GEBZE YÜKSEK TEKNOLOJİ ENSTİTÜSÜ
MÜHENDİSLİK VE FEN BİLİMLERİ ENSTİTÜSÜ

ALTI EKLEMLİ BİR ROBOTUN WINDOWS İŞLETİM SİSTEMİ ALTINDA DENETİM SİSTEMİ TASARIMI

Ufuk ÖZBAY
YÜKSEK LİSANS TEZİ
BİLGİSAYAR MÜHENDİSLİĞİ
ANABİLİM DALI

TEZ DANIŞMANI
Yrd. Doç. Dr. Erkan ZERGEROĞLU

GEBZE
2006

Ufuk ÖZBAY'ın tez çalışması, GYTE Mühendislik ve Fen Bilimleri Enstitüsü Yönetim Kurulu2nun 20.07.2006 tarih ve 2006/26 sayılı kararıyla oluşturulan jüri tarafından Bilgisayar Mühendisliği Anabilim Dalı'nda Yüksek Lisans Tezi olarak kabul edilmiştir.

JÜRİ

ÜYE : Doç. Dr. Selim SİVRİOĞLU

ÜYE : Yrd. Doç. Dr. Erkan ZERGEROĞLU

(Tez Danışmanı)

ÜYE : Yrd. Doç. Dr. İlyas KANDEMİR

ONAY

GYTE Mühendislik ve Fen Bilimleri Enstitüsü Yönetim Kurulu'nun __/__/_____ tarih ve ____/__ sayılı kararı

ÖZET

TEZ BAŞLIĞI: Altı Eklemli Bir Robotun Windows İşletim Sistemi Altında Denetim Sistemi Tasarımı

YAZAR ADI: Ufuk ÖZBAY

Bu çalışmamızda Enstitümüz bünyesinde, robotik manipülatörler için açık sistem ve kod standartlarına uyularak geliştirilmekte olan Matlab®/Simulink® tabanlı gerçek zamanlı benzetim ve denetim sistemi tanıtılmaktadır. Sistemimiz akademik araştırmalarda sıklıkla kullanılan 6 döner ekleme sahip Puma 560 robotları üzerinde denenerek başarı ile uygulanmıştır.

SUMMARY

THESIS TITLE: Design of a Control System for a 6 Link Manipulator Under Windows Operating System

AUTHOR: Ufuk ÖZBAY

In this work, a Matlab®/Simulink® based real time simulation and control environment, developed under open system protocols is presented. The performance and the feasibility of the proposed system has been successfully tested on a commonly used 6 link revolute robot the Puma 560 manipulator

TEŞEKKÜR

GYTE'deki öğrenim hayatım boyunca tüm çalışmalarımda beni destekleyen, tez çalışmamın başından sonuna kadar bana devamlı yol gösteren danışmanım Sayın Yrd. Doç. Dr. Erkan ZERGEROĞLU' na; çalışmalarım boyunca desteklerini benden esirgemeyen sevgili çalışma arkadaşlarım Ümit Ali Tektaş ve Hüsnü Türker Şahin'e, ve tabiî ki bugün burada olmamı sağlayan, hayatım boyunca desteklerini esirgemeyen, sadece anne baba kardeş değil birer dost olan sevgili aileme; sonsuz teşekkür ederim.

İÇİNDEKİLER DİZİNİ

Sayfa

SİMGELER VE KISALTMALAR DİZİNİ

RCCL	Robot Control C Library
SRTK	Simulink Robotic Toolkit

ŞEKİLLER DİZİNİ

TABLOLAR DİZİNİ

1 GİRİŞ

1.1 Giriş

Robotik araştırmalar, birçok farklı bilim dalının temel yöntemlerinden yararlanılmasını gerektirir. Örneğin robot kontrolü üzerine çalışan araştırmacı, sistemi denetlemek için öncelikle gerekli donanımı kurmalı, robotu kontrol edecek denetleyici algoritmasını oluşturmak için, gerekli yazılımları hazırlamalı ve buna uygun bir kullanıcı ara yüzü tasarlamalıdır. Bu işlemleri daha da zorlaştıran bir etmen de; asıl olarak endüstri için tasarlanmış robot manipülatörlerinin çoğunlukla kendi kontrol algoritmaları dışında çalışmamak üzere üretilmiş olmalarıdır. Bu tür platformlarda var olan fonksiyonlar ile kullanılan donanım parçalarını birleştirmekte kullanıcı etkinliği azdır. Dolayısı ile kullanıma sunulan robot denetleme ve hareketlendirme programları üretici firmalar tarafından sağlanmışlar, özel uygulamalar gerçekleştirmek için ise üretici firmalardan ek yazılım paketleri almak veya geliştirmeyi üretici firmaya yaptırmak zarureti ortaya çıkmaktadır. Bahsi geçen sorunu aşabilmek için birçok akademik kullanıcı yüksek seviyeli dillerden yararlanarak kütüphaneler geliştirilmiştir. Bu tür yazılımların en başarılılarından biri, Unix tabanlı iş makinelerinde çalışan RCCL[1] (Robot Control C Library) dir. RCCL, robotik hesaplamalar için gerekli matris işlemlerini ve geometrik hesaplamaları içeren C rutinleriyle birlikte birçok robot için ileri ve ters kinematik çözümlerlerine içermektedir. RCCL in diğer bir özelliği de Unimation firması tarafından Puma robotlari için hazırlanan orijinal VAL kontrolörünün[2] nasıl revize edilerek araştırmacının kendi programını yükleyebileceği hususunda bilgiler içermesidir[3]. Ancak RCCL sadece bir kaç platform üzerinde çalışabilir ve kütüphane dosyasının boyutunun büyük ve karmaşık olması yeni uygulamaların paralelinde değiştirilmesini zor kılmaktadır. RCCL'nin kod karmaşıklığını azaltmak için bir kaç çalışma[4-5] yapılmış olmasına karşın bu çalışmalar da tasarlandıkları platformlar dışına taşınamamışlardır.

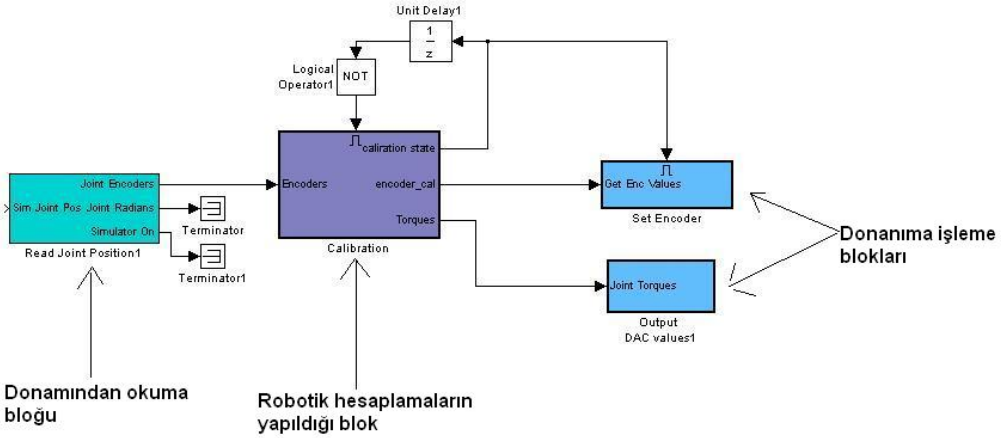
Önceki yazılım araçlarından yeterince verim alamayan birçok araştırmacı Matlab tabanlı robotik uygulamalar geliştirmişlerdir. [6]'da Pires Win32 işletim sistemlerinde otomasyon ekipmanlarını ve bazı endüstriyel robotları kontrol etmek için kullanılan MATROBCOM olarak adlandırılan Matlab tabanlı bir kütüphane

tanıtmıştır. Ne yazık ki, MATROBCOM işlemleri gerçek zamanlı olarak yerine getirememektedir. [7]'de ise Matlab/Simulink ortamında geliştirilen SRTK (Simulink Robotic Toolkit) platformunu tanıtmaktadırlar. Önceki sistemlere göre en büyük farkı Gerçek Zamanlı Linux (RTLT)[8] ve Windows hedefleri (RTWT)[9] ile uyumlu olması olan STRK, RT-Linux ve Windows–98 işletim sistemlerinde gerçek zamanlı olarak çalıştırılabilinir. Ancak STRK tek bir bloktan oluşmasından dolayı farklı uygulamalar için şekillendirilmesi, geliştiricilerinin açıklamalarının aksine zorluklar içermektedir.

Bu çalışmamızda, Matlab/Simulink tabanlı yeni bir robot denetim ve kontrol kütüphanesi geliştirdik. Geliştirdiğimiz kütüphane STRK'nın içerdiği tüm kontrol kiplerini içermesinin yanında sadece Windows–98 için değil diğer bütün win32 işletim sisteminde de (Windows NT/2000/XP), RTWT üzerinden gerçek zamanlı çalıştırılabilmektedir. Çalışmamızın uygulanabilirliğini gösterebilmek amacı ile bir Puma560 robotunu açık sistem ve kod standartlarına uygun olarak orijinal kontrolloründen ayrılarak modifiye edilmiş, modifikasyonlar sonunda genel amaçlı bir giriş/çıkış kartı yardımı ile bilgisayara bağlanan robot denetleyicisinin bütün işlemleri, hazırlanan kütüphaneler yardımı ile yapılabilmiştir.

1.2 Simulink Blok Diagramlarının Organizasyonu

Yazılan robot kontrol kütüphanesi robot kontrol kiplerini yerine getirebilmek için çeşitli hesaplamaları ve robotun donanımına ulaşmamızı sağlayan çeşitli simulink bloklarından oluşmuştur. Şekil 1.1'de örnek olarak sunulan blok diyagram(*Başlangıç Kalibrasyon Kipi*) 3 ana kısımda incelenebilinir. Burada koyu yeşil renkli bloklar direkt donamından okuma bloklarını göstermektedir. Temel görevi robot donanımına ulaşarak encoder değerleri, potansiyometre değerleri gibi verileri elde etmektir. Açık mavi renkli bloklar ise donanıma işleme bloklarıdırlar. Bu bloklar ayrıntılı olarak bölüm 2'de anlatılınacaktır. Mor renkli bloklar çeşitli robotik hesaplamaları yerine getirmektedir.



Şekil 1.1: *Simulink blok diyagramı organizasyonu*

Tezimizin geri kalanı şu biçimde özetlenebilir: Bölüm 2'de genel amaçlı bir robot denetleyicisinde bulunması gereken kipler aktarılacak, Bölüm 3'de bu kiplerin gerçekleştirilmesi için geliştirilen kütüphane tanıtılıcak, Bölüm 4'de geliştirilen kütüphane vasıtası ile gerçekleştirilen demo programlar anlatılacaktır. Bölüm 5'de oluşturulan çıktı ve sonuçlar irdelenip projemizin ileri safhalarında yapılması planlananlar aktarılacaktır.

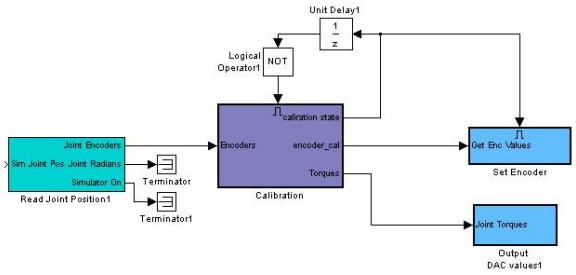
2 ROBOT DENETLEYİCİSİNDE BULUNMASI GEREKLİ PARÇALAR

Genel bir robot denetleyicisi, çalıştırılmaya başladığı zamanda robot eklem konumlarını bilebilmeli (Başlangıç Kalibrasyon Kipi), gerek eklem uzayında gerekse iş uzayında verilen iki veya daha fazla nokta arasında yörünge sinyallerini üretip yumuşak bir sekilde takip edebilmeli (Eklem ve İş Uzaylarında Denetleme Kipleri), kullanıcısının hareket ettirmeyi planladığı noktaları robota öğretmesine olanak sunabilmeli (Konum Öğretme Kipi) ve robota eklenebilecek aygıtlarla kolay entegre edilebilmelidir. Bunların yanı sıra dışardan hazırlanıp robota takip ettirilmesi planlanan yörüngelerin robotun hünerli iş sahası içerisinde kalıp kalmayacağının, (verilen yörüngenin robotun takip etmesine uygun biçimde tasarlanıp tasarlanamadığının) robot sistemine zarar vermeden, test edebilmeye olanak verebilmesi (Benzetim/Simülasyon Kipi) özellikle akademik ortamlarda gerekli bir özelliktir. Bu bölümün kalan kısımlarında bu kiplerin özelliklerin Puma tipi robotlarda nasıl uygulanabileceği ve bu çalışmada hangi formlarda sunulduğundan bahsedeceğiz.

