İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

İKİ AYAKLI YÜRÜYEN PROTOTİP ROBOTUN GERÇEK ZAMANLI DENETİMİ

YÜKSEK LİSANS TEZİ

Burak TASASIZ

Anabilim Dalı : Makine Mühendisliği

Programı : Sistem Dinamiği ve Kontrol

HAZİRAN 2011

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

İKİ AYAKLI YÜRÜYEN PROTOTİP ROBOTUN GERÇEK ZAMANLI DENETİMİ

YÜKSEK LİSANS TEZİ
Burak TASASIZ
(503091636)

Tezin Enstitüye Verildiği Tarih : 06 Mayıs 2011
Tezin Savunulduğu Tarih : 06 Haziran 2011

Tez Danışmanı : Yrd. Doç. Dr. Zeki Yağız BAYRAKTAROĞLU (İTÜ)
Diğer Jüri Üyeleri : Doç. Dr. Şeniz ERTUĞRUL (İTÜ)
Yrd. Doç. Dr. Berat Turgut KARYOT (İTÜ)

HAZİRAN 2011

Aileme,

ÖNSÖZ

Bu tez çalışması İstanbul Teknik Üniversitesi, Makina Fakültesi, Prof. Dr. Nimet Özdaş Otomatik Kontrol Laboratuvarı'nda, 106M340 numaralı ve "İki Ayaklı Yürüme Hareketi Modellemesi Kontrolü ve Prototip İmalatı" başlıklı TÜBİTAK Projesi kapsamında gerçekleştirilmiştir. Tez çalışması, bu proje kapsamındaki robotun kontrol sistemi tasarımı, entegrasyonu ve programlanması konularını kapsamaktadır.

Projeye dâhil olmamı sağlayan ve çalışmam boyunca danışmanlığımı yürüten proje koordinatörü Sayın Yrd. Doç. Dr Zeki Yağız Bayraktaroğlu'na teşekkürlerimi sunarım.

Daha önce aynı projede karşılaştığı sorunları benimle paylaşan ve tezim süresince bana tecrübelerini aktaran Ar. Gör. Mesut Acar' a da ne kadar teşekkür etsem azdır.

Son olarak, beni her zaman, her konuda ve koşulda destekleyen aileme ve yoğun temposuna rağmen yanımda olan Duygu Doğan'a teşekkürü bir borç bilirim.

Mayıs 2011

Burak Tasasız

(Uçak Mühendisi)

İÇİNDEKİLER

Sayfa

ÖNSÖZ ..... **v**
KISALTMALAR ..... **ix**
ÇİZELGE LİSTESİ ..... **xi**
ŞEKİL LİSTESİ ..... **xiii**
ÖZET ..... **xv**
SUMMARY ..... **xvii**
1. GİRİŞ **1**
1.1 Literatür Özeti 2
1.2 ITU İki Ayaklı Robotun Mekanik Yapısı 7
1.3 Sunuş Planı 12
2. DONANIM **13**
2.1 Hareket Sisteminin Seçimi 13
2.2 Bilgisayar Seçimi 15
2.3 Mutlak Kodlayıcı Seçimi 17
2.4 Kuvvet/Moment Algılayıcısının Seçimi 18
2.5 Batarya ve Güç Kaynağı Seçimi 19
2.6 Donanımların Sisteme Eklemlenmesi 20
3. YAZILIM **23**
3.1 Sürücü Kartlarıyla İletişim İçin Gerekli Algoritmalar 23
3.1.1 Motorlara açısal konum gönderilmesi 25
3.1.2 Artırımlı Kodlayıcının değerinin okunması 27
3.1.3 Motorların çektiği akım değerinin okunması 27
3.1.4 Mutlak kodlayıcılarla eklem açılarının okunması 28
3.2 Eklem Yörünge Referanslarının Üretilmesi 30
3.2.1 Robotun belirlenen konfigürasyonu için eklem yörüngeleri oluşturma 30
3.2.2 Robotun ters kinematiği kullanılarak eklem referansları oluşturulması 36
3.3 Eklem Yörünge Referanslarının Uygulanması 36
3.3.1 Eklem referanslarının motor referanslarına dönüştürülmesi 37
3.3.2 Motor referanslarının uygulanması 42
4. DENEYSEL ÇALIŞMALAR **45**
4.1 Örnekleme Zamanı Seçimi 45
4.2 Eklem Kontrocülerinin Parametrelerinin Ayarlanması 47
4.3 Robotun Verilen Referans Eklem Yörüngelerini Takip etmesi 48
5. SONUÇ VE ÖNERİLER **57**
KAYNAKLAR **59**
EKLER **61**

