda servise mesaj gönderebilmek için

```
IEnumerator<ITask>OnDriveControlLoadHandler(OnLoadonLoad)
{
_driveControl = (DriveControl)onLoad.Form;
LogInfo(“Form Load olayı ile kontrol”);
```

```
yieldreturnArbiter.ExecuteToCompletion
(Environment.TaskQuene,Arbiter.FormIteratorHandler(SubscribeToGameController
));
}
```

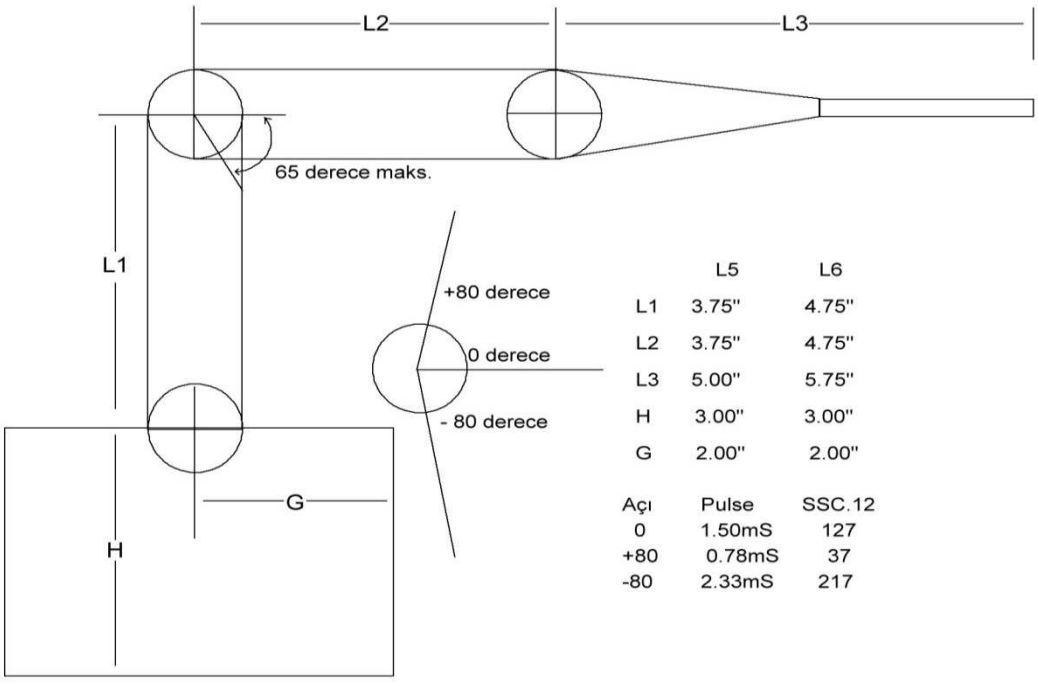
Yukarıdaki kod sayesinde form oluştuğunda servise kontrol için mesaj iletilir.
Form üzerinden klavye yardımıyla bir tuşa basıldığında her hangi bir kontrol yapabilmek için aşağıdaki kod parçasından faydalanılabilinir.

```
#region Klavye Tetikleme
privatevoidKontrol_tus_bas(objectsender, KeyEventArgs e)
{
switch((Keys)e.KeyValue)
{
caseKeys.Up:
ileri();
e.Handled=true;
break;
caseKeys.Down:
geri();
e.Handled=true;
break;
caseKeys.Left:
sag();
e.Handled=true;
break;
caseKeys.Right:
sol();
e.Handled=true;
break;
        }}
```

Yukarıdaki anlaşılır olduğu gibi içerisinde kontrol için ileri(), geri(), sag() ve sol() hareket fonksiyonlarını içermektedir. Bu fonksiyonlardan ileri() fonksiyonu aşağıdaki gibi kodlanmaktadır;

```
void ileri(){
_eventsPort.Post(newOnMove(this,(int)options.MotionSpeed,(int)options.MotionSpe
ed));
}

IEnumerator<ITask>OnMoveHandler(OnMoveonMove)
{
if(_drivePort !=null)
{
drive.SetDrivePowerRequestrequest=newdrive.SetDrivePowerRequest();
request.LeftWheelPower =
(double)onMove.Left*MOTOR_POWER_SCALE_FACTOR;
request.RightWheelPower =
(double)onMove.Right*MOTOR_POWER_SCALE_FACTOR;
drive.SetDrivePowersdp=newdrive.SetDrivePower(request);
sdp.TimeSpan = TimeSpan.FromMiliseconds(1000);
_drivePort.Post(sdp);
yieldreturnArbiter.Chooice(sdp.ResponsePort,delegate(DefaultUpdateResponseType
response)
{//güç güncellemesi}, delegate(Fault f){LogError(f);
if(f.Code.SubCode.Value == DsspFaultCodes.ResponseTimeout)
Console.WriteLine("Harekette Zaman Aşımı");
else
Console.WriteLine(f.Detail);
}
);
}}
```

6.3 Robot Kolu Modelini Kodlama



Şekil 6.6 L5 ve L6 Geometrisi

Koldaki her parçanın boyut tanımları ;

```
staticFloatInchesToMeters(floatinches){return (float)(inches * 0.0254);}
//kolun fiziksel özellikleri
staticfloat L1 = InchesToMeters(4.75f);
staticfloat L2 = InchesToMeters(4.75f);
staticfloat Grip = InchesToMeters(2.5f);
staticfloat L3 = InchesToMeters(5.75f) - Grip;
staticfloat L4 = InchesToMeters(0.03f);
staticfloat H = InchesToMeters(3f);
staticfloat G = InchesToMeters(2f);
staticfloat L1Radius = InchesToMeters(0.7f);
staticfloat L2Radius = InchesToMeters(0.7f);
staticfloat L3Radius = InchesToMeters(0.7f);
staticfloatGripRadius = InchesToMeters(0.2f);
```

H taban yüksekliği ve G çaptır. L1 üst kola karşılık giriş ve L2 alt kola karşılık giriş kısmıdır. L3 bilek ve tutucuyu içerir. L4 girişi bilek dönüşünü gerçekleştirmek için kullanılır.

Her parçaya yeni bir segment eklemek için eklenecek parçayı tanımlamak gerekir. Bunun için aşağıdaki gibi bir sınıf tanımlanabilir;

```
classJointDesc
{
publicstring Name;
publicfloatMin;
publicfloatMax;
publicPhysicsJointJoint;
publicPhysicsJoint Joint2;
publicfloatTarget;
publicfloatCurrent;
publicfloatSpeed;
publicJointDesc(string name, floatmin, floatmax)
{
Name=name;Min=min;Max=max;
Joint=null;Joint2=null;
Current=Target=0;
Speed=30;
}
publicboolValidTarget(floattarget)
{
return ((target>=Min) && (target<=Max));
}
publicboolNeedToMove(float epsilon)
{
if(Joint==null){returnfalse;}
return ( Math.Abs(Target – Current)  > epsilon );
}
```

```
publicvoidUpdateCurrent(double time)
{
float delta=(float)(time*Speed);
if(Target>Current){Current=Math.Min(Current + delta, Target);}
else{Current=Math.Max(Current – delta, Target);}
}
}
```

kol eklemleri için tanımlanan dizi ;

```
JointDesc[] _joints = newJointDesc()
{
newJointDesc(“Base”, -180 , 180 ),
newJointDesc(“Shoulder”, -90 , 90 ),
newJointDesc(“Elbow”, -65  , 115 ),
newJointDesc(“Wrist”, -90  , 90  ),
newJointDesc(“WristRotate”, -90  , 90 ),
newJointDesc(“Gripper”, 0 , InchesToMeter(2) )
};
```

taban yerleşkesi ve şeklinin oluşturulması için gereken kodlar ;

```
floatbaseHeight = H – L1Radius – 0.001f;
State.Name= name;
State.Pose.Position = position;
State.Pose.Orientation = newQuaternion( 0, 0, 0, 1);
State.Assets.Mesh=”L6_Base.obj”;
MeshTranslation = new Vector3(0, 0.026f, 0);
BoxShape = newBoxShape(newBoxShapeProperties( “Base”, 150,
newPose(new Vector3(0, baseHeight / 2, 0), newQuaternion(0, 0, 0, 1)),
new Vector3(G * 2, baseHeight, G * 2) ) );
```