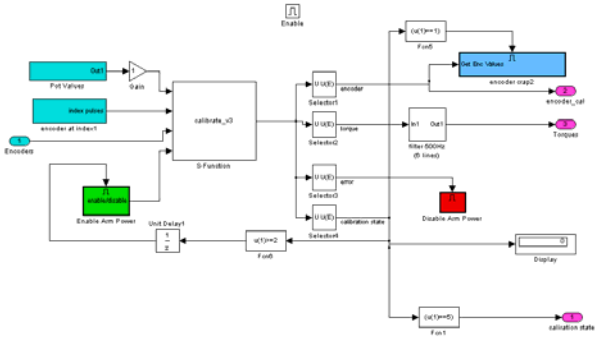
2.1 Başlangıç Kalibrasyon Kipi

Denetim altında tutulmak istenen her sistemde olduğu gibi robot kontrol sistemlerinde de başlama koşullarının (robot denetleyicisi aktif duruma geçtiğinde eklem konumlarının) bilinmesi veya bir şekilde tespit edilmesi gerekir. Bu bilgi özellikle kendi kol ağırlıklarını denetimsiz taşıyamayan robotlarda sonraki daha karmaşık işlemlerin gerçekleştirilmesinde hataların yaşanmaması için son derece önemlidir. Gerçekleştirilecek robotik görevler kalibrasyonun sonucunda elde edilen encoder ayar bilgilerinin kesinliğine göre başarım gösterirler. Eğer kalibrasyon doğru şekilde yapılmazsa diğer kiplerde kullanılan (ileri ve geri kinematik, vb.) hesaplamalar yanlış sonuç verecek dolayısıyla robotun çalışmasında ciddi hatalara yol açacaktır. Son yıllarda kullanılmaya başlanan robotlar pozisyon bilgilerini tutabilen pozisyon ölçerler (encoderlar) kullanmaktadırlar. Ancak çalışmamızda temel olarak kullandığımız Puma robotları, güç devrelerine verilen besleme sinyali kesildiğinde pozisyon bilgisini saklayamayan encoderlar içermektedirler. Başlangıç pozisyon bilgisini algılayabilmenin tek yolu her ekleme yerleştirilmiş olan

potansiyometrelerdir. Potansiyometre değerleri hassas ölçüm yapabilmek için uygun değillerdir, bu yüzden sadece başlangıçta encoderlerin gerçek değerlerine atanmasında kullanılmalı; daha sonra ise denetim için gerekli eklem pozisyonlarının encoder sinyallerinden yararlanılarak bulunmaları gerekmektedir. Bu işlemi yapabilmek için Unimation firmasının önerdiği ve RCCL'de de uygulanan bir yöntem kullanılmıştır. Bu yöntemde potansiyometre değerleri ile encoder değerlerinin birbirleri ile doğrusal bir ilişki içinde bulundukları farz edilir, robot encoder değerleri tam olarak bilinen bir pozisyona getirilir. Birçok araştırmacı bu pozisyonu robotun bütün eklemlerinin yerçekiminden etkilenmediği Hazır (Ready) pozisyonu olarak seçmişlerdir. Özenle ve gerekirse hassas ölçeklendirme aletlerinin yardımı ile bu pozisyona getirilen robotun potansiyometre değerleri okunarak kayıt edilir. Sonra her eklem için robot bütün erişebildiği noktalara hareket ettirilerek encoder iz (index pulse) noktalarında potansiyometre değerleri ile bunlara karşılık gelen encoder değerleri not edilir. Bütün değerler okunduktan sonra iki değer arasında doğrusal bir interpollasyon denklemi kurulup her eklem için bu değerler bir konfigürasyon dosyasında saklanır. Robot denetleyicisi her açıldığında eklemler ilk encoder iz noktasına kadar hareket ettirilerek bu noktadaki potansiyometre değerleri ve interpollasyon denklemi parametrelerinden de faydalanılarak encoder gerçek değerleri saptanır ve bu değerler encoder yazmaçlarına işlenirler. Bu yöntem sayesinde Başlangıç Kalibrasyon Kipinin sonunda robot eklemlerinin pozisyonları tam olarak tespit edilebilmektedir. Şekil 1 ile Şekil 2'de Başlangıç Kalibrasyon Kipi için hazırlanan blok diyagramlar sunulmuştur. Şekil 1'de sunulan blok diyagramda koyu yeşil renkli bloklar direkt donamından okuma, mor renkli bloklar hesaplama ve açık mavi renkli bloklar ise donanıma işleme bloklarıdırlar. İşlemlerin yapıldığı mor bloğun detaylı biçimi ise Şekil 2'de sunulmuştur.



Şekil 2.1: *Başlangıç Kalibrasyon Kipi*



Şekil 2.2: Kalibration Mode Bloğu İç Yapısı

2.2 Eklem Uzayında Denetleme Kipi

Eklem Uzayı Denetleme Kipinin amacı robotu başlangıç pozisyonundan alıp kullanıcının eklem parametreleri (açıları) ile belirlediği istenilen başka bir pozisyona en az takip hatası ile yönlendirmektir. Bu amaç için öncelikle bu iki nokta arasında yumuşak bir yörünge sinyali oluşturulması gerekir. Uygulamanın yumuşak (çokça türevlenebilir) olabilmesi için yörünge oluşturucuda istenilen eklem profili

$$\Delta_S = q_S - q_İ \qquad (1)$$
$$q_d = \Delta_S\left(6\bar{t}^5 - 15\bar{t}^4 + 10\bar{t}^3\right) + q_İ$$

denklemi kullanılarak oluşturulmaktadır. Denklem (1) de bulunan q_S eklemlerin ulaşması gereken eklem açılarını $q_İ$ eklemlerin başlangıç pozisyonlarını içeren

vektörleri ifade etmek için kullanılmış, $\bar{t}$ ise [0,1] aralığında normalize edilmiş zaman sinyalini temsil etmektedir. Denklem (1)'den de anlaşılabileceği gibi seçilen yörünge oluşturma fonksiyonu ikinci dereceden türevlenebilir olup robotun hem hız hem de ivme profillerinin sürekli olmalarını garanti eder. Takip edilmesi istenilen yörünge sinyali oluşturulduktan sonra denetim performansını belirleyen hata sinyali

$$e = q_d - q \tag{2}$$

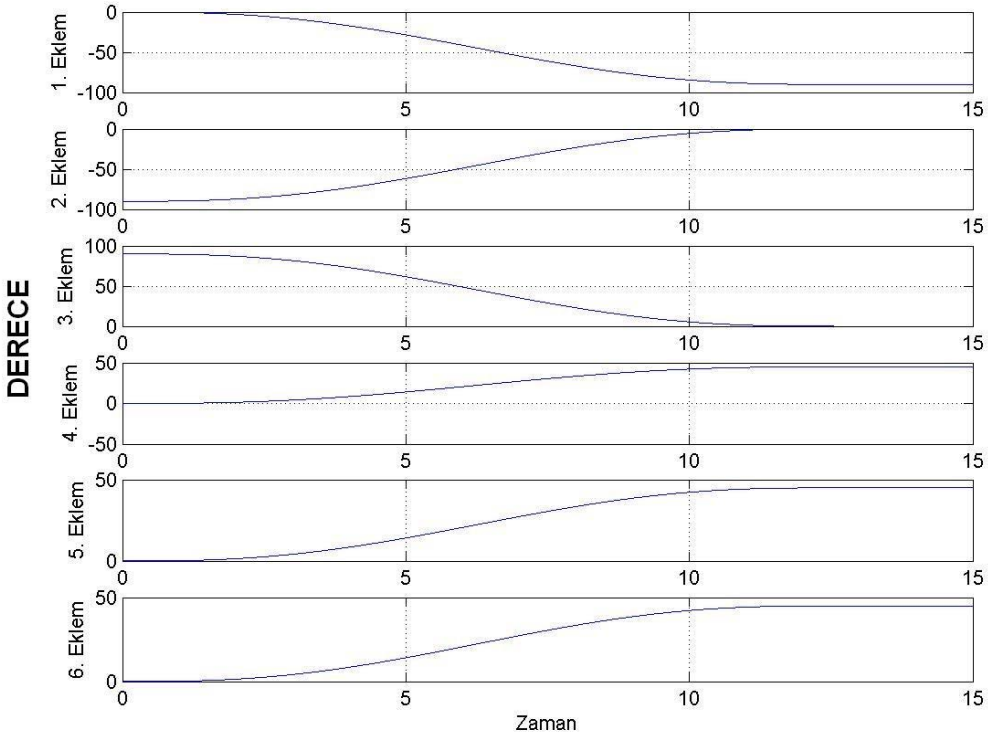
biçiminde tanımlanıp, eklem yörüngelerinin en uygun şekilde takibi için PD+yer çekimi formundaki denetleyici ile aşağıda verildiği şekilde tanımlanabilir

$$\tau = \ddot{q}_d + k_p e + k_d \frac{d}{dt}(e) + G(q) \tag{3}$$

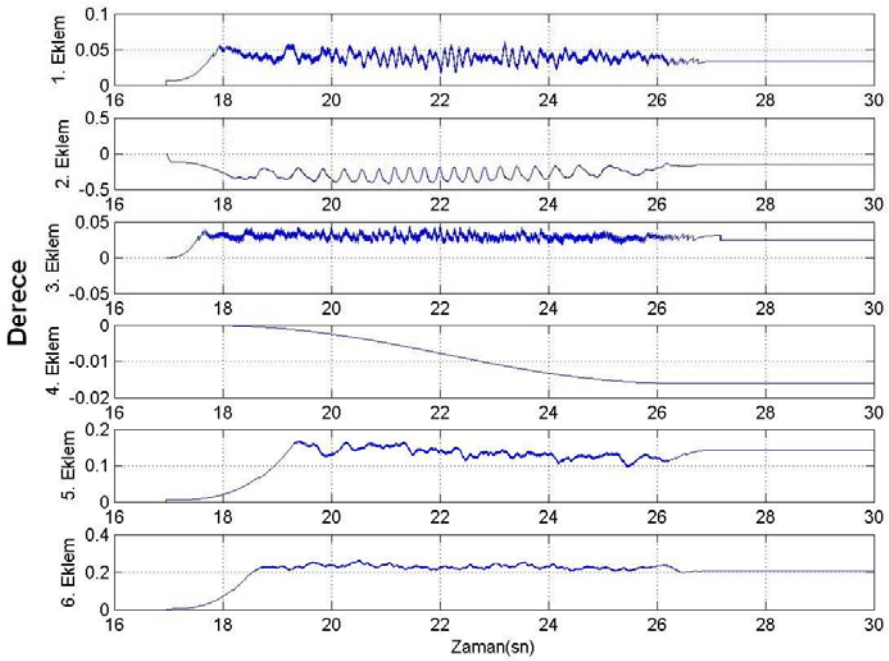
.

Denklem (3)'de verilen $\ddot{q}_d$, (1) ile verilen yörünge sinyalinin ikinci türevini, k_p, k_d pozitif kontrol kazançlarını, *G(q)* robotun yer çekim etkilerini içeren vektörü ve τ eklemlere uygulanacak açısal güç vektörünü temsilen kullanılmışlardır.

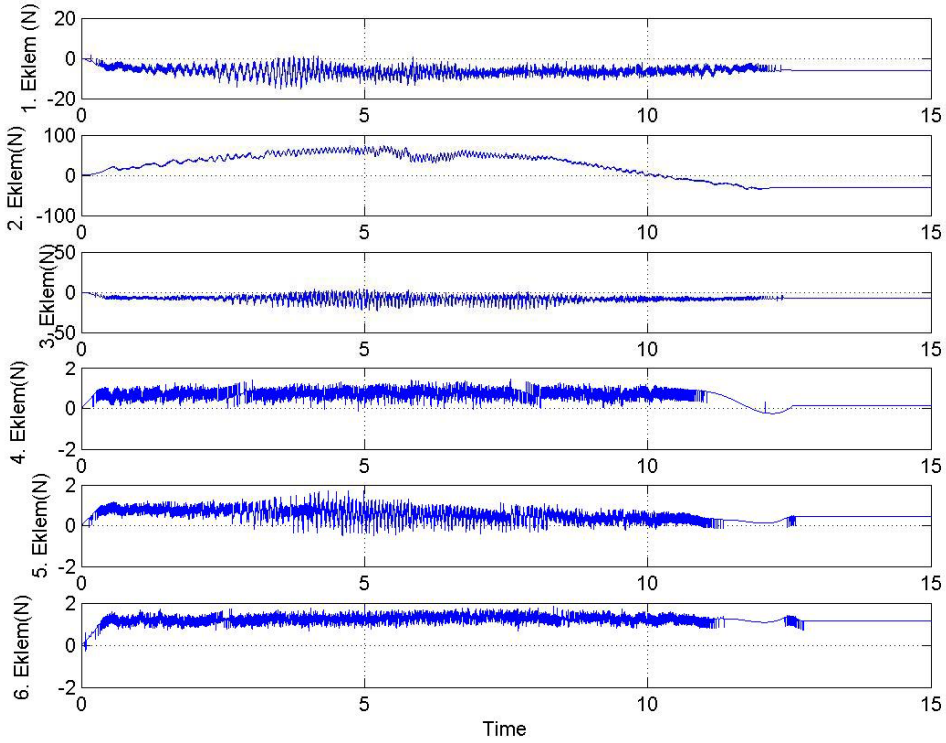
Denklem (1) ile hesaplanan örnek bir yörünge Şekil 2.3 de verilmiş olup robotun yörünge takip hatası da Şekil 2.4'de motorlara uygulanan tork ise şekil 2.5'de sunulmuştur.



Şekil 2.3: Robotun takip etmesini istediğimiz yörünge.



Şekil 2.4: Robotun yörünge takip hatası.



Şekil 2.5: Robotun eklem motorlarına uygulanan tork.

2.3 Kartezyen(İş) Uzayında Denetleme Kipi

Kartezyen Uzayında Denetim Kipi kullanıcıya robotun varacağı konumu, iş uzayındaki robot-ucu koordinatları ile girilmesini sağlar. Böylece kullanıcı her eksenin varış açılarını ayrı ayrı hesaplamak zorunda kalmadan direkt olarak üç boyutlu uzayda robotun gitmesi gereken konumu verebilir. Genel robotik yaklaşımında Kartezyen Uzayında Denetleme için 2 yöntem bulunmaktadır: (i) pozisyon hesaplamalarının direkt iş uzayında yapılarak denetleyicinin iş uzayında tasarımı ve eklemlere uygulanacak kuvvet vektörünün robot jakobyanı kullanılarak bu denetleyici çıkışından hesaplanması; (ii) iş uzayında verilen pozisyonların ters kinematik formülasyonu ile eklem uzayına transferinden sonra eklemlere uygulanacak kuvvet vektörünün tasarımı. Çalışmamızda her iki yöntemde birbirlerinden bağımsız olarak ve başarı ile uygulanmıştır.