KISALTMALAR

DC	**:** Direct Current
ZMP	**:** Zero Moment Point
PCI	**:** Peripheral Component Interconnect
AIST	**:** National Institute of Advanced Industrial Science and Technology
METI	**:** Ministry of Economy, Trade and Industry
CAN	**:** Controller Area Network
KAIST	**:** Korea Advanced Institute of Science and Technology
SBC	**:** Single Board Computer
PID	**:** Proportional Integral Derivative
LAN	**:** Local Area Network
USB	**:** Universal Serial Bus
NiMh	: Nickel Metal Hydride

ÇİZELGE LİSTESİ

Sayfa

Çizelge 1.1 : Robotun eklemlerinin hareket erimleri..... 9
Çizelge 1.2 : Motorlar, eklemler ve aktarma organları..... 10
Çizelge 2.1 : Motorların sağlaması gereken maksimum momentler..... 13
Çizelge 2.2 : Seçilen motorlar ve sürücü kartları..... 14
Çizelge 2.3 : UNO2052E bilgisayarının özellikleri..... 16
Çizelge 2.4 : Bulunan uygun mutlak kodlayıcılar..... 17
Çizelge 3.1 : Verinin 0.baytının alabileceği değerler..... 24
Çizelge 3.2 : İşlem modu nesnesinin yapısı..... 25
Çizelge 3.3 : Denetim nesnesinin yapısı..... 26
Çizelge 3.4 : Pozisyon modu – referans değeri nesnesinin yapısı..... 26
Çizelge 3.5 : Anlık konum bilgisi nesnesinin yapısı..... 27
Çizelge 3.6 : Anlık akım değeri nesnesinin yapısı..... 28
Çizelge 3.7 : Mutlak kodlayıcıdan veri okuma algoritması..... 29
Çizelge 3.8 : Belirlenen robot konfigürasyonu için eklem açıları..... 30
Çizelge 3.9 : Kalça eklemlerindeki aktarma organlarının çevirme oranları..... 38
Çizelge 3.10: Bilek aktarma organı modelinde kullanılan parametreler..... 41
Çizelge 4.1 : Deneylerdeki maksimum referans eklem hızları..... 55
Çizelge 4.2 : Deneylerdeki maksimum referans motor hızları..... 55
Çizelge 4.3 : Deneylerdeki maksimum motor momentleri..... 56
Çizelge 4.4 : Deneylerdeki maksimum eklem momentleri..... 56
Çizelge A.8 : Sürücü kartlarının içinde bulunan nesne sözlüğü..... 72