Eğer oluşturulan simülasyon ortamında bu kolu başka bir giriş noktasına eklemek istersek, her bir kol girişi için koldaki ağırlıkları belirlemek isteyeceğiz. Kol tabanından hesaplamalara başlamak için kinematik kullanımı kodu;

```
State.Flags | = EntitySimulationModifiers.Kinematic;
SphereShape
L0Sphere=newSphereShape(newSphereShapeProperties(“L0Sphere”,50,new
Pose(new Vector3(0,0,0), newQuaternion(0,0,0,1)),L1Radius));
SingleShapeSegmentEntity L0Entity =
newSingleShapeSegmentEntity(L0Sphere,position + new Vector3(0,H,0));
L0Entity.State.Pose.Orientation = newQuaternion(0,0,0,1);
L0Entity.State.Name=name +”_L0”;
L0Entity.State.Assets.Mesh=”L6_L0.obj”;
L0Entity.MeshTranslation=newJointAngularProperties();
L0Angular.Swing1Mode=JointDOFMode.Free;
L0Angular.SwingDrive=newJointDriveProperties(JointDriveMode.Position,newSpri
ngProperties(50000000,1000,0),100000000);
EntityJointConnector[] L0Connectors = newEntityJointConnector[2]{
newEntityJointConnector(L0Entity,new Vector3(0,1,0) ,new Vector3(1,0,0),new
Vector3(0,0,0)),
newEntityJointConnector(L0Entity,new Vector3(0,1,0) ,new Vector3(1,0,0),new
Vector3(0,H,0))
};
L0Entity.CustomJoint = newJoint();
L0Entity.CustomJoint.State= newJointProperties(L0Angular, L0Connectors);
L0Entity.CustomJoint.State.Name=”BaseJoint”;
this.InsertEntityGlobal(L0Entity);
```

L1 kol uzantısı oluşturmak ve posizyonunu vermek için ;

```
CapsuleShape L1Capsule=newCapsuleShape(newCapsuleShapeProperties(
“L1Capsule”,2, newPase (new vector3(0,0,0), newQuaternion (0,0,0,1)),L1Radius,
L1));
SingleShapeSegmentEntity(L1Capsule,position+new vector3(0,H,0));
L1Entity.State.pose.Orientation= newQuaternion(0,0,0,1);
L1Entity.State.name=name+”_L1”;
L1Entity.State.Assets.Mesh=”L6_L1.obj”;
JointAngularProperties L1Angular=newJointAngularProperties();
L1Angular.TwistMode=JointDOFMode.Free;
L1Angular.TwistMode= newJointDriveProperties(JointDriveMode.position,
newSpringProperties(50000000,1000,0),100000000);
EntityJointConnector[] L1Connectors=newEntityJointConnector[2]
{
newEntityJointConnector(L1Entity,
new vector3(0,1,0), new vector3(0,0,1), new vector3(0, -L1/2,0)),
newEntityJointConnector(L0Entity,
new vector3(0,1,0), new vector3(0,0,1), new vector3(0,0,0))
};
L1Entity.CustomJoint=newJoint();
L1Entity.CustomJoint.state= newJointPoperties(L1Angular,L1Connectors);
L1Entity.CustomJoint.state.name=”Shoulder|-80|80|”;
L0Entity.InsertEntityGlobal(L1Entity);
L2 kol uzantısı oluşturmak ve posizyonunu vermek için ;
EntityJointConnector[] L2Connectors =newEntityJonintConnector[2]
{
newEntityJonintConnector(L2Entity,
new vector3(1,0,0), new vector3(0,0,1), new vector3(0,-L2/2,0)),
newEntityJonintConnector(L1Entity,
new vector3(0,1,0), new vector3(0,0,1), new vector3(0,L2/2,0))
};
CapsuleShapeLeftGripCapsule=newCapsuleShape( newCapsuleShapeProperties(
```

```
“LeftGripCapsule”, 1f, newpase (new vector3(0,0,0), newQuaternion(0,0,0,1)),
GripRadius, Grip));
LeftGridCapsule.State.DiffuseColor = new Vector4(0,0,0,1);
LeftGripEntity=newSingleShapeSegmentEntity(LeftGripCapsule, position + new
Vector3(0,H,0));
LeftGripEntity.Position = new xna.Vector3(-0.24f, 0.19f, 0.01f);
LeftGripEntity.Rotation = new xna.Vector3(179.94f, -176.91f, 89.67f);
LeftGripEntity.State.Name=name+”_LeftGrip”;
JointLinearPropertiesLeftGripLinear = newJointLinearProperties();
LeftGripLinear.XMotionMode=JointDOFMode.Free;
LeftGripLinear.XDrive =
newJointDriveProperties(JointDriveMode.Position,newSpringProperties(500000000
0,1000,0),100000000);
EntityJointConnector[] LeftGripConnectors = newEntityJointConnector[2]
{
newEntityJointConnector(LeftGripEntity, new Vector3(1,0,0), new Vector3 (0,0,1),
new Vector3(0, -Grip/2,0)),newEntityJointConnector(L4Entity,new Vector3(1,0,0),
new Vector3(0,0,1), new Vector3(0, L4/2,GripRadius))
};
LeftGripEntity.CustomJoint = newJoint();
LeftGripEntity.CustomJoint.State= newJointProperties(LeftGripLinear,
LeftGripConnectors);
LeftGripEntity.CustomJoint.State.Name= “LeftGripJoint|-0.0254|0|”;
L4Entity.InsertEntityGlobal(LeftGripEntity);
}
```

6.4 Robot Kolu Hareketi

Modellenen robot kolu modelinin hareketi için belli parametreleri "MoveTo" yöntemi kullanılır. Bu parametreler ve kazandırdıkları yetenekler aşağıdaki gibidir;

Tablo 6.1 MoveTo parametreleri

Parametre	Değer Türü	Tanımı
baseVal	Derece	taban ekleminin dönüş açısı
shoulderVal	Derece	omuz ekleminin pivot açısı
elbowVal	Derece	dirsek ekleminin pivot açısı
wristVal	Derece	bilek ekleminin pivot açısı
rotateVal	Derece	bilek ekleminin dönüş açısı
gripperVal	Metre	kıskaçları arası mesafe
Time	Saniye	hareketi tamamlama süresi

Yukarıdaki parametreler ile çağrılan yöntem her çağrıldığından sonra geriye SuccessFailurePort değerini döndermektedir. Yöntem genel olarak hata yakalama blokları içerisinde çağrılmaktadır.

```
SuccessFailurePortresponsePort = newSuccessFailurePort();
if(_moveActive)
{
responsePort.Post(newException("Önceki hareket sonlanmadı"));
returnresponsePort;
}

if(!_joints[0].ValidTarget(baseVal))
{
responsePort.Post(newException()_joints[0].Name+"Eklem yanlış değerde hareket
ettirilmeye çalışılıyor "+ baseVal.ToString()));
returnresponsePort;
}
```

Bütün parametrelerin doğrulaması gerçekleştikten sonra Target yani hedef değerler için değer atamaları yapılır.

```
_joints[0].Target=baseVal;
_joints[1].Target=shoulderVal;
_joints[2].Target=elbowVal;
_joints[3].Target=wristVal;
_joints[4].Target=rotateVal;
_joints[5].Target=gripperVal;

for(int i=0;i<6;i++)
{_joints[i].Speed = Math.Abs( _joints[i].Target - joints[i].Current )}
```

MoveTo yöntemi _moveToActive bayrağını kullanarak cevap olarak true veya false değerleri hareketin doğrulunu ifade edebilmek için kullanılmaktadır. Cevap alınması hareketi gerçekleştirmez ve hareketin gerçekleştrilmesi için update yöntemi kullanılmıştır. Update yöntemi seniyede 60 kere çağrılabilinir.

```
if(_moveToActive)
{
booldone=true;
if(_joints[0].NeedToMove(_epsilon))
{
done=false;
Vector3 normal=_joints[0].Joint.State.Connectors[0].JointNormal;
_joints[0].UpdateCurrent(_prevTime);
_joints[0].Joint.SetAngularDriveOrientation(Quaternion.FromAxisAngle(normal.X,
normal.Y,normal.Z,DegreesToRadians(_joints[0].Current)));
}
}
```

Yukarıda kod ile eklem yeni hedef noktaya hareket ettirilmiştir. Eğer hareket dizisi aktif ise bir sonraki hareket gereklilik şartı sağlandığında ki bu NeedToMove ile alınmaktadır UpdateCurrent yöntemi ile hareket gerçekleştirilmektedir.