Birinci yöntemin uygulamasında, hesaplama sadeliği ve kullanım kolaylığı göz önüne alınarak manipülatörün oryantasyonu, yörünge boyunca denetleyici tarafından belirlenerek kullanıcıdan sadece istenilen son konum bilgisi (p_S) girilmesini istenir. Denklem (1)'e benzer şekilde Kartezyen koordinatlarda bir pozisyon profili (p_d) elde edilir. Elde edilen yörünge ve robotun o anki yörünge sinyallerinden yararlanılarak, istenilen yörüngeyi takip edebilmesi için robot ucuna uygulanması gereken kuvvetler

$$f(p) = K_p e_x + K_d \frac{d}{dt}(e_x) \tag{4}$$

şeklinde hesaplanır. Burada verilen K_p, K_d pozitif denetim katsayıları, e_x

$$e_x = p_d - p \tag{5}$$

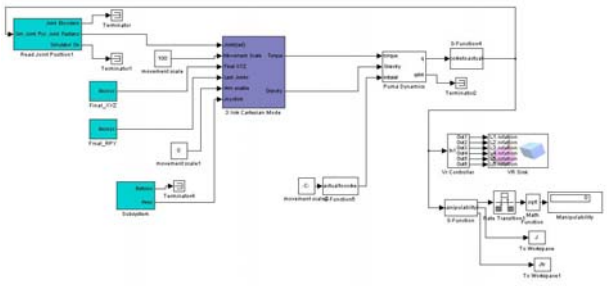
biçiminde tanımlanan robot ucu takip hata sinyali ve aynı denklemde yer alan p ise robotun ilk üç eklemi kullanılarak elde edilen uç pozisyon vektörüdür (Puma robotlarında son üç eklem sadece robot ucu oryantasyonunu değiştirdiğinden robotun ileri kinematik formülasyonu, sadece bu kipe özgü olarak sadece ilk üç eklem kullanılarak hesaplanmıştır). Denklem (4) ile elde edilen eklem ucu güç vektöründen yararlanılarak robotun eksenlerine uygulanacak tork

$$\tau = J^T f + G(q) \tag{6}$$

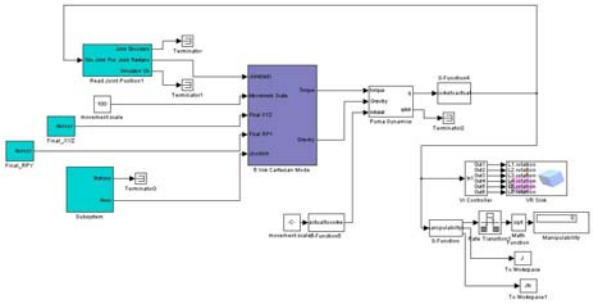
formülasyonu ile hesaplanabilir. Denklem (6) da geçen *J(q)* robot Jakobyenini belirtmek üzere kullanılmıştır.

İş uzayı denetim kipinde yer alan ikinci yöntemde kullanıcıdan robotun gitmesini istediği kartezyen pozisyonla beraber robot ucunun verilen pozisyondaki

oryantasyonu 3-parametreli minimum anlatımı biçiminde istenir. Verilen son nokta kullanılarak iş uzayında bir yörünge tayin edilir ve bu yörüngenin eklem uzayı karşılığı robot hareket ederken (on line olarak) ters kinematik formülasyon kullanılarak hesaplanır. Denetim denklem (3)'de tanımlanan kontroller kullanılarak yapılır. Bu kipte karşılaşılabilecek en büyük sorun ters kinematik formülasyonunun birden fazla çözüm üretmesidir (Puma tipi robotlar için her kartezyen konum başına sekiz adet eklem uzayında karşılık bulunabilir). Bunun oluşturabileceği yan etkiyi ortadan kaldırabilmek için kullanım algoritmamızda, her eklem için o eklemin bir önceki açısal konumuna göre değişimi en küçük olanı alınmıştır. Şekil 2.6 ile Şekil 2.7'de Kartezyen Uzayında Denetleme Kipi için hazırlanan blok diyagramlar sunulmuştur.

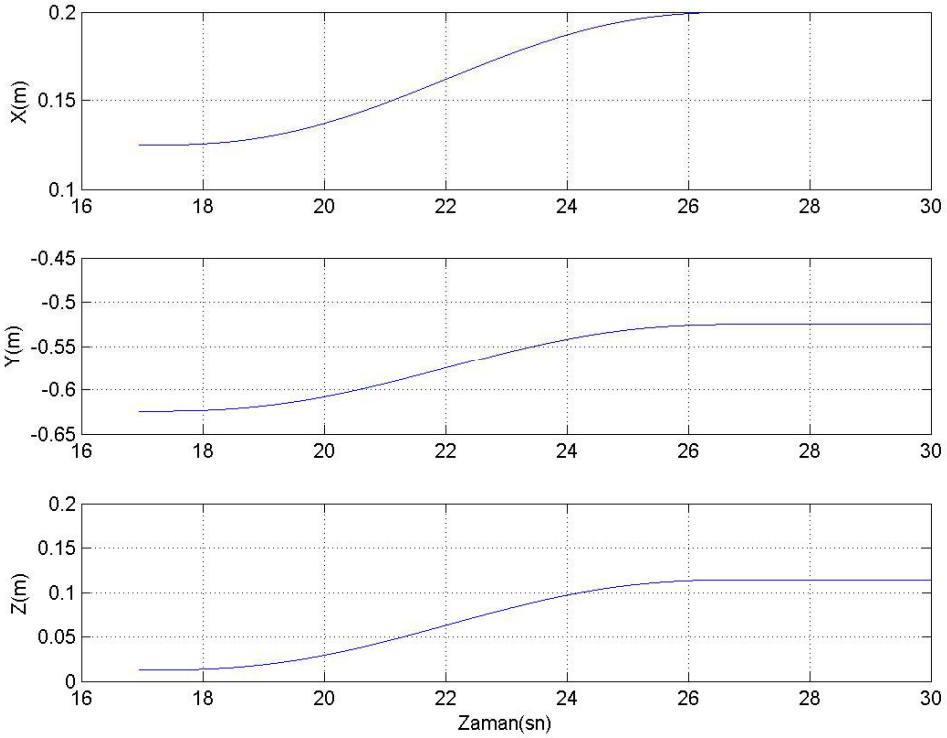


Şekil 2.6:Kartezyen Denetim Kipi Modu 1

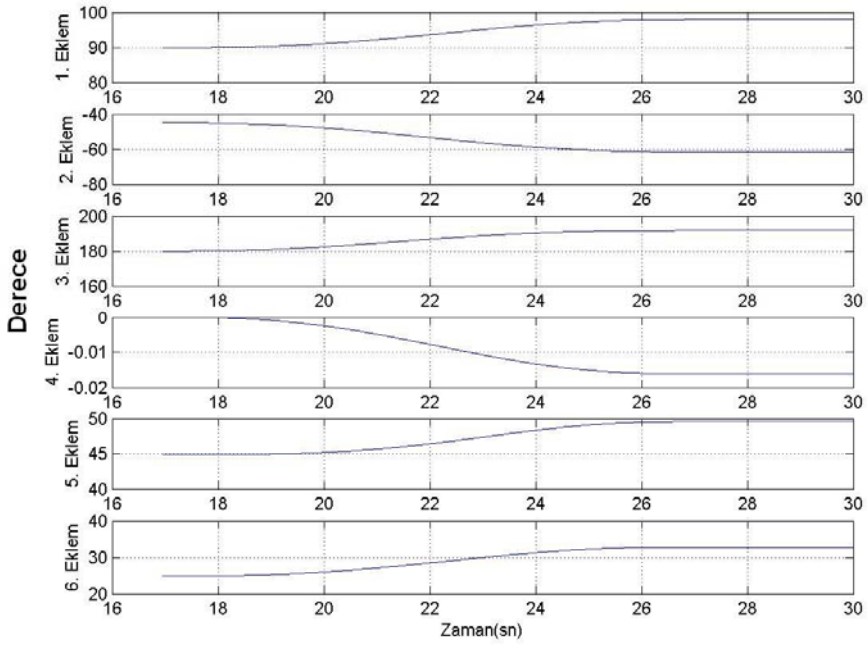


Şekil 2.7:Kartezyen Denetim Kipi Modu 2

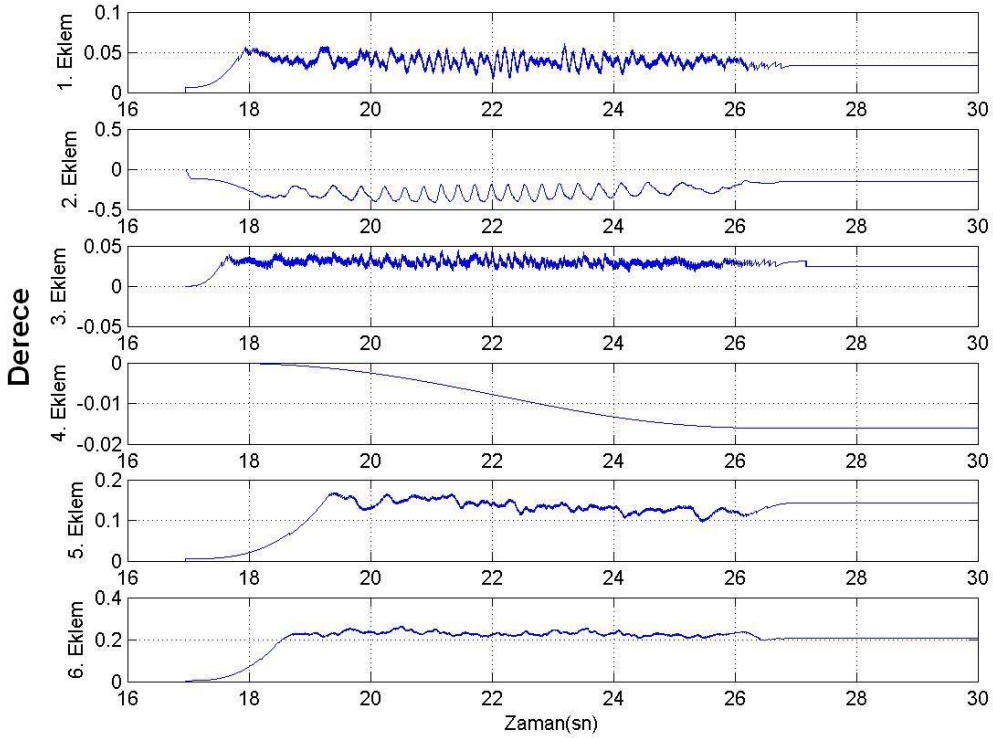
Kartezyen uzayında hesaplanan örnek bir yörünge Şekil 2.8'de verilmiş olup şekil 2.9 da ters kinematik denklem yardımıyla hesaplanan eklem motorlarının yörüngesi verilmiştir. Robotun yörünge takip hatası Şekil 2.10'da, motorlara uygulanan tork ise şekil 2.11'de verilmiştir.



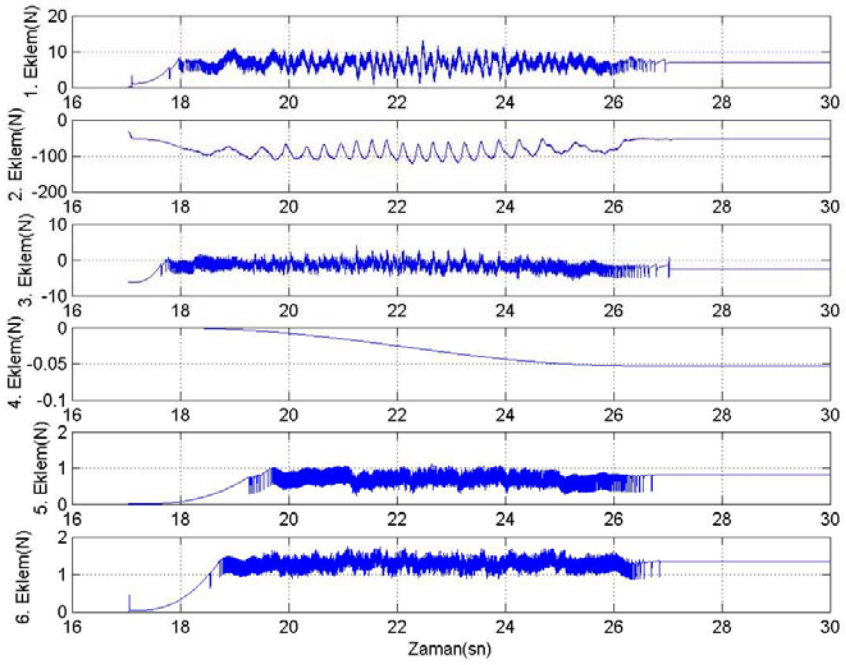
Şekil 2.8:Kartezyen Uzayında hesaplanan Robot yörüngesi



Şekil 2.9:Ters kinematik hesaplamalar sonucu bulunan robot eklem yörüngesi.



Şekil 2.10: Robotun yörünge takip hatası.

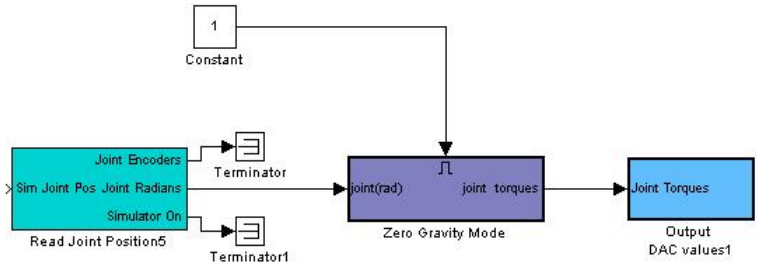


Şekil 2.11: Robotun eklem motorlarına uygulanan tork.

2.4 Konum Öğreneme Kipi

Tezimizin önceki bölümlerinde anlatılan kiplerin büyük bölümü, robotun gitmesi planlanan konumun (eklem ya da kartezyen koordinatlarında) bir şekilde robot denetleyiciye kullanıcı tarafından verilmesine ihtiyaç duymaktadır. Belirlenmesi gereken hedef noktalar, özellikle ince iş yapması gereken robotlar için, büyük bir hassasiyette saptanmalı ve ana iş başlamadan önce robotun belirlenen bu noktalara istenilen hassasiyette gidebildiğinden emin olunmalıdır. Ancak kullanıcının robotun her parametresini bilip iş alanını milimetrik olarak ölçeklendirmesi ve robot kinematiğini kullanarak, hata yapmadan bu pozisyonları saptaması çok külfetli ve matematiksel olarak ağır bir görevdir.

Uygulama sırasında bu matematiksel yaklaşım yerine kullanıcı robotu manüel olarak hareket ettirir ve robotun ucunu iş alanında gelmesi gereken komuma getirip gerekli oryantasyonda tutar. Bir başka kişi bu pozisyonu kaydeder. Yapılması planlanan iş için gerekli noktaların hepsi kaydedildikten sonra, robot bu noktalar arasında hareket ettirilir. Konum öğrenme kipi bu görevin rahat yapılabilmesi için tasarlanmıştır. Bu kipte robotun denetleyicisine sadece robotun her eklemine etkiyen yer çekimi kuvvetini dengeleyebilecek miktarda kuvvet uygulanmıştır. Böylece robotun ucu manüel olarak rahatlıkla hareket ettirilebilir. Robotun bulunması istenilen konum saptandıktan sonra robot konumunu koruyacak ve kullanıcı ikinci bir kişiye ihtiyaç duymadan robotun bu konumunu kayıt edebilmektedir. Bu basit ancak efektif çözümün blok diyagramı Şekil 2.12'de verilmiştir.