ŞEKİL LİSTESİ

Sayfa

Şekil 1.1 : WABIAN-2 [4]..... 2
Şekil 1.2 : ASIMO [6]..... 3
Şekil 1.3 : HRP-4C [7]..... 4
Şekil 1.4 : JOHNNIE [8]..... 5
Şekil 1.5 : LOLA [9]..... 6
Şekil 1.6 : Albert HUBO [13]..... 7
Şekil 1.7 : Robotun katı model görünümleri [15]..... 8
Şekil 1.8 : Robotun serbestlik dereceleri, eksen takımları ve eklem açıları..... 8
Şekil 1.9 : Robotun farklı açılardan görünümleri..... 11
Şekil 2.1 : Hareket Sistemi [18]..... 15
Şekil 2.2 : UNO2052E endüstriyel bilgisayar..... 16
Şekil 2.3 : Scancon SCH24AB mutlak kodlayıcı..... 18
Şekil 2.4 : 6 eksenli Kuvvet/Moment algılayıcısı..... 19
Şekil 2.5 : Benzetim sırasında oluşan toplam akımlar..... 20
Şekil 2.6 : Donanım şeması..... 21
Şekil 2.7 : Robotun üstündeki donanımlar..... 22
Şekil 3.1 : Sürücü kartları ile iletişim için gereken CANbus mesajı [18]..... 24
Şekil 3.2 : Mutlak kodlayıcının iletişim yapısı..... 28
Şekil 3.3 : Sürücü kartları ile mutlak kodlayıcıların bağlantısı..... 29
Şekil 3.4 : Robotun başlangıç konfigürasyonu..... 31
Şekil 3.5 : Oluşturulan konum, hız ve ivme profilleri [19]..... 32
Şekil 3.6 : Homotetik hız profilleri [19]..... 33
Şekil 3.7 : Eklemlere başlangıç pozisyonu için yörüngeler oluşturulması..... 35
Şekil 3.8 : Eklem yörüngeleri uygulama şeması..... 36
Şekil 3.9 : Kalça eklemlerinde bulunan aktarma organları..... 37
Şekil 3.10 : Diz eklemlerinde bulunan aktarma organları..... 38
Şekil 3.11 : Dizdeki aktarma organının modeli..... 39
Şekil 3.12 : Bilek eklemlerinde bulunan aktarma organları..... 40
Şekil 3.13 : Bilekteki aktarma organlarının modeli..... 40
Şekil 3.14 : Sürücü kartlarına konum referanslarının gönderilmesi..... 43
Şekil 3.15 : Programların genel akış şeması..... 44
Şekil 4.1 : Motor referanslarının zaman paylaşımlı olarak gönderilmesi..... 46
Şekil 4.2 : Eklemlerde bulunan kontrol çevrimi [18]..... 47
Şekil 4.3 : Gövdenin 13 cm'lik yatay hareketi için referans eklem yörüngeleri..... 49
Şekil 4.4 : Gövdenin 13 cm'lik yatay hareketi için referans motor yörüngeleri..... 50
Şekil 4.5 : Gövdenin 13 cm'lik yatay hareketi için referans eklem hızları..... 51
Şekil 4.6 : Gövdenin 13 cm'lik yatay hareketi için referans motor hızları..... 52
Şekil 4.7 : Gövdenin 13 cm'lik yatay hareketinde ölçülen motor momentleri..... 53
Şekil 4.8 : Gövdenin 13 cm'lik yatay hareketindeki eklem momentleri..... 54

Şekil A.1 : RE-30 doğru akım motorunun veri kâğıdı..... 63
Şekil A.2 : RE-40 doğru akım motorunun veri kâğıdı..... 64
Şekil A.3 : RE-50 doğru akım motorunun veri kâğıdı..... 65
Şekil A.4 : Kullanılan sürücü kartlarının veri kâğıdı..... 66
Şekil A.5 : Kullanılan endüstriyel bilgisayarın veri kâğıdı..... 67
Şekil A.6 : Kullanılan mutlak kodlayıcının veri kâğıdı..... 68
Şekil A.7 : K/M algılayıcısının veri kâğıdı..... 70
Şekil C.1 : Gövdenin 5 cm'lik düşey hareketi için referans eklem yörüngeleri..... 92
Şekil C.2 : Gövdenin 5 cm'lik düşey hareketi için referans motor yörüngeleri..... 93
Şekil C.3 : Gövdenin 5 cm'lik düşey hareketi için referans eklem hızları..... 94
Şekil C.4 : Gövdenin 5 cm'lik düşey hareketi için referans motor hızları..... 95
Şekil C.5 : Gövdenin 5 cm'lik düşey hareketinde ölçülen motor momentleri..... 96
Şekil C.6 : Gövdenin 5 cm'lik düşey hareketindeki eklem momentleri..... 97
Şekil C.7 : Gövdenin 10 cm'lik düşey hareketi için referans eklem yörüngeleri..... 98
Şekil C.8 : Gövdenin 10 cm'lik düşey hareketi için referans motor yörüngeleri..... 99
Şekil C.9 : Gövdenin 10 cm'lik düşey hareketi için referans eklem hızları.....100
Şekil C.10 : Gövdenin 10 cm'lik düşey hareketi için referans motor hızları.....101
Şekil C.11 : Gövdenin 10 cm'lik düşey hareketinde ölçülen motor momentleri.....102
Şekil C.12 : Gövdenin 10 cm'lik düşey hareketindeki eklem momentleri..... 103
Şekil C.13 : Gövdenin 7 cm'lik yatay hareketi için referans eklem yörüngeleri.....104
Şekil C.14 : Gövdenin 7 cm'lik yatay hareketi için referans motor yörüngeleri..... 105
Şekil C.15 : Gövdenin 7 cm'lik yatay hareketi için referans eklem hızları..... 106
Şekil C.16 : Gövdenin 7 cm'lik yatay hareketi için referans motor hızları..... 107
Şekil C.17 : Gövdenin 7 cm'lik yatay hareketinde ölçülen motor momentleri..... 108
Şekil C.18 : Gövdenin 7 cm'lik yatay hareketindeki eklem momentleri.....109
Şekil C.19 : Gövdenin 10 cm'lik yatay hareketi için referans eklem yörüngeleri.....110
Şekil C.20 : Gövdenin 10 cm'lik yatay hareketi için referans motor yörüngeleri..... 111
Şekil C.21 : Gövdenin 10 cm'lik yatay hareketi için referans eklem hızları..... 112
Şekil C.22 : Gövdenin 10 cm'lik yatay hareketi için referans motor hızları..... 113
Şekil C.23 : Gövdenin 10 cm'lik yatay hareketinde ölçülen motor momentleri..... 114
Şekil C.24 : Gövdenin 10 cm'lik yatay hareketindeki eklem momentleri..... 115