Ters Kinematik yöntemi kullanılarak eklemlere hareket yeteneği verilmek istendiğinde yukarıdaki kod bloklarında değişiklik yapmak gerekmiştir. Çünkü Ters Kinematik mantığındaki farklılık dolayısıyla algoritma ve kullanılan yöntemleri de farklı kılmaktadır. Ters kinematik mantığı ile hareket kabiliyeti için MoveToPosition yönetimi kullanılır. Bu fonksiyonun aldığı parametreler ;

Tablo 6.2 MoveToPosition parametreleri

Parametre	**Değer Türü**	**Tanımı**
X	Metre	kıskaç ucu merkezinin X pozisyonu
Y	Metre	kıskaç ucu merkezinin Y pozisyonu
Z	Metre	kıskaç ucu merkezinin Z pozisyonu
P	Açısal	-90 kıskancın düşüş pozisyonu için, kıskacın açısal yaklaşımı
W	Açısal	bilek ekleminin açısal dönüşü
Grip	Meter	kıskaç aralığı mesafesi
Time	Saniye	hareketi tamamlama süresi

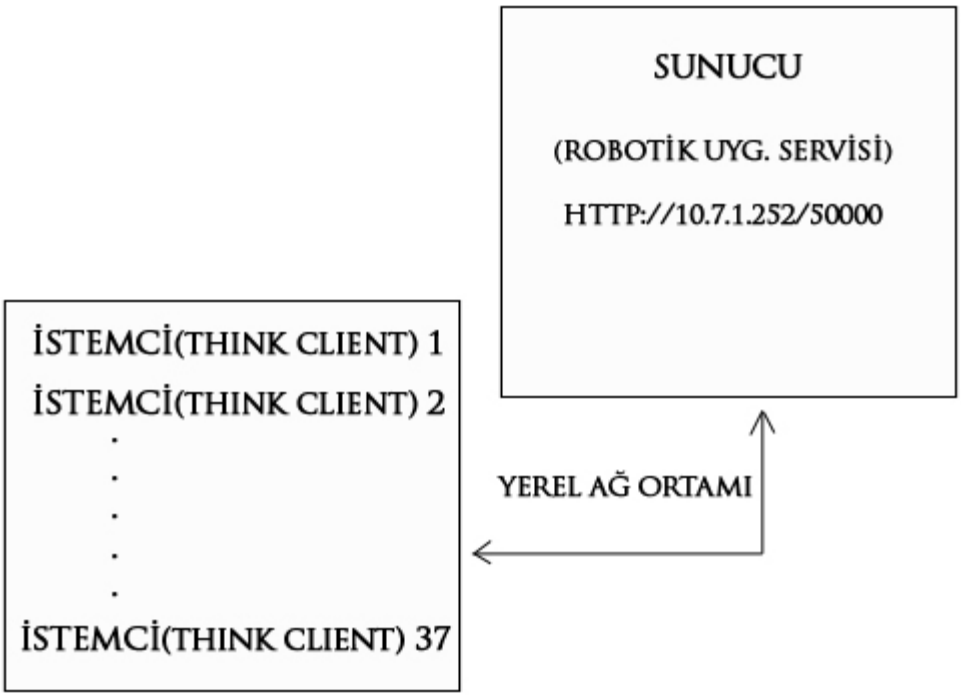
MoveToPosition yöntemi için doğrulama bloğu aşağıdaki şekilde kodlanmıştır;

```
publicSuccessFailureMoveToPosition(
float x,
float y,
float z,
float p,
float w,
float grip,
float time)
{
float L1 = InchesToMeters(4.75f);
float L2 = InchesToMeters(4.75f);
float Grip = InchesToMeters(2.5f);
float L3 = InchesToMeters(5.75f);
float H = InchesToMeters(3f);
float G = InchesToMeters(2f);
```

```
float r = (float)Math.Sqrt( x*x + z*z);
floatbaseAngle = (float)Math.Atan2(-z, -x);
floatpRad = DegreesToRadians(p);
floatrb= (float)(( r –  L3 * Math.Cos(pRad)) / (2 * L1));
floatyb= (float)(( y – H – L3 * Math.Cos(pRad)) / (2 * L1));
float q = (float)(Math.Sqrt(1/(rb*rb+yb*yb) – 1));
float p1=(float)(Math.Atan2(yb+q*rb , rb – q*yb));
float p2=(float)(Math.Atan2(yb+q*rb , rb + q*yb));
floatshoulder = p1 – DegreesToRadians(90);
floatelbow = p2 – shoulder;
floatwrist = pRad – p2;
return _16Arm.MoveTo(
RadiansToDegrees(baseAngle),
RadiansToDegrees(shoulder),
RadiansToDegrees(elbow),
RadiansToDegrees(wrist),
w, grip, time);}
```

7. LABORATUAR UYGULAMASI

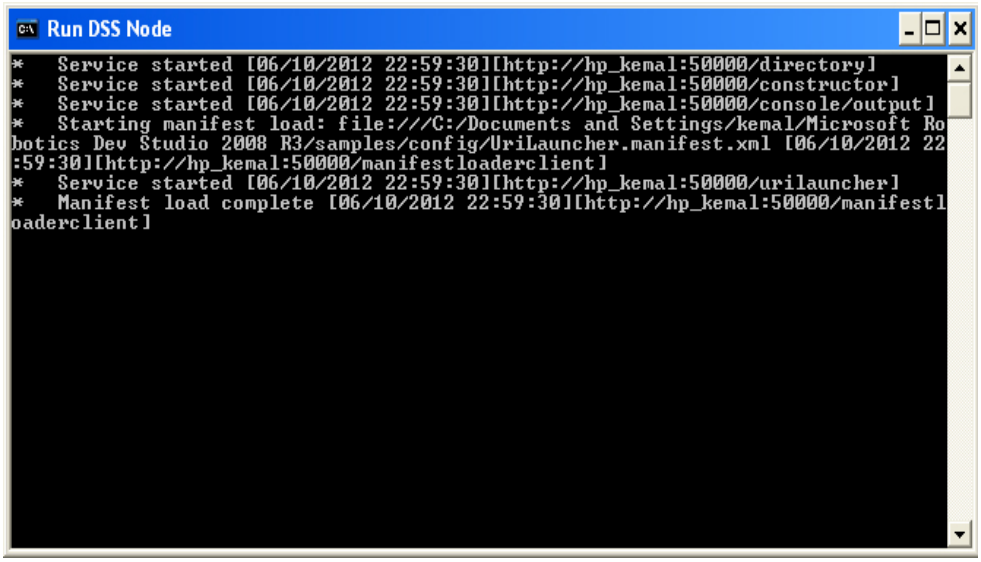
Çalışmada 37 thinkclient tipi bilgisayarın bağlı olduğu bir sunucu sistemi ve ağ ortamı kullanılmaktadır.