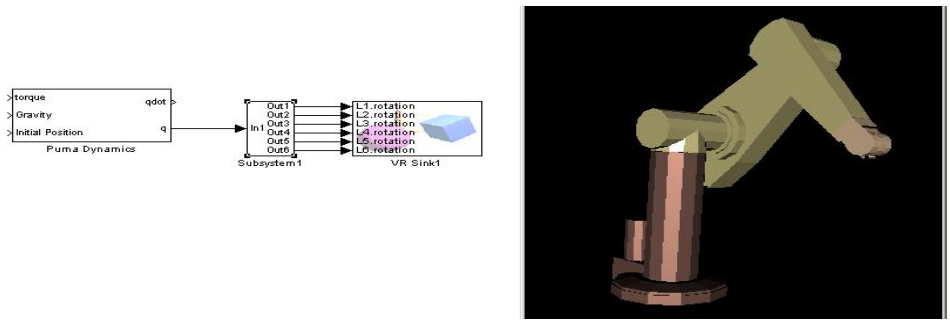
Şekil 2.12: Yer çekimi dengeleme ve Öğrenme Kipi

2.5 Benzetim/Simülasyon Kipi

Gerek robot için tasarlanmış hareketlerin gerçek sisteme verilmeden önce görülüp uygun düzeltilmelerin yapılabilmesi, gerekse bir manipülatör düzeneği olmadan da (yörünge tayini, ileri denetim problemleri de dâhil olmak üzere) robotik problemler üzerinde çalışılabilmesi için benzetim kipi önemlidir. Endüstriyel robotlarda bulunan benzetim kipleri ne yazık ki sadece yörünge takibi hususunda kullanıcıya bilgi sunabilmekte, ancak sistemin dinamiğini içermedikleri için tasarlanabilecek denetim davranışlarını göstermekte yetersiz kalmaktadırlar. Çalışmamızda sunulan Benzetim/Simülasyon Kipi robot dinamiğini de entegre etmiş ve denetim tasarımı ve uygulamaları amaçlı olarak da kullanılabilir durumdadır. Kipte kullanılan Robot Dinamiği bloğumuz en basit biçimde

$$\ddot{q} = M^{-1}\left[\tau - G(q) - C(q,\dot{q})\dot{q}\right] \tag{7}$$

ile verilen denklemi gerçek zamanda çözmekte ve ard arda iki kez integral alarak robotun o an için pozisyon, hız ve ivme vektörlerini hesaplamaktadır. Denklem (7)'de belirtilen *M(q)*, $C(q,\dot{q})$ 6x6 matris fonksiyonları, sırası ile Puma robotunun pozitif tanımlı, simetrik (dolayısı ile her zaman tersi alınabilir) atalet matrisi ile centripedal ve coriolis etkilerini içerisinde barındıran matrisi anlatmak üzere kullanılmışlardır. Aynı denklemde geçen τ eklemlere uygulanan güç $G(q)$ vektörünü, ise yer çekimi etkilerini içermektedirler. Verilen bu matrislerin parametreleri [10]'dan direkt olarak alınarak koda entegre edilmişlerdir. Dinamik hesabı sonunda elde edilen eklem pozisyon vektörü görsel öğelerden yararlanılarak kullanım kolaylığı sunabilmek amacıyla VRML kodu kullanılarak hazırlanan 3-Boyutlu Puma modeline verilerek, benzetim çalışırken kullanıcının robotun nasıl hareket ettiğini sanal olarak görmesi sağlanmıştır. Şekil 2.13'de Benzetim Kipi blok diyagramı ile 3 boyutlu robot hareketinin oluşumu gösterilmektedir.



Şekil 2.13: Benzetim/Simülasyon Kipi

2.6 Diğer Özel Amaçlı Kipler

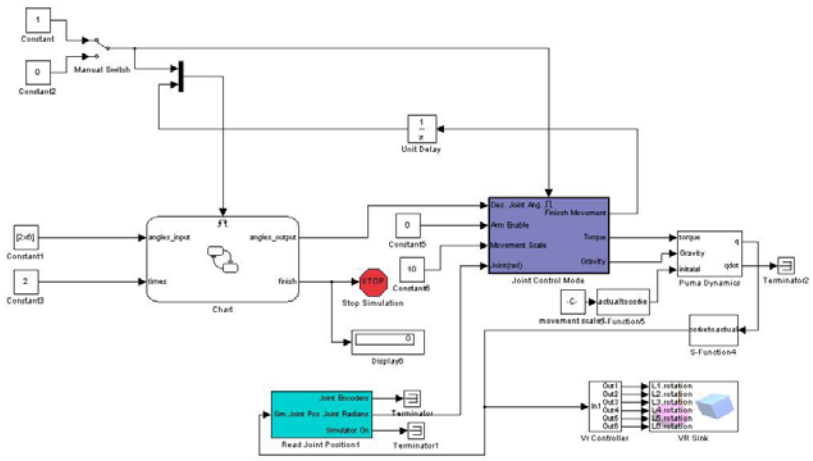
Giriş bölümünde bahsedildiği üzere bir robot denetleyicinin kullanıcıya tanıması gereken bütün çalışma kipleri, yukarıda verilen modlardan ibarettir. Bunların üzerine yeni kipler eklenmesi ve robota yeni sensorler (görsel geri besleme, kuvvet/güç sensorleri gibi) ilave edilmesi ve bunların denetimi genellikle yüksek maliyet gerektirdiğinden, kullanımı üretici firmanın inisiyatifine bırakılmıştır. Ayrıca eklenen yeni birimlerle sadece yörünge tayinine eklemeler yapılabilmekte, yani kinematik denetim olanağı sunulmaktadır. Kendi denetim algoritmalarını uygulamak isteyen araştırmacılar için bu kabul edilemez bir durumdur.

Bu alt bölümümüzde oluşturduğumuz kütüphane yardımı ile genel robot denetleyicilerin aksine yeni kipler ile sensor bağlantıları kolaylıkla oluşturulup eklenebileceğini gösterebilmesi amacıyla basit bir kaç kip seçilerek uygulanmıştır. Bu kipler aşağıda özetlenmektedir:

2.6.1 Çoklu Hareket Kipi

Kullanıcı, bazı durumlarda robotun öğretilen her pozisyona birebir gitmesini istemeyebilir. “Al ve yerleştir” türü robot hareketleri buna bir örnektir. Bu tip operasyonlarda kullanıcı robotun belirli bir cismi bir noktadan diğer noktaya taşımasını, taşıma işlemi yaparken de robotun çevresine veya kendisine zarar vermemesi için belirli ara noktalara yönelmesini bekler. Robotun seçilen ara noktalara tam olarak uğraması işin yapılması için gerekli değildir. Bu kipte hem

eklem hem de kartezyen denetleme kiplerinden farklı olarak robot kullanıcının seçtiği çoklu noktalardan sadece istenilen bir kaçına uğramakta diğer noktalara belirlendiği kadar yaklaşıp bir sonraki noktaya yönelmektedir. Böylece robota yumuşak bir yörünge kazandırılmış olmakta ve robot hareketi daha yumuşak (keskin ve kesikli olmadan) olarak gözlenebilmektedir. Bu kipin uygulanması ile kullanıcının istediğinde robot yörünge profilini istediği biçimde kolayca değiştirebileceği kanıtlanmış olmaktadır. Şekil 2.14'de çoklu hareket kipinin uygulamalarından bir tanesi (yörünge oluşturma türüne bağlı olarak birden fazla çoklu hareket kipi başarı ile uygulanmıştır) gösterilmektedir.



Şekil 2.14:Çoklu hareket kipi

2.6.2 Yönetme Kolu ile Denetim Kipi

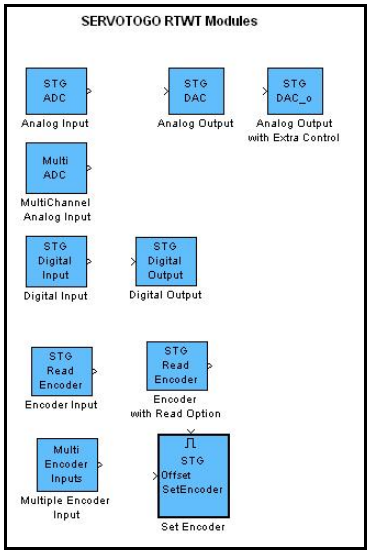
Bazı uygulamalarda robota hareket öğretmek için robot iş alanına erişim mümkün olmayabilir. Böyle durumlarda robotun, PC ye direkt bağlanabilen basit bir Yönetme Kolu (USB Joystick) ile uzaktan kumanda edilmesi kullanışlı olur düşüncesi ile bu kip oluşturulmuştur. Bu kip kartezyen kontrol kipinin farklılaştırılmış halidir Yönetme kolu yardımıyla robot ucunu hareket ettirmektir. Bu kullanışlı kip hem robot kullanımını eğlenceli kılmakta, hem de geliştirdiğimiz robot denetleyicisinin ucuz, kullanımı kolay kişisel bilgisayara takılabilen sade araçlarla kullanımına uygunluğunu göstermektedir.

3 GELİŞTİRİLEN ROBOT KONTOL KÜTÜPHANESİ

Yukarıda belirtilen kiplerin her birinin oluşturulmasında kullanılan bloklar birer Simulink® kütüphanesi biçiminde tasarlanmış olup temel olarak 3 bölümde sunulabilirler. Bunlar (i) Donamına ulaşma blokları, (ii) Genel robot algoritmaları (kinematik, ters kinematik, yörünge tayin blokları gibi), (iii) Gerçek zamanlı işlemlerde kullanılmak üzere Puma robotları için optimize edilmiş ve sadece Puma 560 robotlarına has bloklar. Oluşturulan kütüphanenin kurulumu EK 1'de anlatılmıştır.

3.1 Donanıma Ulaşma Blokları

Puma robotları, orijinal durumlarında VAL denetleyicileri ile çalışırlar. Ancak sunumumuzun çeşitli bölümlerinde söz ettiğimiz üzere, piyasada bulunan birçok robot kontrolörleri gibi, VAL denetleyicilerinin kullanımında da robota sadece dışardan yörünge verilebilir. Kendi tasarladığımız denetleyicileri kullanabilmek için VAL denetleyicinin ana kısmı (mikro bilgisayar ve denetleyici kartları) sökülerek bütün giriş-çıkış sinyalleri ayırt edildi. Bu işlem daha önce bir kaç robotik grubu tarafından[3-4-5 ve 12] da yapılmış olmasına karşın, her robotta bir kaç değişiklik içerebilmektedir. Sonuç olarak robotu bilgisayara bağlıyıp denetleyebilmek için 6 analog giriş (her linkte bulunan potansiyometre değerlerini okuyabilmek için), 6 analog çıkış (her bir ekleme hesaplanan güç değerlerini verebilmek için), 6 encoder-iz okuyucu (eklem pozisyonlarını okuyabilmek için) ve 4 adet dijital giriş/çıkış'a sahip bir giriş/çıkış kartı gerekmektedir. Çalışmalarımızda "Servo To Go" firmasının[13] ISA Model II kartı kullan ılınmıştır. Bu kart için MATLAB/Simulink ortamı için sürücü kütüphanesi oluşturulmuş (bakınız Şekil 3.1), kartın Puma robotunun orijinal VAL kontrolörlünde bulunan motor sürücü devrelerine erişebilmesi için de bir geçiş kartı tasarlanmıştır. Bu kart EK3'de verilmiştir.



Şekil 3.1: Servo To Go Kütüphanesi

3.2 Servo To Go Kütüphanesi (stglib)

3.2.1 Sayısal Giriş Çıkışlar (Digital Input, Digital Output)

"D*igital Input*" Sayısal bir sinyali okumak, "*Digital Output*" dışarıya sayısal bir işaret vermek için kullanılır. Çalışmamızda eklem motorlarında güç olup olmadığını okumak için "*Digital Input*", motorlara ilk güçlerini vermek için de "*Digital Output*" kullanılmıştır.

Şekil 3.2 "*Digital Input*" bloğunun maskesi gösterilmiştir.

Blok parametreleri:

"Digital Input" ve "Digital Output" da bulunan parametrelerin tanımı:

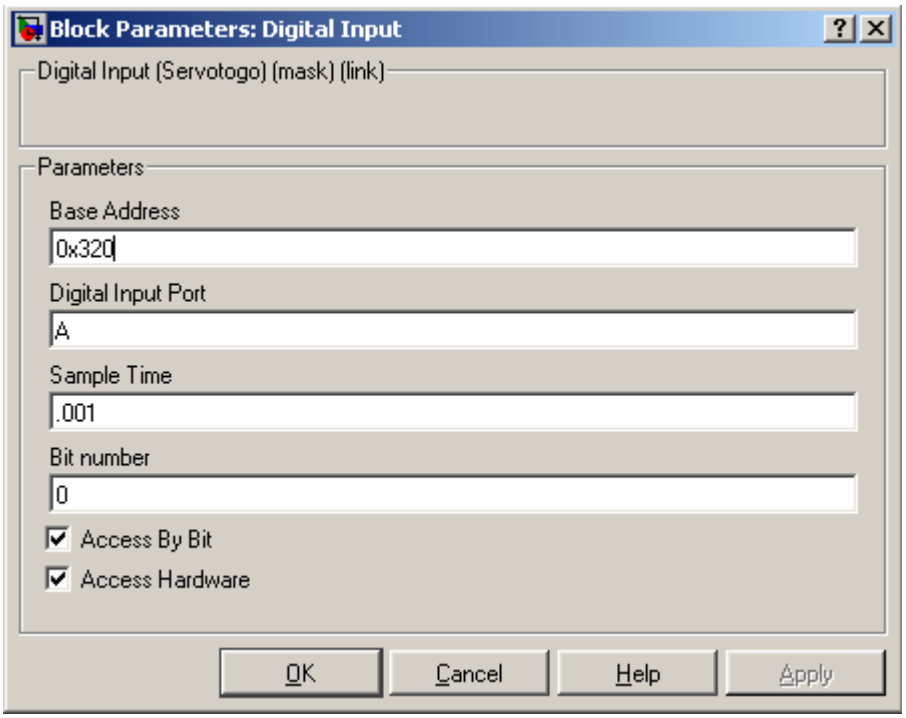
Base Address: Bu servo to go kartının bilgisayarda bulundu adresidir

Digital Input(Output) Port: bu alana kullanılacak port girilir. Bit number parametresi ile de kaçıncı çıkışın kullanılacağı belirlenir Servo to go kartının da A,B,C ve de olmak üzere 4 tane 8 er bitlik sayısal çıkış ve giriş portu vardır

Access By bit: Bu onay kutusu yardımıyla kartın sayısal portlarına teker teker ulaşmamız sağlanır.