İKİ AYAKLI YÜRÜYEN PROTOTİP ROBOTUN GERÇEK ZAMANLI DENETİMİ

ÖZET

12 serbestlik dereceli ITU Biped robotun, katı modeli daha önceden oluşturulmuş ve benzetim ortamında yürüme hareketlerinde başarı sağlanmıştır. Bu tezde, robotun prototipinde kullanılmak üzere elektronik donanımlarının seçimi, birbiriyle iletişimi, programlanması ve robota eklem yörüngesi oluşturulması konuları ele alınmış ve robota konum kontrolü uygulanarak iki ayak üzerinde dengede durması sağlanmıştır.

Robotun üstünde algılayıcı olarak, motor millerinin açısal konumunu ölçmek için artırımlı kodlayıcı, eklem açılarını ölçmek için mutlak kodlayıcı ve yer tepki kuvvetlerini ölçmek için 6 eksen kuvvet/moment algılayıcısı kullanılmıştır. Robotun motorları, motor sürücü kartları ve artırımlı kodlayıcılar ile beraber robotun üzerinde birbirinden bağımsız 12 adet kapalı çevrim servo sistem oluşturulmuştur. Böylelikle, sürücü kartlarına motor millerinin istenen açısal konumu gönderilerek sürücü kartlarının artırımlı kodlayıcılardan aldığı geri besleme ve içindeki PID kontrolörle bu istenen açısal konumları gerçekleştirmesi sağlanmıştır. Sürücü kartlarına, bütün motor millerinin açısal konum değeri CANbus haberleşme yolu üzerinden gömülü bir bilgisayar tarafından gönderilmiştir. Bu bilgisayarda gerçek zamanlı Linux işletim sistemi kullanılmıştır. Ayrıca eklem açıları mutlak kodlayıcılar tarafından ölçülerek artırımlı kodlayıcıların başlangıç değer problemi çözülmüştür.

İstenen eklem referanslarını robota uygulamak için tez kapsamında C dilinde iki program yazılmıştır. Bu programlardan birincisi, uygulanacak eklem referanslarını motor referanslarına çevrilmesi için oluşturulmuştur. İkinciyse, birinci programın oluşturduğu motor referanslarının robota uygulatılması için yazılmıştır. Bu iki program sayesinde, istenen eklem referansları bir dosya içinde gömülü bilgisayara yüklenerek robotun hareketi sağlanmıştır.