Şekil 7.1 Sistemin Çalışma Ortamı

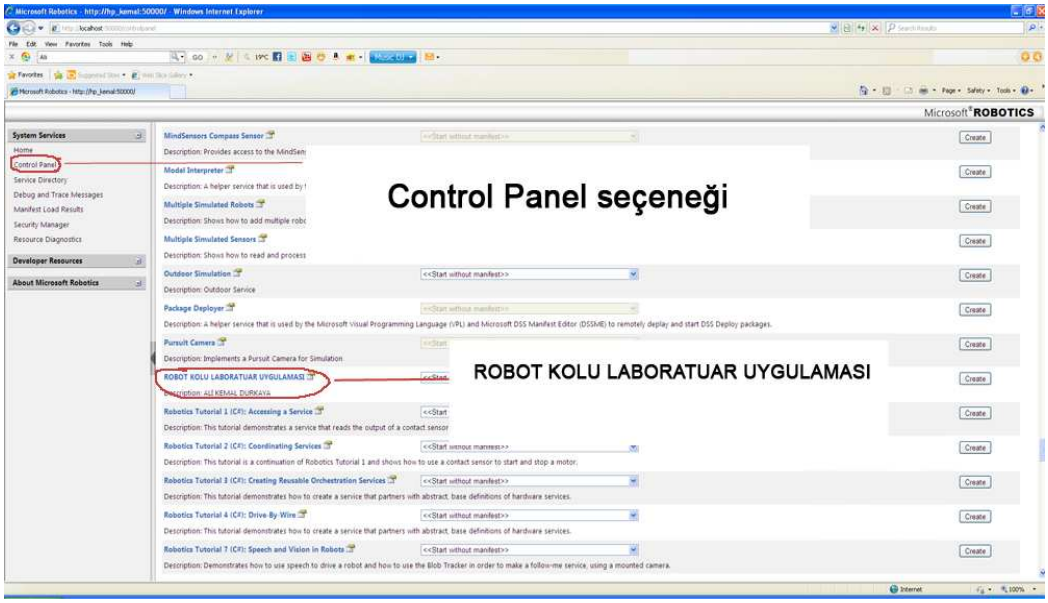
Sunucu üzerinde hizmet veren servislerimiz yerel ağ içerisinde 10.7.1.252 ıp adresi üzerinden çalışmaktadır. Bu ıp adresi üzerinden çalışmakta olan sunucumuz içerisinde 50000 numaralı port kullanılarak robotik uygulama servislerimiz hizmet verir. Yukarıdaki sistemde yer alan istemcilere ait herhangi bir internet browser adres cubuğunahttp://10.7.1.252/50000 adresi yazıldığında hizmet vermekte olan servislere erişim sağlanmış olur.

Servisin başlatılması için “Run DSS Node” exe dosyası yürütülmelidir. Bu komut yürütüldüğünde şekil 7.2 ‘de gösterildiği gibi bir komut penceresi çalışmaya başlayacaktır.



Şekil 7.2 Run DSS Node komut pençeresi

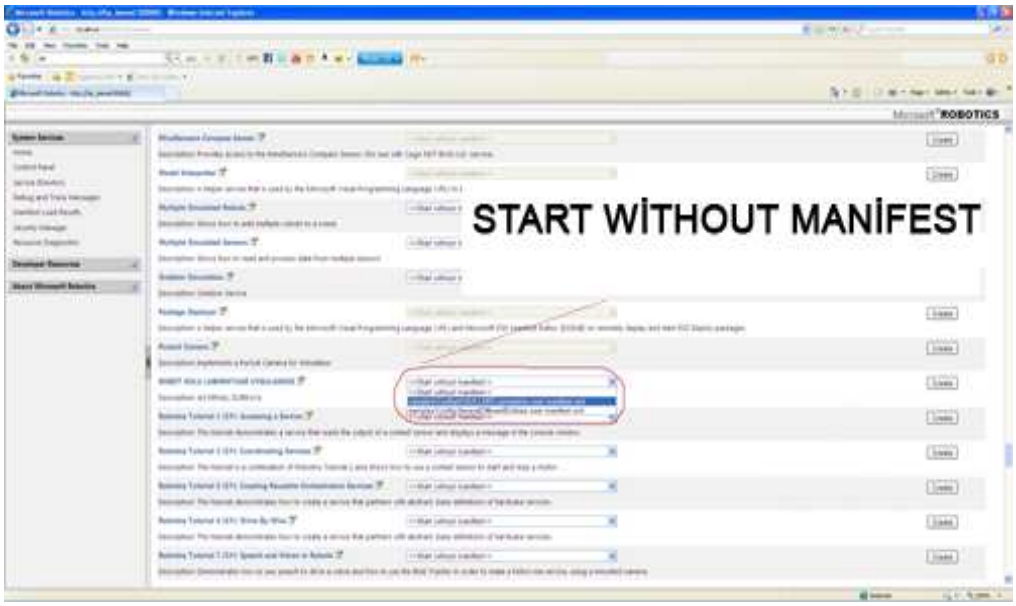
Komut çalıştırıldığında sunucu Şekil 7.3 ‘deki gibi 50000. port üzerinden servis hizmetini vermeye başlar.



Şekil 7.3 Servis Web Hizmeti

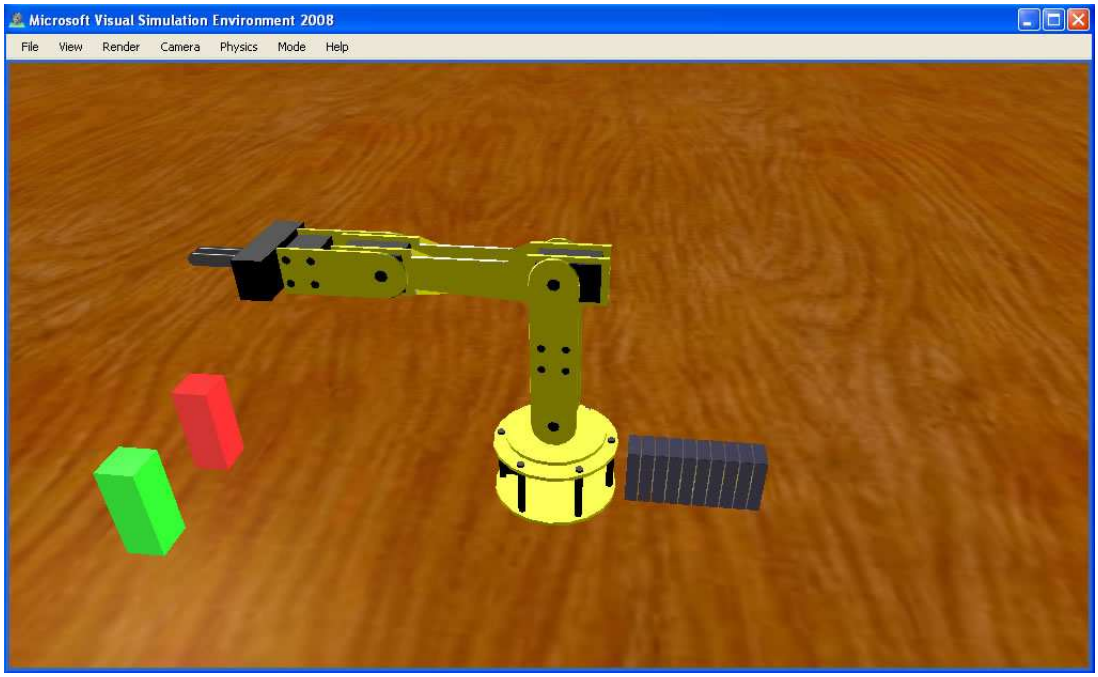
Browser ile arayüzüne erişilen servis hizmeti için Microsoft İnternet Explorer 7.0 ve üzeri sürümlerinden faydalanmak gerekmektedir. Sayfada sol tarafra yer alan “System Service” sekmesi altındaki” “Control Panel” seçeneği ile kullanılabilecek olan servisler görülmektedir. SimulatedLynxL6Arm,SimulatedWebcam,

Simulation Engine Servisleri sırasıyla robot kolu bilgilerini, robot kolu simülasyon ortamı kamera bakışını ve fiziksel ortamdaki bütün büyükleri vermektedir. Servisi başlatmak için Şekil 7.4 de gösterildiği gibi “startwithoutmanifest” seçeneği seçilerek servislerden istenileni tıklanır.