Access Hardware: Bu onay kutusu ile program direkt olarak servo to go donanımına ulaşır ve işlemlerini yerine getirir aksi taktirde donanıma ulaşmaz.

Sample Time: Örnekleme zamanını belirtir, bu alan yazılan simulink programının örnekleme zamanı ile aynı olmalıdır.

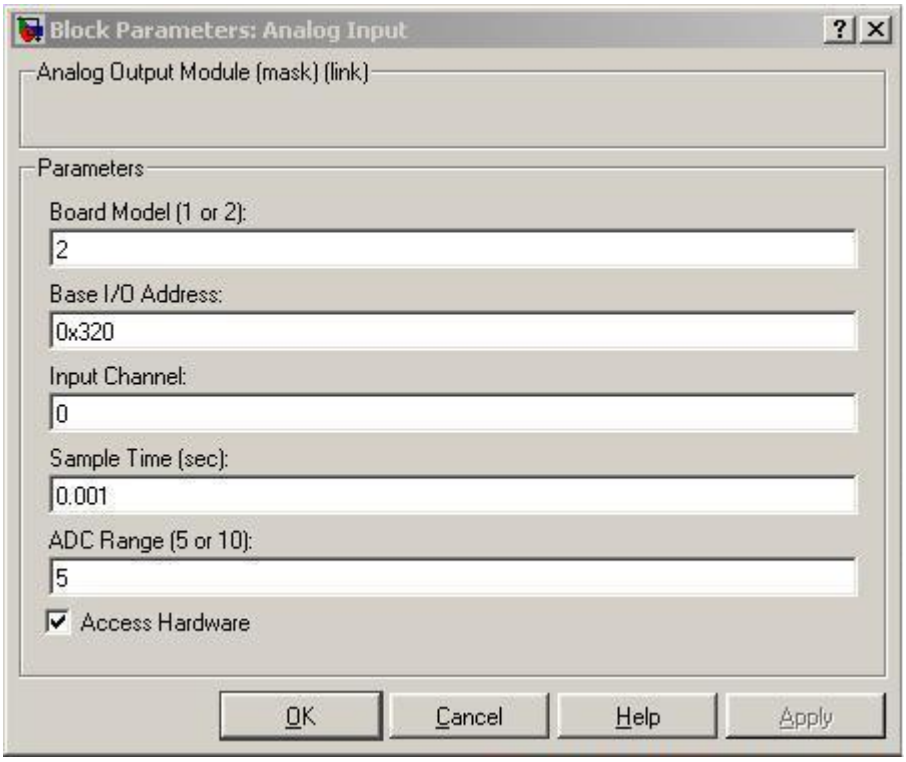


Şekil 3.2: "Digital Input" bloğunun maskesi

3.2.2 Analog Giriş (Analog Input, Multi Channel Analog Input)

Bir analog kaynağın değerini okumak için kullanılır. Biz çalışmamızda puma560 robotunun her ekleminde bulunan potansiyometre değerlerini okumak için kullandık. "*Analog Input*" aynı anda bir kanalın değerini okuyabilirken "*Multi Channel Anolog Input*" birden fazla kanaldaki değeri okuyabilir.

Şekil 3.3 "*Analog Input*" bloğunun maskesi gösterilmiştir.



Şekil 3.3: "Analog Input" bloğunun maskesi

"*Analog Input*" ve "*Multi Channel Analog Input*" bloklarında bulunan parametrelerin tanımı:

Board Model: Servo to go kartının iki modeli vardır. Biz "MODEL II" kartını kullandığımızdan buraya 2 değerini girdik.

Input Channel: = Kartın üzerinde kaçıncı analog girişi kullandığımızı belirler. 0 - 7 arası bir değer alır.

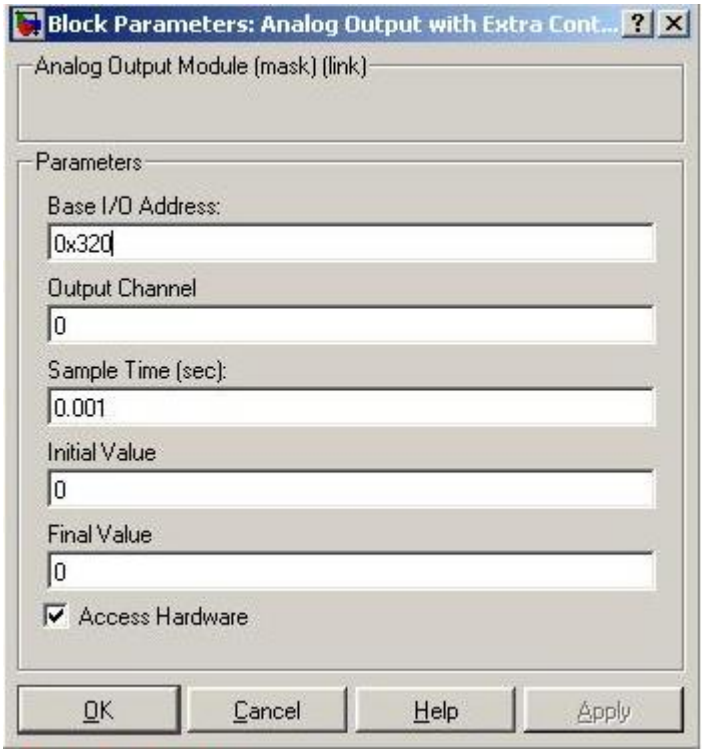
ADC Range: Servo to Card ±5v ve ±10v olucak şekilde ayarlanabilinir

3.2.3 Analog Çıkış (Analog Output, Analog Output With Extra Control)

Bilgisayardan dışarıya ±10 volt arası bir değer vermek için kullanılır. Çalışmamızda robot eklemlerinde bulunan DC motorları sürmek için kullanıldı. "*Analog Output*" başlangıç ve bitiş değerini 0 olarak kabul eder "*Analog Output With*

Extra Control" farkı olarak analog çıkışların başlangıç ve bitiş değerleri parametre olarak ayarlanmasına izin verir.

Şekil 3.4 "*Analog Output with Extra Control*" bloğunun maskesi gösterilmiştir.

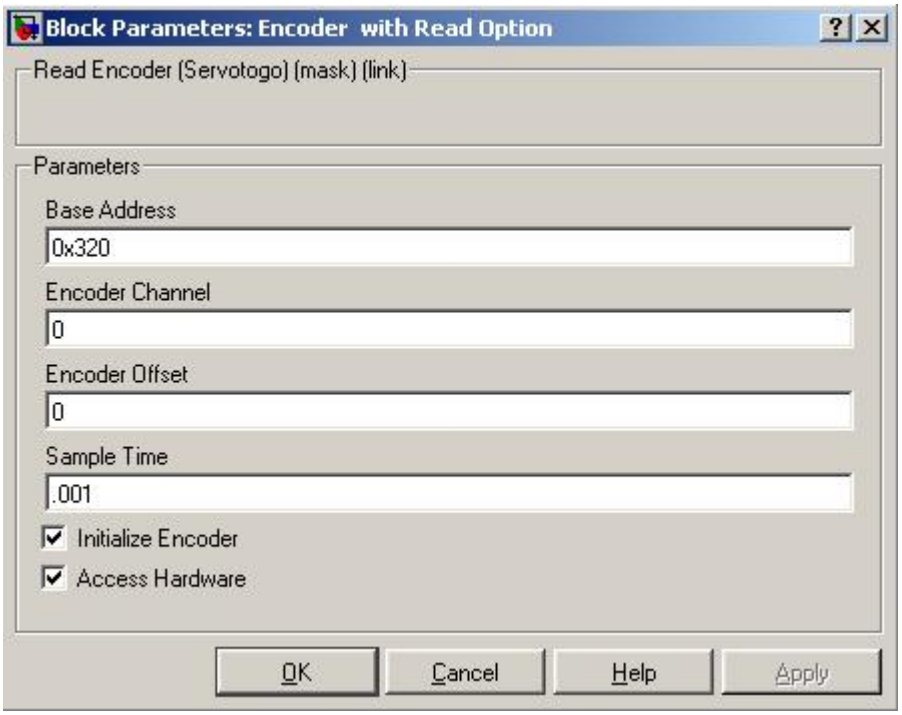


Şekil 3.4: "Analog Output with Extra Control" bloğunun maskesi

3.2.4 Encoder Girişleri (Encoder Input, Encoder Input With Read Option)

Bağlı olduğu motorun encoder değerlerini elde etmek için kullanılır. "*Encoder Input*" bloğu ilk başlangıç anında Servo to Go kartında bulunan encoder yazmaçlarını sıfırlar. "*Encoder Input With Read Option*" ile bu yazmaçların ilk değerlerinin korunması sağlanabilinir.

Şekil 3.5 "*Encoder with Read Option*" bloğunun maskesi gösterilmiştir.



Şekil 3.5: "Encoder with Read Option" bloğunun maskesi

Encoder offset: Encoder'ın ilk değeri buraya girilebilinir. Eğer *"Initialize Encoder"* onay kutusu seçili ise buraya girilen değer encoder'ın ilk değeri olarak alınır.

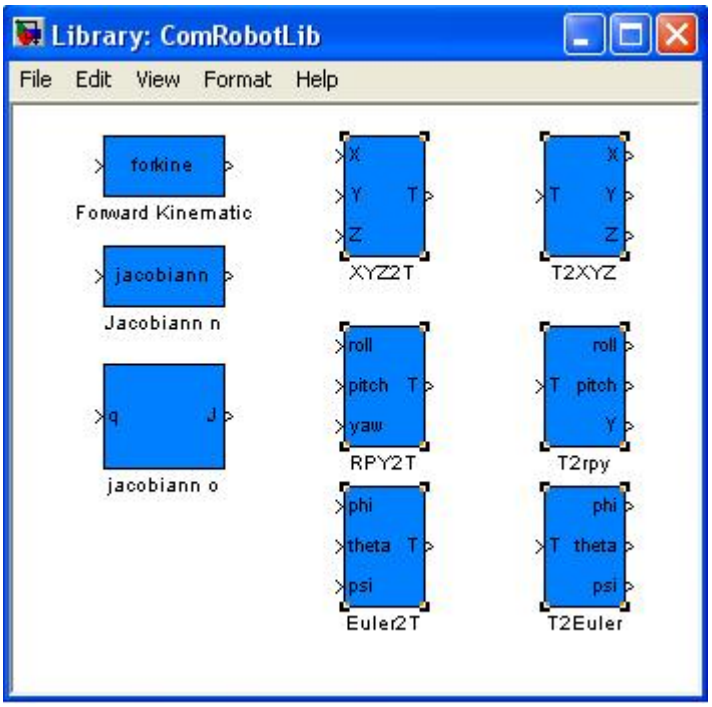
3.2.5 Encoder değerlerini ayarlamak (Set Encoder)

Servo to Go kartında bulunan encoder yazmaçlarına ulaşmamızı ve istenilen değerleri yazmamızı sağlar.

3.3 Genel Robot Kütüphanesi

Bu kütüphanede genel robotik hesaplamalarında kullanılabilecek homojen transformasyon, döndürme ve ötelemelerle ilgili hesaplarla birlikte yine genelleştirilmiş bir robot objesi (çalışmamızda [14] de kullanılan robot obje tipi kullanılmıştır) için kullanılabilecek ileri kinematik, Jacobyan ve bazı dönüşüm blokları yer almaktadır. [14]'den farklı olarak uygulamamız tamamen C-Mex S-fonksiyonları kullanılarak yapılmıştır. Bu simulink ortamında gerçek zamanlı

çalışma yapabilmemizi sağlamaktadır. Şekil 3.6'da genel robot kütüphanesi verilmiştir.



Şekil 3.6: Genel Robot Kütüphanesi

3.3.1 İleri Kinematik (Forward Kinematic)

Verilen manipülatör modeli için girilen eklem değişken değerleri için robot uçunun konumunu ve yönelimini hesaplar. Eklem değişkenleri, eklemin döner olması durumunda uzuvlar arasındaki açı, eklemin kayar olması durumunda uzuv uzanma miktarıdır.

3.3.2 Jacobian Hesaplanması (Jakobyen n, Jakobyen 0)

"Jakobyen n" verilen manipülatör modelinin giriş olarak girilen eklem değişkenleri değerleri için sonlandırıcı koordinat çerçevesinde "Jakobyen O" ise temel koordinat çerçevesinde Jakobyeni hesaplar.

Jakobyen işlemi eklem uzayı ile koordinat uzayı arasında denklem 8'deki gibi tanımlanan matematiksel bir ifadedir

$$ {}^{n}\dot{X} = {}^{n}\dot{j}(q)\dot{q} \tag{8} $$

Jakobyen yardımı ile kartezyen koordinatlarda hesaplanan eklem ucu gücü

$$ \tau = {}^{n}\dot{j}(q)^{n}F \tag{9} $$

eklem güçlerine dönüştürülebilinir.

3.3.3 XYZ2T, T2XYZ Blokları

XYZ2T bloğu giriş olarak aldığı kartezyen koordinatları çıkış olarak homojen dönüşüm matrisi biçiminde verir. *T2XYZ* bloğu ise homojen dönüşümden Kartezyen koordinatlara geçer.

3.3.4 RPY2T, T2RPY Blokları

RPY2T bloğu roll / pitch / yaw açılarından homojen dönüşüm matrisine geçer. Roll / Pitch / Yaw açıları sırasıyla z ekseni etrafında Φ açısı kadar, y ekseni etrafında θ açısı kadar ve x ekseni etrafında Ψ açısı kadar dönmelere karşılık gelmektedir. Burada dönme matrisi şu şekilde elde edilir

$$ R = \begin{bmatrix} c\phi c\theta & -s\phi c\psi + c\phi s\theta s\psi & s\phi s\psi + c\phi s\theta c\psi \\ s\phi c\theta & c\phi c\psi + s\phi s\theta s\psi & -c\phi s\psi + s\phi s\theta c\psi \\ -s\theta & c\theta s\psi & c\theta c\psi \end{bmatrix} \tag{10} $$

Çıkış da

$$\left[\begin{array}{ccc|c} & & & 0 \\ & R & & 0 \\ & & & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right]_{4x4} \tag{11}$$

homojen matris formundadır.