Robotun eklemlerini rastgele konumlardan belirlenmiş başlangıç konfigürasyonuna ulaşmasını sağlayacak eklem referansları üreten bir program da yazılmıştır. C dili kullanılarak yazılan bu programda, robotun her bir eklemine yamuk hız profilleri ile homotetik eklem yörüngeleri oluşturulmuştur. Son olarak da robotun benzetim ortamında gerçekleştirdiği çeşitli hareketlerin belirli örnekleme zamanındaki eklem açıları alınmış ve bu verilerle gerçek prototiple aynı hareketler gerçekleştirilmiştir. Çalışmada ele alınan bütün konular üretilen prototipte denenmiş ve deney sonuçları tezde sunulmuştur.

REAL TIME CONTROL OF BIPED ROBOT

SUMMARY

Previously, CAD design of 12 DOF ITU Biped Robot was realized and its walking simulations have succeeded. In this study, electrical component selection of the robot, hardware communication between the components, programming and trajectory generation of the robot are treated and the static equilibrium of the robot has been maintained with an implemented position controller.

Incremental encoders, absolute encoders, and 6-axes force/torque sensors were used on the robot in order to measure the angular positions of motors, joint angular positions, and contact forces respectively. At each axis, motor, motor driver and incremental encoder form an independent servo control system. Thus, motor drivers drive the motors to follow the desired angular positions by using the incremental encoder feedback and PID control algorithm. An embedded computer generates the reference trajectories for the motors and provides them to the motor drivers via the CANBus. A Real-time Linux operating system is used on the computer. Moreover, initial positions of the incremental encoders were determined by measuring the angular positions of the joints.

Two programs are written in C programming language so that the robot follows the reference trajectory. The first program computes the motor reference trajectories by using the joint reference trajectories. The second program applies the motor reference trajectories to the robot. Desired joint trajectories are loaded to the embedded computer in a file and these two programs control the motion of the robot.

Aditionally, an algorithm was developed in order to bring the joint angles to the predetermined initial configuration. In this algorithm which is programmed in C, homothetic joint trajectories were generated for each joint by using the trapezoidal velocity profiles. Finally, joint trajectories for some specific motions from simulations were discretized and this data was used in realizing the same motions on the prototype. All of the coverage of this thesis is tested experimentally on the constructed prototype and experimental results are presented.

1. GİRİŞ

Günümüzde algılayıcı, eyleyici, imalat ve bilgisayar teknolojilerindeki gelişmeler ve işçi maliyetlerinin görece artması, insanların yaptığı bir takım rutin işleri yapabilecek otonom sistemlere olan ilgiyi artırmıştır. Endüstride mevcut olarak kullanılan robot kolları birçok rutin işi yüksek hız ve hassasiyet performanslarıyla yerine getirebilmektedir. Fakat bu robot kolları yere sabit oldukları için çalışma uzayları sınırlıdır. Diğer yandan mobil robotlar, verilen görevleri çalışma uzaylarını görevin bulunduğu bölgeye taşıyarak başarabilir. Mobil robotlar, hareket sistemlerinin yapısına göre tekerlekli, paletli ve ayaklı robotlar olarak sınıflandırılabilir. Tekerlekli robotlar daha çok engebesiz ve düzgün yüzeylerde kullanım için tasarlanmaktadır. Bu tür robotların, insanın yaşadığı ortamlarda sık rastlanan merdiven tırmanması ve inmesi veya belirli yükseklikteki engelleri aşması mümkün değildir. Paletli robotlar ise engebeli yüzeylerde çalışabilir; ancak düşük enerji verimliliği ve büyük boyutları yüzünden tercih edilmezler[1]. İnsanların yaşadığı ortamlarda çalışacak robotların ayaklı olmasının önemli avantajları vardır. Çünkü ayaklı robotlar kesintili ve engebeli çalışma uzaylarında çalışabilirler. Hareket kabiliyeti açısından daha fazla esneklik sunan iki ayaklı robotlar üzerine yapılan çalışmaların esin kaynağı insandır. Robotların, fiziksel yapılarının insana benzemesi insanlarla aynı çalışma ortamını paylaşabilmesi açısından önemlidir.