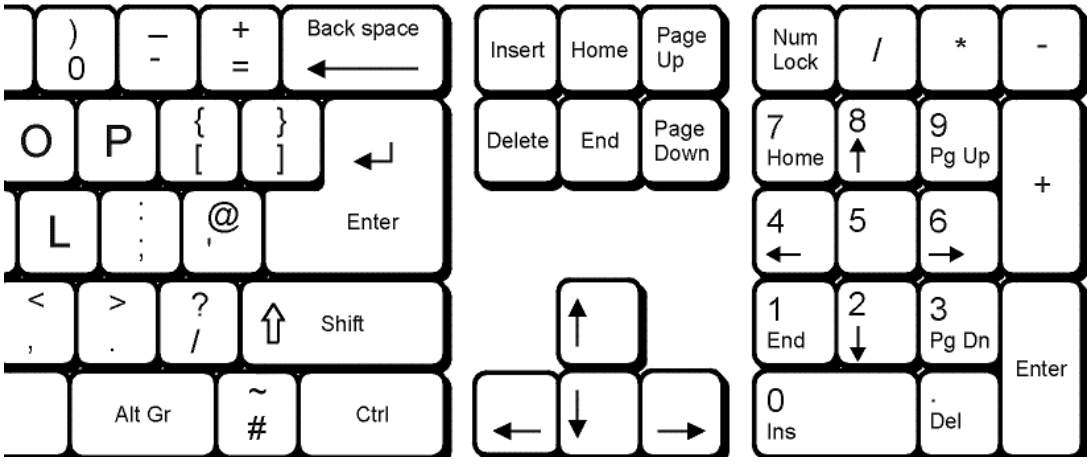
Şekil 7.4 Servisi Başlatma

Uygulama servisini başlattığımızda Şekil 7.5’ deki gibi robot kolu simülasyonu ve Şekil 7.7’deki gibi kontrol arayüzü açılacaktır. Kontrol arayüzü kullanılarak simülasyon içerisindeki kol kontrolü sağlanmaktadır.

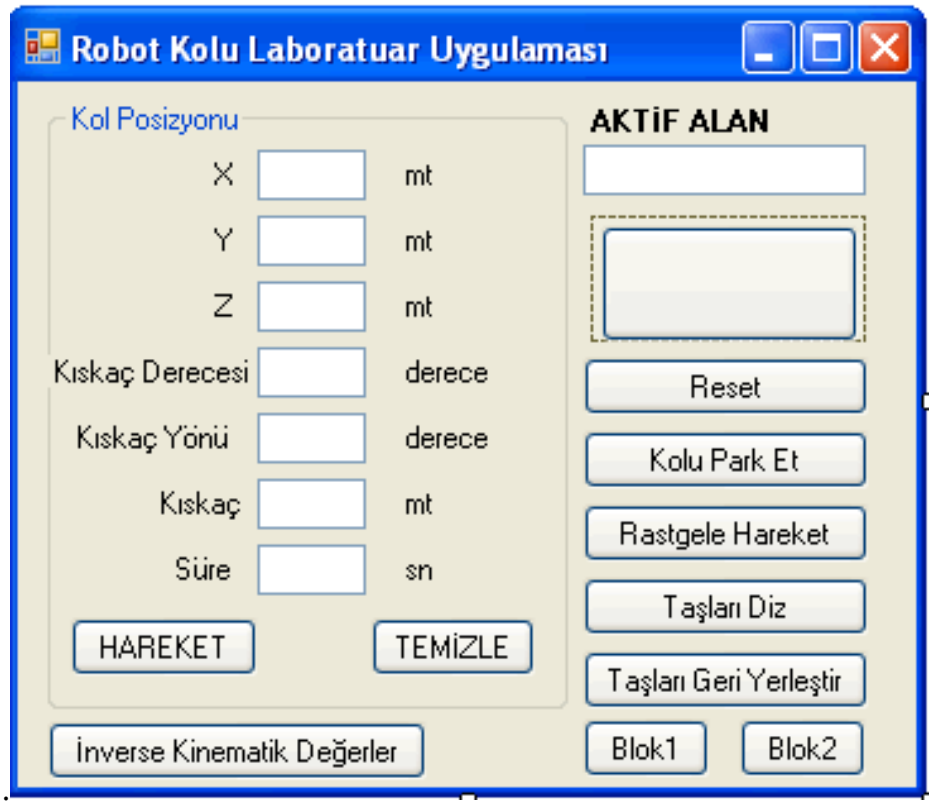


Şekil 7.5 Simülasyon ortamı

Arayüz üzerinden simülasyonda yer alan robot koluna ait kol posizyon bilgileri anlık olarak görülebilmektedir. Hareket hem sistem yerleştirilmiş algoritmalar üzerinden hem de Aktif Alan seçilerek klavyede yer alan tuş takımı üzerinden yapılabilir.



Şekil. 7.6 Tuş kontrol



Şekil 7.7 Kontrol Arayüzü

Tuş kontrol üzerinde;

7/1 tuşları z ekseninde, 6/4 x ekseninde, 8/2 y ekseninde hareketi, +/- kıskaçlar arası uzaklığı, *// kıskaçların dönme hareketini, 9/3 kıskaç açısını kontolü sağlamaktadır.

Kontrol Arayüzüne ait kodlar ;

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
//using System.Data;
//using System.Drawing;
using System.Text;
using System.Windows.Forms;

namespace ProMRDS.Simulation.LynxL6Arm
{
publicpartialclassSimulatedLynxL6ArmUI : Form
    {
FromWinformEvents _fromWinformPort;

// Short term memory for Save/Restore
Single _x;
Single _y;
Single _z;
```

```
Single _gripAngle;
Single _gripRotation;
Single _grip;
Single _time;

public SimulatedLynxL6ArmUI(FromWinformEvents EventsPort)
        {
            _fromWinformPort = EventsPort;

            InitializeComponent();


_fromWinformPort.Post(newFromWinformMsg(FromWinformMsg.MsgEnum.Loade
d, null, this));
        }

publicvoid SetErrorText(string error)
        {
            _errorLabel.Text = error;
        }

publicvoid SetPositionText(float x, float y, float z, float p, float
w, float grip, float time)
        {
            _xText.Text = x.ToString();
            _yText.Text = y.ToString();
            _zText.Text = z.ToString();
            _gripAngleText.Text = p.ToString();
            _gripRotationText.Text = w.ToString();
            _gripText.Text = grip.ToString();
            _timeText.Text = time.ToString();
        }

privatevoid _startButton_Click(object sender, EventArgs e)
        {

_fromWinformPort.Post(newFromWinformMsg(FromWinformMsg.MsgEnum.Start
, null));
        }

privatevoid _resetButton_Click(object sender, EventArgs e)
        {

_fromWinformPort.Post(newFromWinformMsg(FromWinformMsg.MsgEnum.Reset
, null));
        }

privatevoid _submitButton_Click(object sender, EventArgs e)
        {
try
            {
MoveToPositionParameters moveParams = newMoveToPositionParameters();

                _errorLabel.Text = string.Empty;

                moveParams.X = Single.Parse(_xText.Text);
                moveParams.Y = Single.Parse(_yText.Text);
                moveParams.Z = Single.Parse(_zText.Text);
```

```
                moveParams.GripAngle =
Single.Parse(_gripAngleText.Text);
                moveParams.GripRotation =
Single.Parse(_gripRotationText.Text);
                moveParams.Grip = Single.Parse(_gripText.Text);
                moveParams.Time = Single.Parse(_timeText.Text);


_fromWinformPort.Post(newFromWinformMsg(FromWinformMsg.MsgEnum.MoveT
oPosition, null, moveParams));
            }
catch
            {
                _errorLabel.Text = "Invalid Value";
            }
        }

privatevoid _reverseButton_Click(object sender, EventArgs e)
        {

_fromWinformPort.Post(newFromWinformMsg(FromWinformMsg.MsgEnum.Rever
seDominos, null, null));
        }

privatevoid _parkButton_Click(object sender, EventArgs e)
        {

_fromWinformPort.Post(newFromWinformMsg(FromWinformMsg.MsgEnum.Park,
null, null));
        }

privatevoid _toppleButton_Click(object sender, EventArgs e)
        {

_fromWinformPort.Post(newFromWinformMsg(FromWinformMsg.MsgEnum.Toppl
eDominos, null, null));
        }

privatevoid _randomButton_Click(object sender, EventArgs e)
        {

_fromWinformPort.Post(newFromWinformMsg(FromWinformMsg.MsgEnum.Rando
mMove, null, null));
        }