T2RPY bloğu ise homojen dönüşümden roll/pitch/yaw açılarına geçer. Homojen dönüşüm matrisi tekil ise

$$\begin{aligned} roll &= 0 \\ pitch &= a\tan\left(\frac{-T[3,1]}{T[1,1]}\right) \\ yaw &= a\tan\left(\frac{-T[3,1]}{T[2,2]}\right) \end{aligned} \tag{12}$$

tekil değilse

$$\begin{aligned} roll &= a\tan\left(\frac{T[2,1]}{T[1,1]}\right) \\ sp &= \sin(roll) \\ cp &= \cos(roll1) \\ pitch &= a\tan\left(\frac{-T[3,1]}{cp * T[1,1] + sp * T[2,1]}\right) \\ yaw &= a\tan\left(\frac{sp * T[1,3] - cp * T[2,3]}{cp * T[2,2] - sp * T[1,2]}\right) \end{aligned} \tag{13}$$

olarak hesaplanır. Burada *T* giriş matrisidir (homojen dönüşüm matrisi).

3.3.5 Euler2T, T2Euler Blokları

Euler2T, Euler açılarından homojen dönüşüme geçer. Euler açıları sırasıyla z ekseni etrafında Φ açısı kadar, y ekseni etrafında θ açısı kadar, tekrar z ekseni

etrafında Ψ açısı kadar dönmelere karşılık gelmektedir. **cos** = c , **sin** = s olmak üzere dönme matrisi şu şekilde elde edilir :

$$
\begin{aligned}
&c = \cos; \\
&s = \sin; \\
&R = \begin{bmatrix} c\phi c\theta c\psi - s\phi s\psi & -c\phi c\theta s\psi - s\phi c\psi & c\phi s\theta \\ s\phi c\theta c\psi + c\phi s\psi & -s\phi c\theta s\psi + c\phi c\psi & s\phi s\theta \\ -s\theta c\psi & s\theta s\psi & c\theta \end{bmatrix}
\end{aligned}
\tag{14}
$$

Çıkış da

$$
\left[\begin{array}{ccc|c} & & & 0 \\ & R & & 0 \\ & & & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right]_{4x4} \tag{15}
$$

homojen matris formundadır.

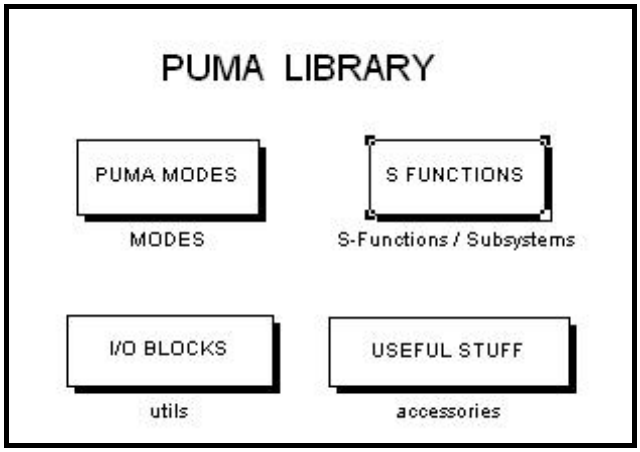
T2EULER bloğu ise homojen dönüşümden Euler açılarına

$$
\begin{aligned}
&\phi = a\tan\left(\frac{T[2,3]}{T[1,3]}\right) \\
&sp = a\tan(\phi) \\
&Cp = a\tan(\phi) \\
&\theta = a\tan\left(\frac{cp * T[1,3] + sp * T[2,3]}{T[3,3]}\right) \\
&\psi = a\tan\left(\frac{sp * T[1,1] + cp * T[2,1]}{-sp * T[1,2] + cp * T[2,2]}\right)
\end{aligned}
\tag{16}
$$

formülü ile geçer.

3.4 Puma Robotu Özel Kütüphanesi

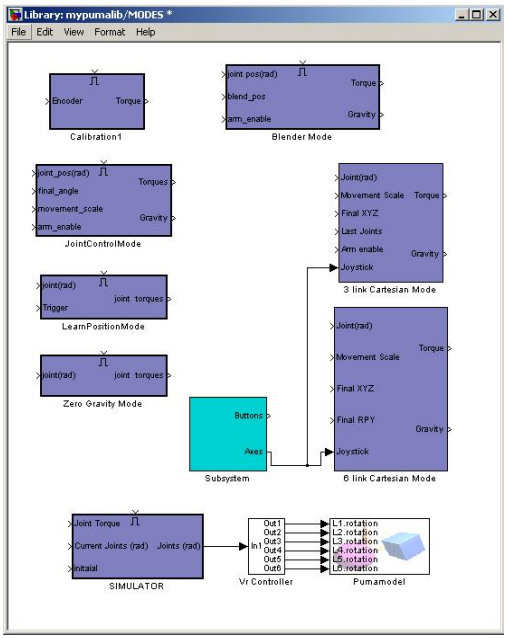
Her ne kadar genel robot kütüphanemizde DH (Denavit-Hartenberg) parametreleri belirlenmiş her hangi bir robot için ileri kinematik, jakobyeni ve çeşitli dönüşüm hesapları yapılabilse de gerçek zamanlı görevlerde, sistemimizin (Puma) daha verimli çalışmasını sağlama amacıyla, ilgili kodları C programlama dili ile yeniden hazırladık. Ortaya çıkan kütüphane hem gerçek zamanlı benzetim hem de uygulamalarda robot üzerinde başarı ile denenmiştir. Şekil 3.7 Puma kütüphanemizin ana hatlarını sunmaktadır.



Şekil 3.7:Genel Puma Kütüphanesi

3.4.1 Puma Kipleri (Puma Modes)

"Puma Modes" Bloğu Bölüm 2 de anlatılan kipleri içermektedir. Şekil 3.8'de içyapısı gösterilmiştir.



Şekil 3.8: "Puma Modes" Bloğu

3.4.2 Yazılan S-Fonksiyonu Blokları (S function)

"S-function" bloğu puma robotunun kiplerinde kullanılan "C-Mex" fonksiyonlarını içermektedir. Bu fonksiyonlar şu şekilde sıralanabilinir.

Encoder to Radian: Okunan encoder değerlerini radyana çeviririr.

Trajectory Generator: Robotun o an bulunduğu pozisyonla belirtilen pozisyon arsında bir yörünge oluşturur.

Blender: Robotun bulunduğu pozisyonla kullanıcı tarafından verilen pozisyonlar(en fazla 7 pozisyon) arasında bir yörünge oluşturur.

Forward Kinematic: Puma robotunun bulunduğu konumdaki ileri kinematik denklemini hesaplar.

Jacobian Transpose: Puma robotunun bulunduğu noktada jacobianini bulur ve sonucu hata sinyali ile çarparak ($X-X_d$) motorlara uygulanması gereken gücü hesaplar.

Cartesian Trajectory Generator: Robotun bulunduğu kartezyen pozisyonla gitmesinin istendiği kartezyen koordinatlar arasında bir yörünge belirler.

RPY Trajectory Generator: Robotun bulunduğu oryantasyon ile gitmesinin istendiğimiz oryantasyon arasına bir yörünge belirler.

Safety: Robot ekleminin ulaşabileceği bölgenin dışına çıkmasını engeller.

Pd Control: Oran + türev şeklinde basit bir kontrolcüdür.

Gravity Load: Robotun bulunduğu konumda eklemlere uygulanan yer çekimi kuvvetini hesaplar

Puma Dynamic: Manipülatöre uygulanacak tork, etki eden yer çekimi kuvveti ve başlangıç konumundan robotun varacağı konumu hesaplar.

Inverse Kinematic: Robotun ters kinematiğini hesaplar. Bizim kullandığımız Puma robotunda her kartezyen pozisyona karşılık eklem uzayında 6 ayrı pozisyon bulunmaktadır. 6 olası sonuç içinden robotun o an için bulunduğu konuma en yakın olanını hesaplar.

3.4.3 Giriş Çıkış Blokları (I/O BLOCKS)

Bu blok puma560 robotunun donanımına ulaşmamızı sağlayan fonksiyonları içermektedir.

Read Joint Position: Robotun bulunduğu pozisyonu bulur çıkış olarak hem encoder değerlerini hem de karşılık gelen radyan değerlerini verir.

Read Joint Position Sim Mode: "Read Joint Position" dan farklı olarak robot donanımına ulaşmadan robotun dinamiğinden gelen pozisyonu çıkışa verebilmemizi sağlamasıdır. Bu özellik simülasyon ortamında çalışmamızı sağlar.

Finangle: Robotun gitmesini istediğimiz pozisyonları girmemizi sağlar.

Blender Position: 7 pozisyona kadar robotun gitmesini istediğimiz noktaları girmemizi sağlar.

Joint Torque: robotun motorlarına girilen torka karşılık gelen gücü uygulamamızı sağlar. Motorlara uygulanması gereken güçten daha yüksek bir güç uygulandığı takdirde benzetim durdurulur.

Set Encoder ve *Encoder / Manual Set:* Encoder değerlerine istenilen değeri yazmamızı sağlarlar. Fakat Encoder set istenilen değerin yazılıp yazılmadığını içinde kontrol eder ve istenilen değer yazılana kadar işlemi sürdürür.

Enable Arm Power: Eklem motorlarını aktif hale getirir

Diasable Arm Power: Eklem motorlarının gücünü geçer ve pasif hale getirir

3.4.4 Sıklıkla Kullanılan Bloklar (USEFUL STUFF)

Burada sıklıkla kullanılan bazı kullanışlı fonksiyonlar verilmektedir.

Encoder Reseting: Encoder değerlerine istenilen değeri yazmamızı sağlarlar. İstenilen değer encoder yazıldıktan sonra "enc set done" çıkışı aktif (1) olur

Overfilter Unit: Giriş olarak verilen yörünge profilinin arka arkaya 2 defa türevi alarak, hız ve ivme değerlerini elde eder.

Gravity Load: Robotun bulunduğu konumda eklemlere uygulanan yer çekimi kuvvetini hesaplar.

Safety: Robot ekleminin ulaşabileceği bölgenin dışına çıkmasını engeller.

4 OLUŞTURULAN DEMO PROGRAMLAR

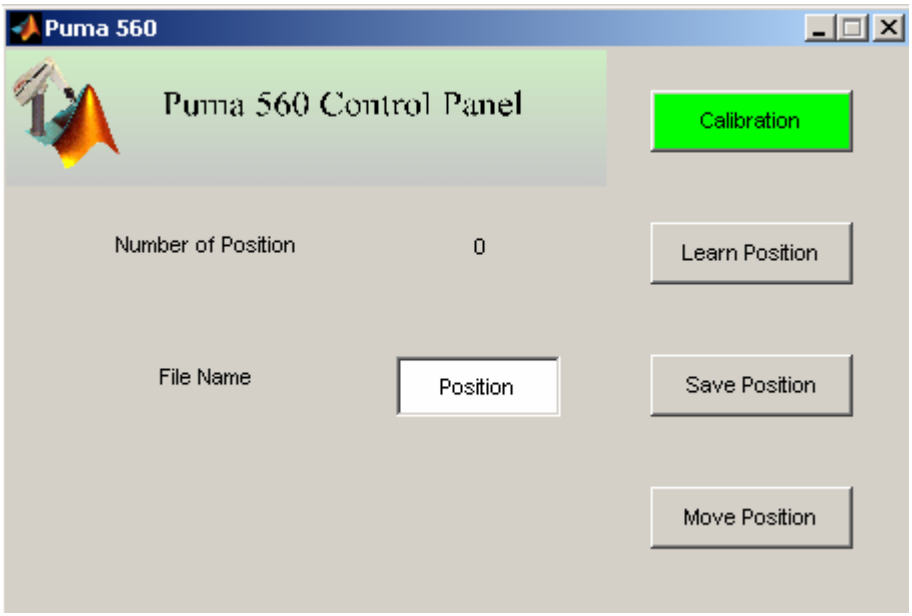
Bu bölümde çalışmamızın uygulanabilirliğini gösterebilmek amacıyla hazırladığımız kütüphaneler yardımı ile oluşturulan 2 demo programı tanıtılınacaktır. Bunlardan ilkinde kullanıcı öğrenme kipi yardımıyla robota varmasını istediği pozisyonları öğretebilmekte, bu pozisyonları bilgisayarda saklayabilmekte ve daha sonra robotun bu konumlara hareketini sağlayabilmektedir. İkinci demo ise robot park pozisyonuna hareket ettikten sonra kullanıcının robot ucunu bir USB joystick yardımıyla hareket ettirmesini sağlamaktadır.

4.1 Öğrenme ve Eklem Uzayı Denetleme Demosu

Demoyu çalıştırmak için MATLAB komut satırında (Command Window)

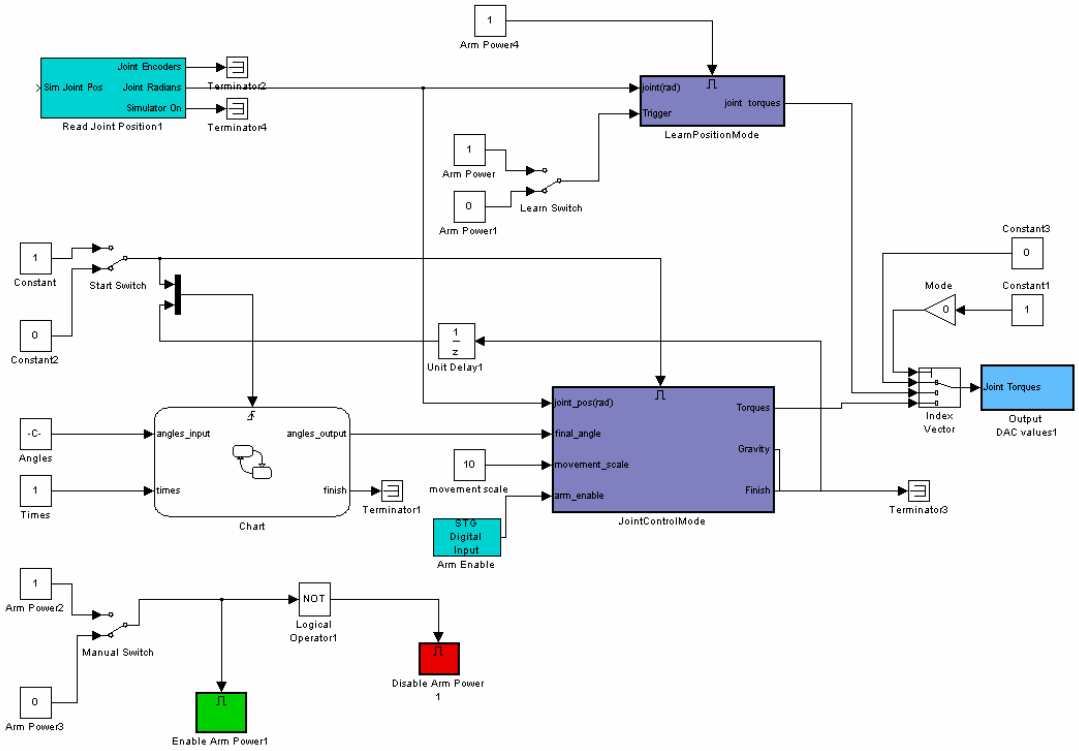
>>DemoLearnPos

Yazıp enter'a basın. Şekil 4.1'de görülen bu demo için oluşturulmuş grafik ara yüzü ve şekil 4.2'de kullanılan simulink programı gösterilmektedir.