// Save and Restore functions for pose
privatevoid _saveButton_Click(object sender, EventArgs e)
        {
Single x, y, z, gripAngle, gripRotation, grip, time;
try
            {
                _errorLabel.Text = string.Empty;

                x = Single.Parse(_xText.Text);
                y = Single.Parse(_yText.Text);
                z = Single.Parse(_zText.Text);
                gripAngle = Single.Parse(_gripAngleText.Text);
                gripRotation = Single.Parse(_gripRotationText.Text);
```

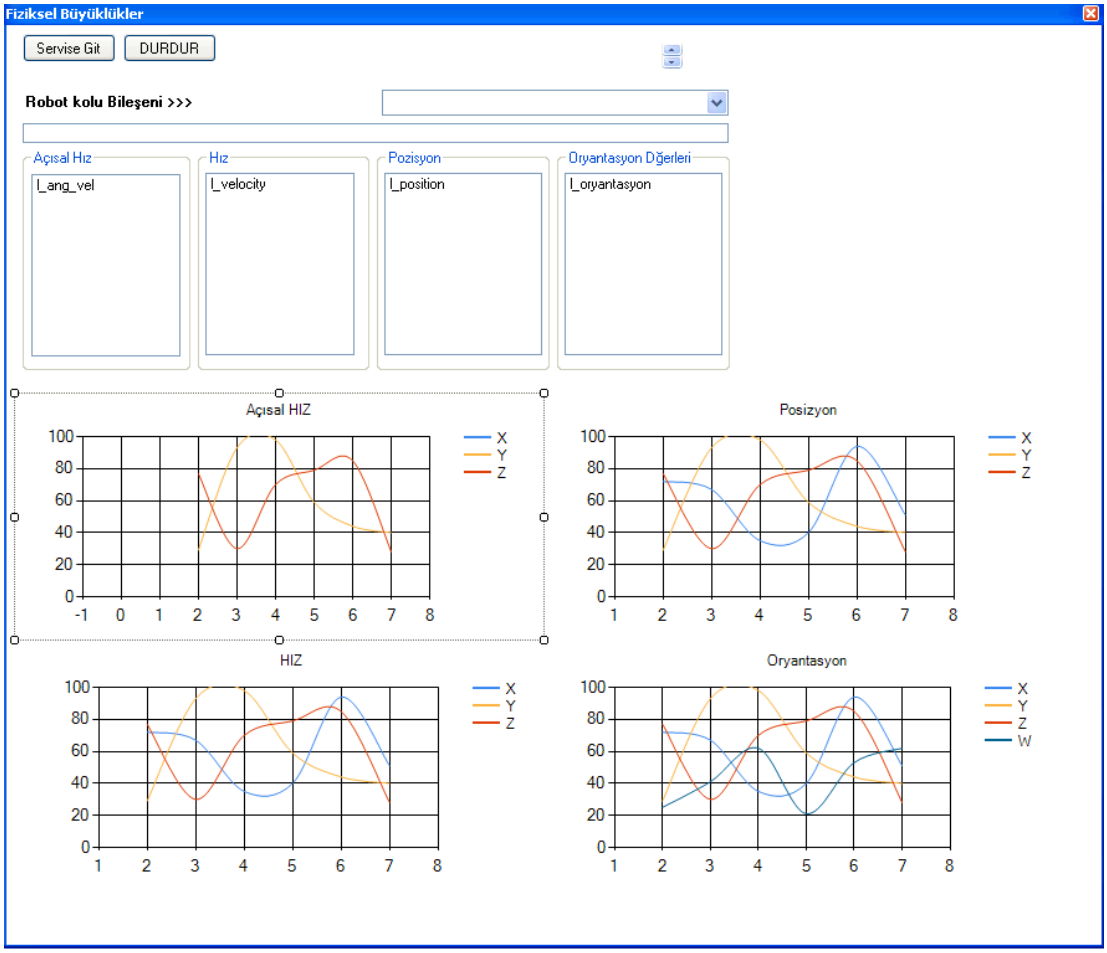
```
                grip = Single.Parse(_gripText.Text);
                time = Single.Parse(_timeText.Text);
// Succeeded in parsing value so set them now
                _x = x;
                _y = y;
                _z = z;
                _gripAngle = gripAngle;
                _gripRotation = gripRotation;
                _grip = grip;
                _time = time;
            }
catch
            {
                _errorLabel.Text = "Invalid Value";
            }

        }

privatevoid _restoreButton_Click(object sender, EventArgs e)
        {
            SetPositionText(_x, _y, _z, _gripAngle, _gripRotation,
_grip, _time);
        }



privatevoid button1_KeyPress(object sender, KeyPressEventArgs e)
        {


if (_xText.Text.Length>0)
            {
if(e.KeyChar=='6'){
                move_arm((Single.Parse(_xText.Text)+0.05f) + "",

Single.Parse(_yText.Text) + "", Single.Parse(_zText.Text) + "",
Single.Parse(_gripAngleText.Text)+"",
Single.Parse(_gripRotationText.Text)+"",
Single.Parse(_gripText.Text)+"",Single.Parse(_timeText.Text)+"");
                _xText.Text = "" + (Single.Parse(_xText.Text) +
0.05f);
                textBox1.Text = "eksen X pozitif";
            }
if (e.KeyChar == '4')
            {
                move_arm((Single.Parse(_xText.Text) - 0.05f) + "",
Single.Parse(_yText.Text) + "", Single.Parse(_zText.Text) + "",
Single.Parse(_gripAngleText.Text) + "",
Single.Parse(_gripRotationText.Text) + "",
Single.Parse(_gripText.Text) + "", Single.Parse(_timeText.Text) +
"");
                _xText.Text = "" + (Single.Parse(_xText.Text) -
0.05f);
                textBox1.Text = "eksen X negatif";
            }
            }
if(_yText.Text.Length>0)
            {
```

```
if(e.KeyChar=='8'){



move_arm(Single.Parse(_xText.Text) +
"",(Single.Parse(_yText.Text)+0.05f) + "", Single.Parse(_zText.Text)
+ "", Single.Parse(_gripAngleText.Text)+"",
Single.Parse(_gripRotationText.Text)+"",
Single.Parse(_gripText.Text)+"",Single.Parse(_timeText.Text)+"");
                _yText.Text = "" + (Single.Parse(_xText.Text) +
0.05f);
                textBox1.Text = "eksen y pozitif";
            }
if (e.KeyChar == '2')
            {
                move_arm(Single.Parse(_xText.Text) +
"",(Single.Parse(_yText.Text)-0.05f) + "", Single.Parse(_zText.Text)
+ "", Single.Parse(_gripAngleText.Text) + "",
Single.Parse(_gripRotationText.Text) + "",
Single.Parse(_gripText.Text) + "", Single.Parse(_timeText.Text) +
"");
                _yText.Text = "" + (Single.Parse(_yText.Text) -
0.05f);
                textBox1.Text = "eksen y negatif";
            }
            }

if (_zText.Text.Length > 0)
            {
if (e.KeyChar == '1')
                {

move_arm(Single.Parse(_xText.Text) + "", Single.Parse(_yText.Text) +
"",(Single.Parse(_zText.Text) + 0.05f) + "",
Single.Parse(_gripAngleText.Text) + "",
Single.Parse(_gripRotationText.Text) + "",
Single.Parse(_gripText.Text) + "", Single.Parse(_timeText.Text) +
"");
                    _zText.Text = "" + (Single.Parse(_zText.Text) +
0.05f);
                    textBox1.Text = "eksen z pozitif";
                }
if (e.KeyChar == '7')
                {
                    move_arm(Single.Parse(_xText.Text) + "",
Single.Parse(_yText.Text) + "", (Single.Parse(_zText.Text) - 0.05f)
+ "", Single.Parse(_gripAngleText.Text) + "",
Single.Parse(_gripRotationText.Text) + "",
Single.Parse(_gripText.Text) + "", Single.Parse(_timeText.Text) +
"");
                    _zText.Text = "" + (Single.Parse(_zText.Text) -
0.05f);
                    textBox1.Text = "eksen z negatif";
                }
            }

if (_gripText.Text.Length > 0)
            {
```

```
if (e.KeyChar == '+')
                {

 move_arm(Single.Parse(_xText.Text) + "", Single.Parse(_yText.Text)
+ "", Single.Parse(_zText.Text) + "",
Single.Parse(_gripAngleText.Text) + "",
Single.Parse(_gripRotationText.Text) + "",
(Single.Parse(_gripText.Text) + 0.001f) + "",
Single.Parse(_timeText.Text) + "");
                    _gripText.Text = "" +
(Single.Parse(_gripText.Text) + 0.001f);
                    textBox1.Text = "Kıskaç Aç";
                }
if (e.KeyChar == '-')
                {
                    move_arm(Single.Parse(_xText.Text) + "",
Single.Parse(_yText.Text) + "", Single.Parse(_zText.Text) + "",
Single.Parse(_gripAngleText.Text) + "",
Single.Parse(_gripRotationText.Text) + "",
(Single.Parse(_gripText.Text) - 0.001f) + "",
Single.Parse(_timeText.Text) + "");
                    _gripText.Text = "" +
(Single.Parse(_gripText.Text) - 0.001f);
                    textBox1.Text = "Kıskaç Kapa";
                }
            }

if (_gripRotationText.Text.Length > 0)
            {
if (e.KeyChar == '*')
                {
                    move_arm(Single.Parse(_xText.Text) + "",
Single.Parse(_yText.Text) + "", Single.Parse(_zText.Text) + "",
Single.Parse(_gripAngleText.Text) + "",
(Single.Parse(_gripRotationText.Text)+10) + "",
Single.Parse(_gripText.Text) + "", Single.Parse(_timeText.Text) +
"");
                    _gripRotationText.Text = "" +
(Single.Parse(_gripRotationText.Text) + 10);
                    textBox1.Text = "Kıskaç Dönder +";
                }
if (e.KeyChar == '/')
                {
                    move_arm(Single.Parse(_xText.Text) + "",
Single.Parse(_yText.Text) + "", Single.Parse(_zText.Text) + "",
Single.Parse(_gripAngleText.Text) + "",
(Single.Parse(_gripRotationText.Text)-10) + "",
Single.Parse(_gripText.Text)  + "", Single.Parse(_timeText.Text) +
"");
                    _gripText.Text = "" +
(Single.Parse(_gripRotationText.Text) - 10);
                    textBox1.Text = "Kıskaç Dönder -";
                }
            }

if (_gripRotationText.Text.Length > 0)
            {
if (e.KeyChar == '9')
                {
```

```
                    move_arm(Single.Parse(_xText.Text) + "",
Single.Parse(_yText.Text) + "", Single.Parse(_zText.Text) + "",
(Single.Parse(_gripAngleText.Text)+10) + "",
Single.Parse(_gripRotationText.Text)+ "",
Single.Parse(_gripText.Text) + "", Single.Parse(_timeText.Text) +
"");
                    _gripAngleText.Text = "" +
(Single.Parse(_gripAngleText.Text) + 10);
                    textBox1.Text = "Kıskaç Açı +";
                }
if (e.KeyChar == '3')
                {
                    move_arm(Single.Parse(_xText.Text) + "",
Single.Parse(_yText.Text) + "", Single.Parse(_zText.Text) + "",
(Single.Parse(_gripAngleText.Text)-10) + "",
Single.Parse(_gripRotationText.Text) + "",
Single.Parse(_gripText.Text) + "", Single.Parse(_timeText.Text) +
"");
                    _gripAngleText.Text = "" +
(Single.Parse(_gripAngleText.Text) - 10);
                    textBox1.Text = "Kıskaç Açı -";
                }
            }
        }

privatevoid move_arm(string x,string y,string z,string
kskc_degree,string kskc_rotation,string kskc_mt,string time_to_move)
{

try
            {
MoveToPositionParameters moveParams = newMoveToPositionParameters();

                _errorLabel.Text = string.Empty;

                moveParams.X = Single.Parse(x);
                moveParams.Y = Single.Parse(y);
                moveParams.Z = Single.Parse(z);
                moveParams.GripAngle = Single.Parse(kskc_degree);
                moveParams.GripRotation =
Single.Parse(kskc_rotation);
                moveParams.Grip = Single.Parse(kskc_mt);
                moveParams.Time = Single.Parse(time_to_move);


_fromWinformPort.Post(newFromWinformMsg(FromWinformMsg.MsgEnum.MoveT
oPosition, null, moveParams));
            }
catch
            {
                textBox1.Text = "Invalid Value";
            }

        }

privatevoid button1_Click(object sender, EventArgs e)
        {
            textBox1.Text = "KONTROL KLAVYE";
        }
```

```
privatevoid button2_Click(object sender, EventArgs e)
        {
secret yeniform = newsecret();
            yeniform.Show();
        }
privatevoid button3_Click(object sender, EventArgs e)
        {

_fromWinformPort.Post(newFromWinformMsg(FromWinformMsg.MsgEnum.Toppl
eBlock1, null, null));
        }

privatevoid button4_Click(object sender, EventArgs e)
        {

        }


    }

}
```



Şekil 7.8 Grafik Arayüzü

Grafik Analizi ve veritabanında verilerin kaydedilmesi için sisteme ait kodlar;

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using Microsoft.Dss.Core.Attributes;
using Microsoft.Robotics.Simulation;
using System.Xml;
using System.Data.SqlClient;

namespace ProMRDS.Simulation.LynxL6Arm
{
publicpartialclasssecret : Form
    {
public secret()
        {
            InitializeComponent();
        }
```

```
privateint arm_bilesen = 0;

string path = "C://Documents and Settings//kemal//Microsoft Robotics
Dev Studio 2008 R3//simulationenginestate.xml";

XmlDocument xml = newXmlDocument();

SqlConnection baglan = newSqlConnection(@"Data
Source=.\SQLEXPRESS;AttachDbFilename=C:\Documents and
Settings\kemal\Belgelerim\datalar.mdf;Integrated
Security=True;Connect Timeout=30;User Instance=True");
SqlCommand komut = newSqlCommand();
privatevoid secret_Load(object sender, EventArgs e)
        {
// TODO: This line of code loads data into the
'datalarDataSet1.ang_velocity' table. You can move, or remove it, as
needed.
this.ang_velocityTableAdapter.Fill(this.datalarDataSet1.ang_velocity
);
// TODO: This line of code loads data into the
'datalarDataSet1.posizyon' table. You can move, or remove it, as
needed.
this.posizyonTableAdapter.Fill(this.datalarDataSet1.posizyon);
            arm_bilesen = 8;


        }


privatevoid button1_Click(object sender, EventArgs e)
        {

            webBrowser1.Refresh();
            baglan.Open();
            komut = baglan.CreateCommand();
            komut.CommandText = "delete from posizyon";
            komut.ExecuteNonQuery();
            komut.CommandText = "delete from ang_velocity";
            komut.ExecuteNonQuery();
            Data_al();
            timer1.Start();
        }
int i = 0; int a = 0; int h = 0; int o = 0;
privatevoid Data_al() {

try
            {
                xml.Load(path);

XmlNode xnNode =
xml.DocumentElement.ChildNodes.Item(10).ChildNodes.Item(arm_bilesen)
.ChildNodes.Item(0).ChildNodes.Item(0);
                textBox1.Text = "" + xnNode.InnerText;

XmlNode xnNodePosition =
xml.DocumentElement.ChildNodes.Item(10).ChildNodes.Item(arm_bilesen)
.ChildNodes.Item(0).ChildNodes.Item(2).ChildNodes.Item(0);

            l_position.Items.Clear();
```

```
foreach (XmlNode xn in xnNodePosition)
            {
                l_position.Items.Add(xn.InnerText);

            }
            komut.CommandText = "insert into posizyon(x,y,z)
VALUES(" + float.Parse(l_position.Items[0].ToString()) + "," +
float.Parse(l_position.Items[1].ToString()) + "," +
float.Parse(l_position.Items[2].ToString()) + ")";
            komut.ExecuteNonQuery();
            chart3.Series["X"].Points.AddXY(i,
float.Parse(l_position.Items[0].ToString()));
            chart3.Series["Y"].Points.AddXY(i,
float.Parse(l_position.Items[1].ToString()));
            chart3.Series["Z"].Points.AddXY(i,
float.Parse(l_position.Items[2].ToString()));
            i++;

XmlNode xnNodeOrian =
xml.DocumentElement.ChildNodes.Item(10).ChildNodes.Item(arm_bilesen)
.ChildNodes.Item(0).ChildNodes.Item(2).ChildNodes.Item(1);

            l_oryantasyon.Items.Clear();

foreach (XmlNode xn in xnNodeOrian)
            {
                l_oryantasyon.Items.Add(xn.InnerText);

            }
            chart4.Series["X"].Points.AddXY(o,
float.Parse(l_oryantasyon.Items[0].ToString()));
            chart4.Series["Y"].Points.AddXY(o,
float.Parse(l_oryantasyon.Items[1].ToString()));
            chart4.Series["Z"].Points.AddXY(o,
float.Parse(l_oryantasyon.Items[2].ToString()));
            chart4.Series["W"].Points.AddXY(o,
float.Parse(l_oryantasyon.Items[3].ToString()));
            o++;
XmlNode xnNodeVelcty =
xml.DocumentElement.ChildNodes.Item(10).ChildNodes.Item(arm_bilesen)
.ChildNodes.Item(0).ChildNodes.Item(3);

            l_velocity.Items.Clear();

foreach (XmlNode xn in xnNodeVelcty)
            {
                l_velocity.Items.Add(xn.InnerText);

            }

            chart2.Series["X"].Points.AddXY(h,
float.Parse(l_velocity.Items[0].ToString()));
            chart2.Series["Y"].Points.AddXY(h,
float.Parse(l_velocity.Items[1].ToString()));
            chart2.Series["Z"].Points.AddXY(h,
float.Parse(l_velocity.Items[2].ToString()));
            h++;
```

```
XmlNode xnNodeAng_Velcty =
xml.DocumentElement.ChildNodes.Item(10).ChildNodes.Item(arm_bilesen)
.ChildNodes.Item(0).ChildNodes.Item(4);

            l_ang_vel.Items.Clear();

foreach (XmlNode xn in xnNodeAng_Velcty)
            {         l_ang_vel.Items.Add(xn.InnerText);
            }
            komut.CommandText = "insert into ang_velocity(x,y,z)
VALUES(" + float.Parse(l_ang_vel.Items[0].ToString()) + "," +
float.Parse(l_ang_vel.Items[1].ToString()) + "," +
float.Parse(l_ang_vel.Items[2].ToString()) + ")";
            komut.ExecuteNonQuery();
            chart1.Series["X"].Points.AddXY(a,
float.Parse(l_ang_vel.Items[0].ToString()));
            chart1.Series["Y"].Points.AddXY(a,
float.Parse(l_ang_vel.Items[1].ToString()));
            chart1.Series["Z"].Points.AddXY(a,
float.Parse(l_ang_vel.Items[2].ToString()));
            a++;
                }
catch (Exception hata) { textBox1.Text = hata.Message; }

        }
privatevoid comboBox1_SelectedIndexChanged(object sender, EventArgs
e)
        {
            arm_bilesen = comboBox1.SelectedIndex+8;
            i = 0; a = 0; h = 0; o = 0;
            chart1.Series["X"].Points.Clear();
            chart1.Series["Y"].Points.Clear();
            chart1.Series["Z"].Points.Clear();
            chart2.Series["X"].Points.Clear();
            chart2.Series["Y"].Points.Clear();
            chart2.Series["Z"].Points.Clear();
            chart3.Series["X"].Points.Clear();
            chart3.Series["Y"].Points.Clear();
            chart3.Series["Z"].Points.Clear();
            chart4.Series["X"].Points.Clear();
            chart4.Series["Y"].Points.Clear();
            chart4.Series["Z"].Points.Clear();
            chart4.Series["W"].Points.Clear();
            Data_al();
        }

privatevoid timer1_Tick(object sender, EventArgs e)
        {
            webBrowser1.Refresh();

ang_velocityTableAdapter.Fill(datalarDataSet1.ang_velocity );

            posizyonTableAdapter.Fill(datalarDataSet1.posizyon);

            Data_al();
        }

privatevoid button2_Click(object sender, EventArgs e)
        {
```

```
            baglan.Close(); i = 0; a = 0; h = 0; o = 0;
            timer1.Stop();
        }

publicvoid set_droptime() {
            textBox1.Text = "Pos:" + i + "-AçıcsalHIZ :" + a + "-Hız
:" + h + "-Orynt :" + o;        }

    }
}
```