Şekil 4.1: DemoLearnPos programının GUI si

Simulink programında görüldüğü gibi daha önce anlatılan öğrenme kipi ve eklem uzayı denetleme kipi bloklarından oluşmaktadır. Kullanımı kolaylaştırmak ve amacıyla basit bir GUI ekranı tasarlanmıştır. Bu ekranda "calibration" tuşu motor eklemlerine güç vermektedir. Ancak bundan sonra diğer düğmeler aktif hale gelmektedir. "Learn Position" tuşuna her basılışta robot bulunduğu konumu öğrenmekte daha sonra bu konumlar Save Position tuşu ile istenilen bir isimde kayıt edilebilinmektedir. "Move position" tuşu yardımıyla da robot bu noktalar arasında hareket ettirilebilinmektedir.



Şekil 4.2: DemoLearnPos programının Simulink Diyagramı

4.2 Joystick Mode demosu

Demoyu çalıştırmak için MATLAB command Window'da

>>DemoJoystick

Yazın ve entera basın.

Motor eklemlerine güç uygulanmasıyla robot park pozisyonuna yönelir. Park pozisyonuna geldikten sonra kullanıcı USB joyistik yardımı ile robotun uçunu hareket ettirebilir.

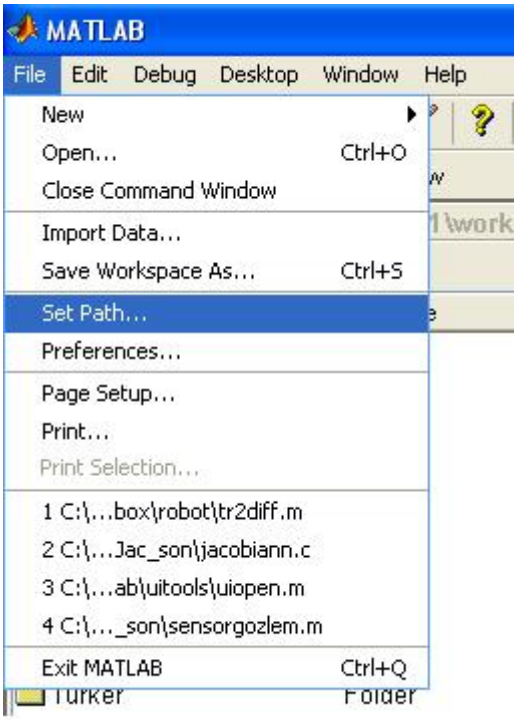
5 SONUÇLAR

Çalışmamızda açık sistem ve kod standartlarına uyularak robotik manipülatörler için geliştirilmekte olan, Matlab®/Simulink® tabanlı gerçek zamanlı benzetim ve denetim sistemi tanıtılmış ve Puma 560 robotu üzerinde başarı ile uygulanmıştır. Simulink Kütüphaneleri biçiminde ağırlıklı olarak C–kodu kullanılarak yazılan benzetim-denetim platformumuz, DH parametreleri tanımlanmış her robot manipülatörünün benzetim ve Puma tipi robotların gerçek zamanlı denetiminde kullanılabilir. Yapılan çalışmanın kodunun açık olması ve kullanımının kolaylığı göz önüne alınarak, az sayılabilecek bir ek çalışma ile başka manipülatör sistemlere de entegre edebilir ve PC ile bütünleştirilen arabirimleri robot sistemine eklenebilir. Çalışmamız ilerli bölümlerinde, robot ucuna takılmış kuvvet/güç sensoru ve çoklu kamera sistemi entegrasyonları yapılıp ileri kontrol algoritma uygulamaları geliştirilecektir.

Ek 1 Oluşturulan Robot Kontrol Kütüphanesinin Kurulumu

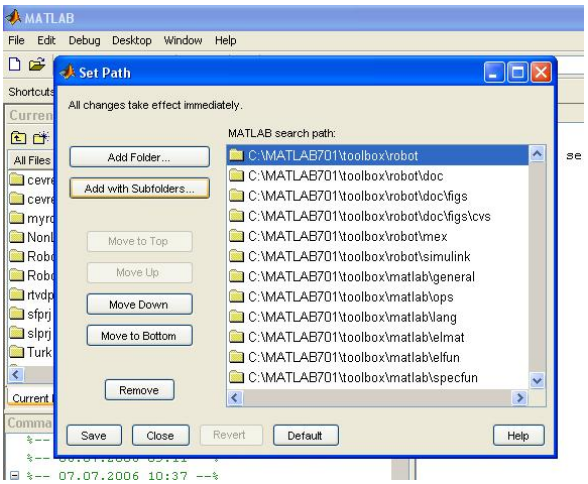
- Oluşturulmuş kütüphane fonksiyonları(GYTEROBOTLIB) Matlabın kurulu olduğu ana dizini altındaki toolbox klasörünün altına kopyalanır.

- Matlab programında FILE/SET PATH seçilir



Şekil EK1.1: FILE/SET PATH seçilir

- Açılan Pencerede "Add With Subfolders…" komutu seçilir



Şekil EK1.2: Add With Subfolders seçilir

- Açılan pencereden MATLAB'ın kurulu olduğu ana klasörün alında toolbox/GYTEROBOTLIB seçilir ve "Tamam" tıklanır.

Şekil EK1.3: toolbox/GYTEROBOTLIB seçilir

- Açık olan pencerede sırasıyla "Save" ve "Close" tuşlarına basılır.

- Gerçekleştirilecek programların ve var olan demoların gerçek zamanlı olarak çalıştırılabilinmesi için "*Real-Time Windows Target*

Kernel" inde kurulması gerekmektedir. Bu konu hakkında detaylı bilgi Matlabın yardımı dosyalarında "*Content/Real-Time Window Target/Installation and Configuration/Real-Time Windows Target Kernel*" başlığı altında bulunabilir.

- Ayrıca kurulacak bilgisayarda *GYTELIB* kütüphanesinin de kurulu olması gerekmektedir.

Ek 2 Servo to Go ISA Model II kartı

IBM PC'lerin ISA portuna takılan özellikle servo motorların kontrolünde kullanılan çok amaçlı bir giriş/çıkış kartıdır. Kartın fonksiyonları

- Encoder girişleri (8 tane)

Basit (single) ve farksal (differential) girişli olacak şekilde ayarlanabilir. Girişler A,B ve I olmak üzere encoder sinyallerini kabul etmekte, ve 24 bit sayıcı içermektedir.

- Analog çıkışlar (8 tane)

13 bit hassasiyetlidir. ±10v arası analog çıkış üretebilir

- Sayısal Giriş ve Çıkışlar(8x4 32 bit)

Ayrı ayrı giriş ve çıkış olarak tanımlanabilir ve opto-22 uyumludur.

- Analog giriş(8 tane)

13 bit hassasiyettedir. ±10v ve ±5v olarak ayarlanabilinir

- Internal timers

25 mikro saniye artımlarla 10 dakikaya kadar ayarlanabilinir

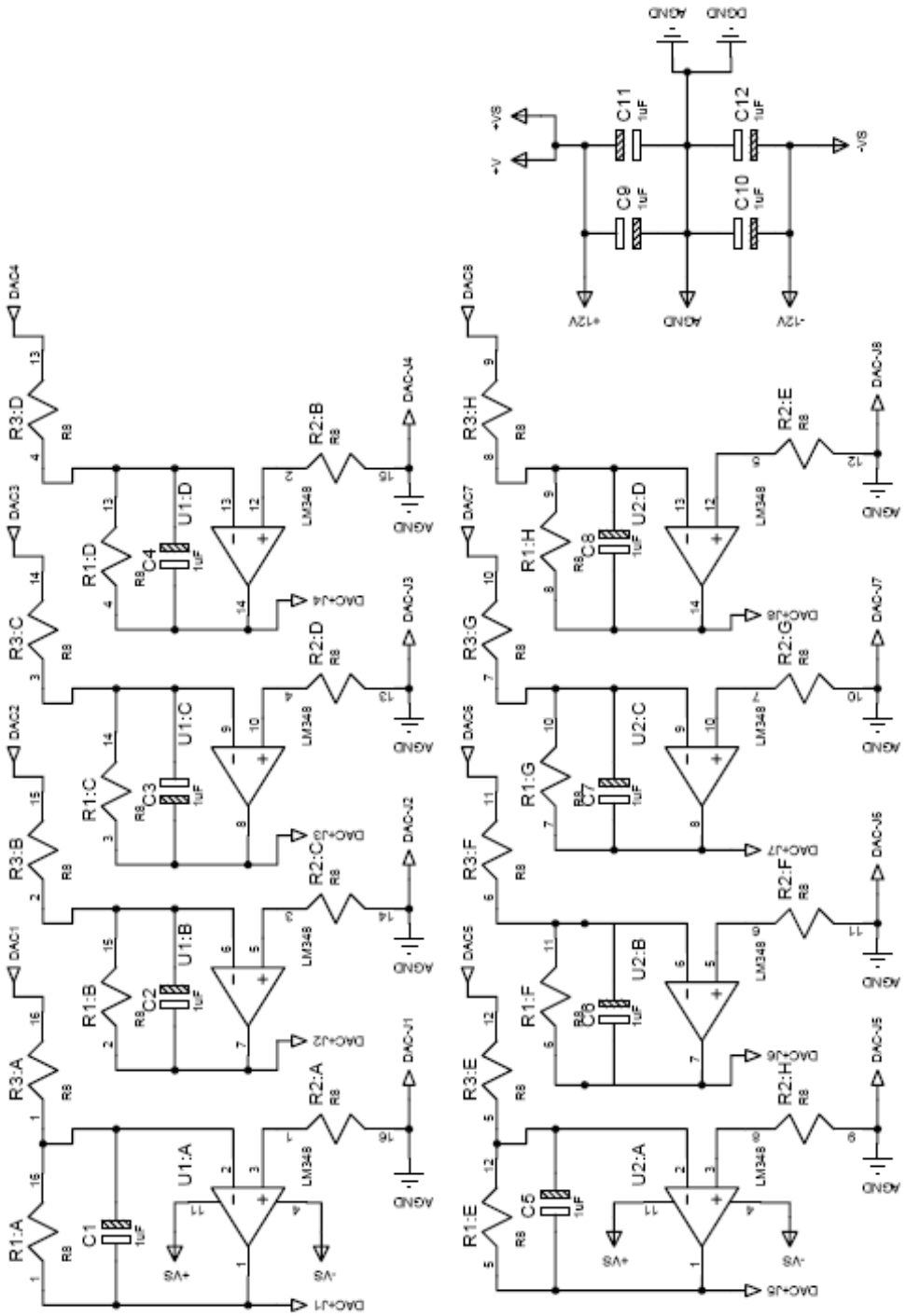
- Pil yedekleme girişi

Güç kesintilerine karsın encoder değerlerini saklamasını sağlar.

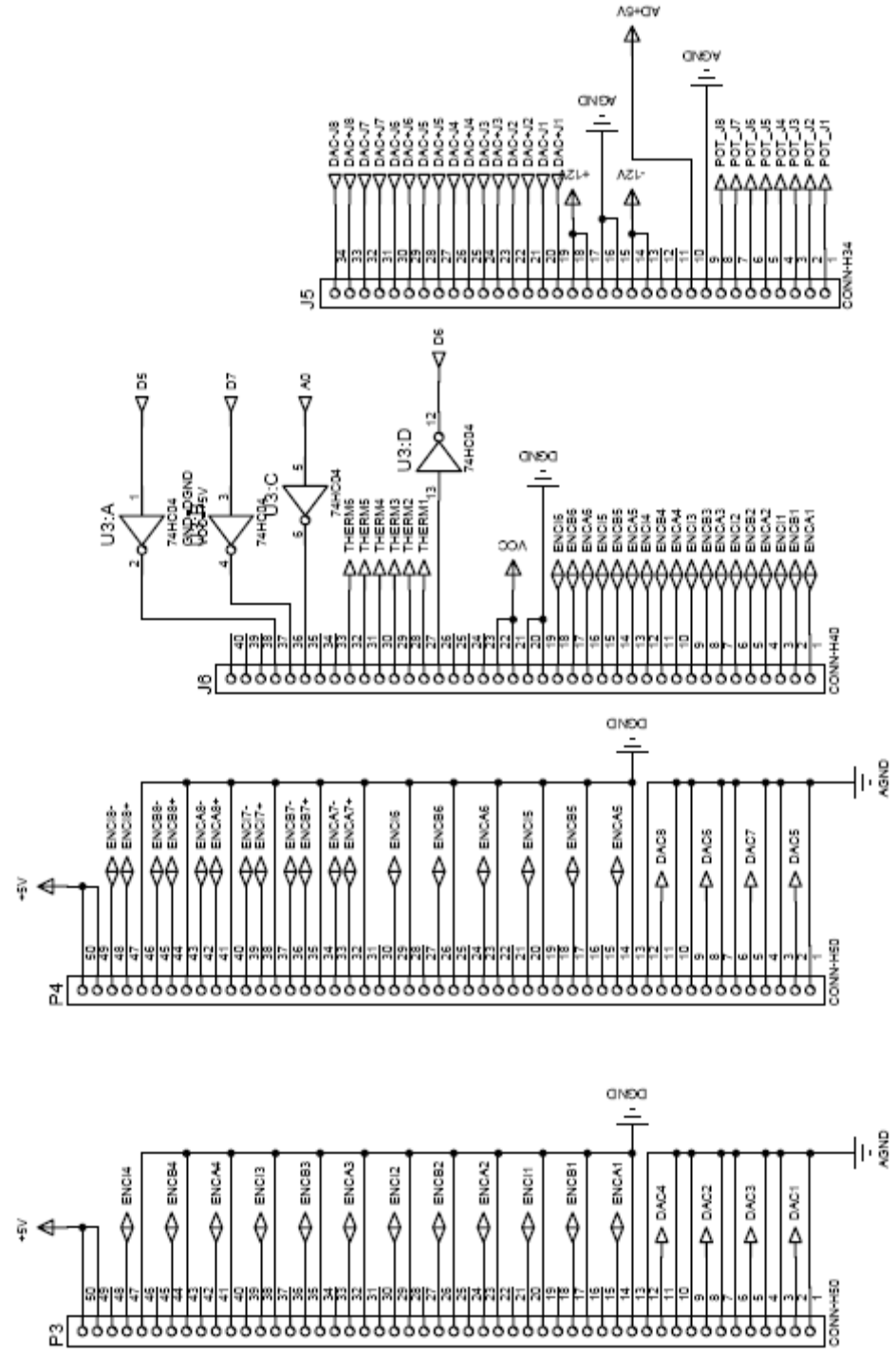
- Watchdog Timer

Ayrıntılı bilgi için referans [12] ye bakın.