SONUÇ

Robot kolunun uygulama alanlarının oldukça geniş olması çalışmanın da önemini arttırmaktadır. Robot kolu uygulamaları hemen hemen bütün üretim sektörlerinde seri üretimin ve üretim hatalarının iyileştirilmesi adına tercih edilmektedir. Geniş uygulama alanına sahip bu tür teknolojiler üzerine yapılan çalışmalar genellikle pahalı ve gerçekleştirilmesi zaman alan çalışmalar olmaktadır. Bu sebeple uygulamaların bilgisayar ortamında simülasyonu oldukça önem taşımaktadır.

Çalışma, uygulamada gerçekleştirilmesi maliyetli ve zaman alacak bir sistemi bilgisayar ortamına taşımış ve simülasyonunu gerçeklemiştir. Çalışmada IP adresi 10.7.1.252 sunucu üzerinden uygulama hizmete geçirilmiş ve laboratuvar ortamında sunucuya bağlı 37 bilgisayardan kullanılmıştır. Uygulama boyunca yapılan her hareket verisi XML formatında alınabilmekte bu sayede yörüge tayini, engel/hedef takibi ve benzeri uygulama çalışmalarının değerlendirilmesi için kullanılacak algoritmaların çıkarımları bakımından veriler sunabilmektedir.

Çalışmanın devamında robot kolu uygulamaları için geliştirilen algoritmalarınsimülasyon üzerinde kullanılabileceği, hareket analizlerinin yapılabileceği görülmektedir.

Çalışma laboratuar ortamında sunucu-istemci taraflı çalışan bir uygulama olduğu için konu ile ilgili lisans ve lisansüstü eğitimlerin yapılmasında oldukça faydalı deneysel bir ortam sunmaktadır.

KAYNAKLAR

1. Alper Bayrak, Beş Eksenli Bir Robot Kolunun Simülasyonu ve Kontrolü,Yüksek Lisans Tezi,Gazi Üniversitesi Fen Bilimleri Enst., Mayıs 2007

2. Alper Bayrak, Müzeyyen Sarıtaş, Beş Eksenli Bir Robot Kolunun Simülasyonu ve Engel/Hedef Takibi, Elektrik Mühendisleri Odası

3. JohannesSchützner, RainerPollak, Dr.ThomasBräunl,RoboSim - A Robot ManipulatorSimulator, University of Stuttgart

4. A.H.Overmars, D.J. Toncich, Application of DSP technologytoclosed-position-loopservodrivesystems, IJ of AMT, v11, nl, 27-33, 1996

5. L.Sciavicco, B.Siciliano, Modellingand Control of Robot Manipulators, p121-151, McGraw-HillBook Co.1996

6. H.Asada, J.E.Slotine,Robot Analysis and Control, Wiley, New York,1986

7. C.S. Lee, Robot ArmKinematics, Dynamics, Control, Computer, Vol.15,no. 12, pp. 62-80, 1982

8 http://www.kuka-robotics.com/usa/en/products/software/educational_framework/arm_tutorials/PS_Content_Arm1.htm

9. http://www.kuka-robotics.com/usa/en/products/software/educational_framework/arm_tutorials/PS_Content_Arm2.htm

10. http://www.kuka-robotics.com/usa/en/products/software/educational_framework/arm_tutorials/PS_Content_Arm3.htm

11. http://www.kuka-robotics.com/usa/en/products/software/educational_framework/arm_tutorials/PS_Content_Arm4.htm

12. Kyle Johns, Trevor Taylor, Professional Microsoft Robotics Developer Studio, 2008

ÖZGEÇMİŞ

1982yılında Elazığ'da doğan Ali Kemal DURKAYA, orta ve lise öğrenimini sırasıyla Namık Kemal İlkokulu ve Elazığ Anadolu Lisesinde tamamlamıştır. 2000 yılında kazandığı Fırat Üniversitesi Mühedislik Fakültesi Bilgisayar Mühendisliği Bölümünü 2005 yılında başarıyla bitirmiştir.

2005 yılında Erciyes Üniversitesi , 2007 yılında Bozok Üniversitesi Meslek Yüksekokulunda öğretim elemanı olarak göreve başlamış, 2009 yılında Microsoft Corporation Referal Sisteme girmiş, 2013 yılında özel sektöre geçerek ilgili kurumda ICT Koordinatörü ve Müdürü olarak görev yapmaktadır.

İletişim Bilgileri

Adres : Güzelyalı Mahallesi Sahil Yolu Sokak Biotek Vektör Kontrol LTD. ŞTİ.

Pendik/İSTANBUL

Telefon: (216) 445 64 17

Faks: (216) 373 83 49

E-posta: kemal.durkaya@biotekhasere.com