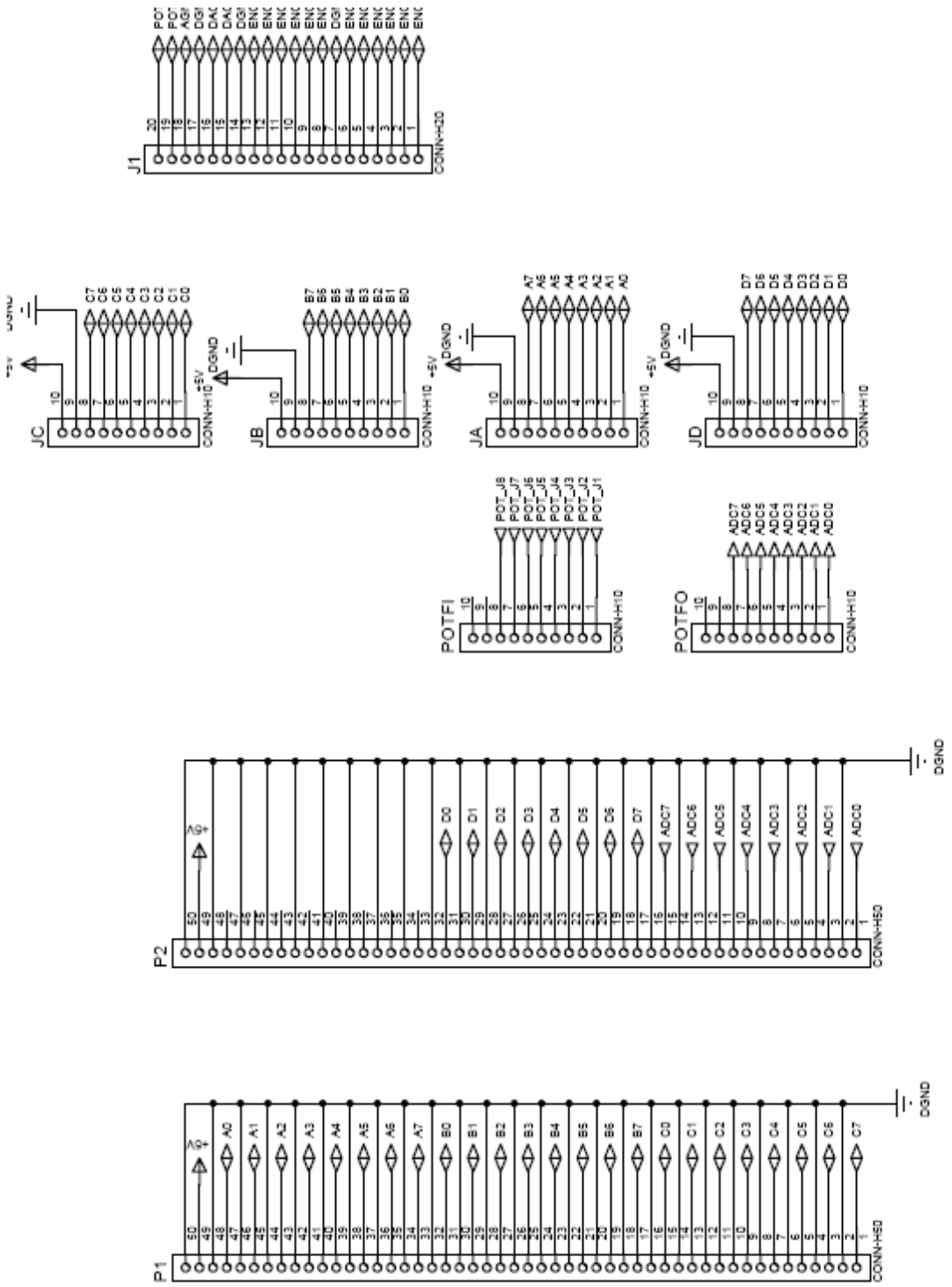
Ek 3 Tasarlana Geçiş Katının Şeması ve PCB Diyagramı



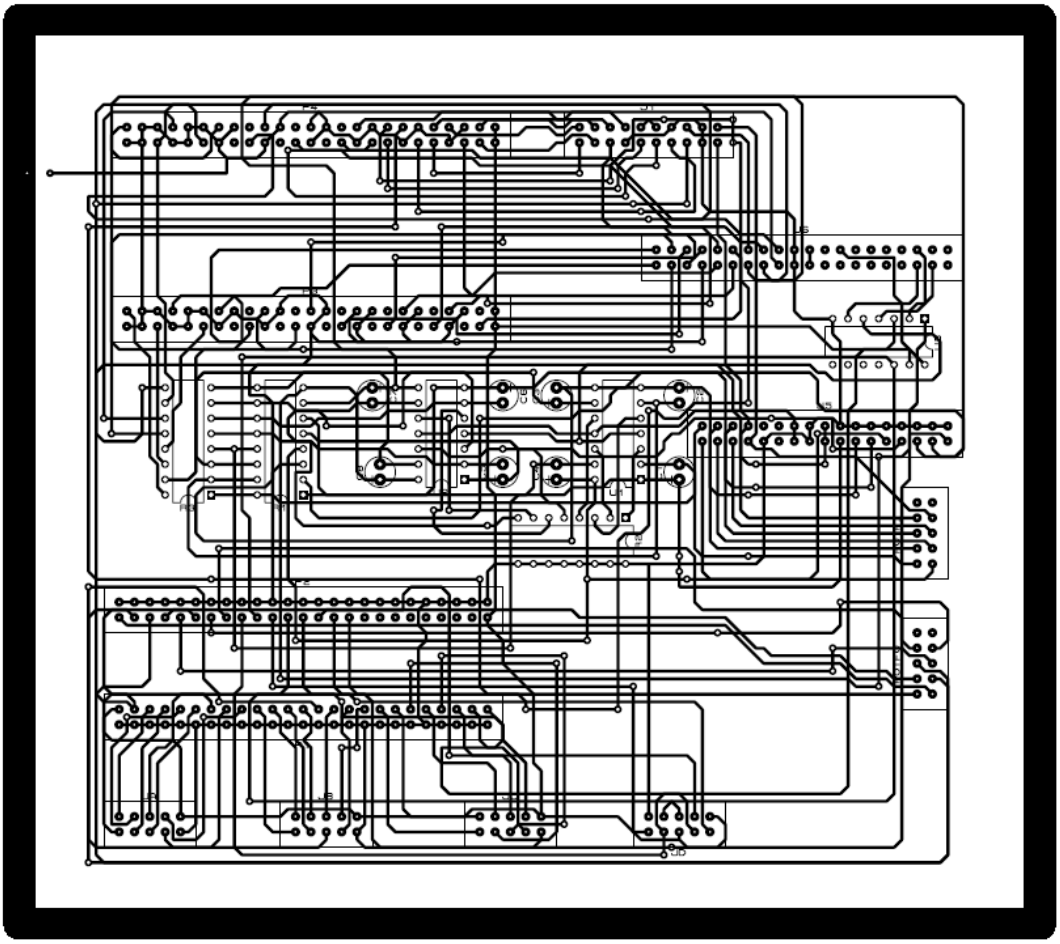
Şekil EK3.1 Geçiş Kartı şeması ADC katı



Şekil EK3.2 Geçiş Kartı şeması Konektör bağlantıları 1



Şekil EK3.3 Geçiş Kartı şeması Konektör bağlantıları 2



Şekil EK3.3 Geçiş Kartı Baskı Devre Resmi

Ek 4 Puma 560 Robotunun Özellikleri

Şekil Ek4.1 Puma560 robotunun resmi

Tablo Ek4.1 Puma Robotunun Genel Özellikleri

Genel	Eksen	6
	Sürücü	DC motor
	Kontrol	Sayısal
	Konum Kontrolü	Encoder
	Koordinatlar	Kartezyen
	Eklem Tipi	Döner

Çalışma Alanı	Bilek ulaşabilirliği 1.Eklem sınırı 2.Eklem sınırı 3.Eklem sınırı 4.Eklem sınırı 5.Eklem sınırı 6.Eklem sınırı	.864mm(1. ve 5. eklem arasında) 320^o 250^o 270^o 300^o 200^o 532^o
Yük Kapasitesi	Yük kapasitesi İzin verilen Bilek Yükü	4kg 2.5 kg eklem 5'den 127mm ve eklem 6'dan 37.6 mm uzakta
Performans	Hassasiyet Maksimum Hızı	±0.1mm 1.0 mm/s
Çevre	Çalışma ortamı sıcaklığı Çalışma ortam nemi Güç kaynağı	5^0-40^0 %50 40^0 %90 20^0 110V AC
Ağırlık	Kol	63kg

Tablo EK4.2 Dişli Oranları

Eklem	Dişli Oranı
G1	62.62
G2	107.36
G3	53.69
G4	76.01
G5	71.91
G6	76.63
G45	$-1/G_5$
G46	$-1/G_6$
G56	$-13/72$

Eklem ve motor açıları arsındaki ilişki denklem EK4.1 de verilmiştir

$$\theta_J = \begin{bmatrix} \frac{1}{G_1} & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{G_2} & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{G_3} & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{G_4} & 0 & 0 \\ 0 & 0 & 0 & \frac{G_{45}}{G_4} & \frac{1}{G_5} & 0 \\ 0 & 0 & 0 & \frac{G_{46}+G_{56}G_{45}}{G_4} & \frac{G_{56}}{G_5} & G_6 \end{bmatrix} \theta_m \qquad \text{(EK4.1)}$$

Tablo Ek4.3 Puma 560 robotunun Denavit-Hartenberg Parameteleri

Eklem	α	A_i	D_i
1	-90	0	0
2	0	431.8	149.09
3	90	-20.32	0
4	-90	0	433.07
5	90	0	0
6	0	0	56.25

KAYNAKLAR

[1] J. Lloyd, M. Parker and R. McClain, "Extending the RCCL Programming Environment to Multiple Robots and Processors", Proc. IEEE Int. Conf. Robotics & Automation (1988) s: 465 – 469.

[2] Unimation Inc., "500 Series Equipment and Programming Manual", Danbury, Connecticut, 1983.

[2] J. Lloyd, "Implementation of a Robot Control Development Environment", Masters Thesis, McGill University, Aralık 1985

[3] P. Corke, "The ARCL Kinematic Interface". Technical report, CSIRO Division of Manufacturing Technology, Mart 1990.

[4] N. Costescu, M. Loffler, E. Zergeroglu, and D. Dawson, "QRobot-A Multitasking PC Based Robot Control System", Microcomputer Applications Journal Special Issue on Robotics, Vol 18 No. 1, ss: 13-22

[5] J. N. Pires, "Interfacing Industrial R&A Equipment Using MATLAB", IEEE Robotics and Automation Magazine, Vol.7, No. 3, s: 32-41, 2000.

[6] W. E. Dixon, D. Moses, I. D. Walker, and D. M. Dawson, "A Simulink-Based Robotic Toolkit for Simulation and Control of the PUMA 560 Robot Manipulator", Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Maui, HI, Kasım 2001, s:. 2202-2207

[7] Z. Yao, N. P. Costescu, S. P. Nagarkatti, and D. M. Dawson, "Real- Time Linux Target: A MATLAB-Based Graphical Control Environment", Proc. of the IEEE Conference on Control Applications, Anchorage, ABD, Eylül 2000, s:173-178.

[8] The MathWorks, 3 Apple Hill Drive, Natick, MA, ABD 01760-2098, http://www.mathworks.com.

[9] B. Armstrong, O. Khatib, and J. Burdick, “The explicit dynamic model and inertial parameters of the Puma 560 arm,” in Proc. IEEE Int. Conf. Robotics and Automation,vol. 1, (Washington, USA), s: 510–18, 1986

[10] A. Berkay, M. Seker and E. Zergeroglu, "First Steps Towards a Windows-Based Control Design and Implementation Platform for Intelligent Systems” The International Conference on Intelligent Knowledge Systems (IKS-2004) Robotics and Automation conference under Automation/Real-time Control topics, Mayıs 2004, Truva, Türkiye.

[11]Richard Voyles' PUMA type Manipulators Page http://www.cs.cmu.edu/~deadslug/trc4um.pdf, Alınma tarihi: Mayıs 2006.

[12] Servo To Go, Inc., 8117 Groton Lane Indianapolis, IN 46260-2821, ABD., www.servotogo.com , Alınma Tarihi Mayıs 2006.

[13] P.I. Corke, “A Robotics Toolbox for MATLAB”, Robotics & Automation Magazine, IEEE Volume 3, Issue 1, Mart 1996 s:24 -32.

ÖZGEÇMİŞ

1982 yılında Kocaeli İzmit'te doğdu ilk orta ve liseyi İzmit'te bitirdi. 2000 yılında Sakarya Üniversitesi Mühendislik Fakültesi Elektrik ve Elektronik Müh. Bölümünü kazandı 2001 yılında Kocaeli Üniversitesi Elektronik ve Haberleşme Müh. Bölümüne yatay geçiş yaptı. 2004 yılında mezun oldu ve aynı yıl Gebze Yüksek Teknoloji Enstitüsünde Bilgisayar Bölümünde yüksek lisans öğrenimine başladı. 2004 yılından bu yana Gebze Yüksek Teknoloji Enstitüsü Bilgisayar Mühendisliği Bölümünde yüksek lisans çalışması yapmakta 2005 yılında Gebze Yüksek Teknoloji Enstitüsünde Bilgisayar Bölümünde araştırma görevlisi olarak çalışmaya başladı. Çalışma alanları: Nonlineer kontrol sistemleri; Gerçek zamanlı işletim sistemleri ve bu sistemler üzerinde yazılım geliştirme; Endüstriyel sistemlerin kontrolü ve performanslarının geliştirilmesidir.