İki ayaklı insansı robotların elektronik tasarımı imalat sırasında büyük önem kazanmaktadır. Benzetim ortamında yürüme algoritmalarında başarı sağlansa bile bunun gerçek bir robot üzerinde sorunsuz denenebilmesi için elektronik tasarımın özenle yapılmış olması gerekmektedir. Elektronik tasarımda kullanılacak eyleyicilerin, algılayıcıların, bilgisayarın seçimi ve bu donanımlar arasındaki iletişimin sağlanması önemli konu başlıkları olarak sayılabilir. Algılayıcı, eyleyici ve bilgisayar teknolojisindeki gelişmeler, çok serbestlik derecesine sahip iki ayaklı insansı robotların elektronik tasarımı ve imalatını her geçen gün daha da kolaylaştırmaktadır.

1.1 Literatür Özeti

1968 yılında Vukobratovic tarafından iki ayaklı yürüme hareketinin teorik temellerinin sunulmuştur[2]. Japonya'daki Waseda Üniversitesi'nde ise 1973 yılında Wabot-1 isimli robotla "statik yürüme" hareketi gerçekleştirilmiştir[3]. Günümüze dönersek Waseda Üniversitesi'nde yapılmış WABIAN-2 isimli robot günümüz literatüründeki gelişmiş insansı robotlardan bir tanesidir. WABIAN-2 (Şekil 1.1) her bacağında 7'şer, belinde 2, gövdesinde 2, her kolunda 7'şer, boynunda 3, her elinde de 3'er adet serbestlik derecesine sahiptir. Robotun boyu 1.53 m ve bataryaları dahil ağırlığı 64.5 kg'dır. Robotun bütün eklemleri, içinde DC motor, sürücü kartı ve hız ölçer barındıran hareket sistemleriyle sürülmektedir ve tüm eklemlerde hız kontrolü yapılmaktadır. Her ekleme de ilk pozisyonlara karar verebilmek için optik mikro algılayıcılar yerleştirilmiştir. Ayrıca yer tepki kuvvetlerini ölçmek ve ZMP'nin yerini tespit edebilmek amacıyla da robotun ayak bileklerine 6 eksenli kuvvet/moment algılayıcıları yerleştirilmiştir. WABIAN-2 gövdesine monte edilmiş bir bilgisayar ile kontrol edilir. Bu bilgisayarda QNX gerçek zamanlı işletim sistemi kullanılmaktadır[4].



Şekil 1.1 : WABIAN-2 [4].

Honda firması tarafından geliştirilmiş ve mevcut literatürdeki en gelişmiş insansı robot ASIMO'dur. Honda firmasının 1986 yılından beri yürüttüğü çalışmaların son

meyvesi olan ASIMO, 26 serbestlik derecesine sahip olup, 1.20 m boyunda ve 52 kg ağırlığındadır(Şekil 1.2). ASIMO'nun hareket sistemi ile ilgili literatürdeki bilgi çok azdır. Hareket için servo sistem kullanmaktadır. Bunun dışında, bacaklarında 6 eksenli kuvvet algılayıcıları ve gövdesinde de jiroskopla ivme ölçer bulunmaktadır. ASIMO'nun içinde tek bir bilgisayar yerine farklı görevler için farklı bilgisayarlar bulunmaktadır. Bu bilgisayarlar arasındaki iletişim ise PCI bus haberleşme protokolü ile sağlanmaktadır. Bu bilgisayarlardan bazıları gerçek zamanlı işletim sistemine sahipken görüntü işleme ve ses işleme bilgisayarları gibi bilgisayarlar gerçek zamanlı işletim sistemine sahip değildir. ASIMO'da ayrıca dışardan bir bilgisayar ile bağlantı kurabilmek için üstünde kablosuz iletişim modülü bulunmaktadır[5].



Şekil 1.2 : ASIMO [6].

Japonya'daki bir diğer önemli çalışma ise AIST ve Kawada Industries firması desteğiyle yürütülmekte olan HRP projesidir. Proje ilk olarak Honda firmasının P3 isimli robotu üzerinde yapılan çalışmalarla başlamıştır. En son prototip olan HRP-4C (Şekil 1.3), 1.58 m uzunluğunda ve bataryaları dahil 43 kg'dır. HRP-4C'nin dikkat çekici özelliği görünümünün insana benzemesidir. Robotun dış görünümü

tasarlanırken genç Japon kadınların vücut ölçülerini göz önüne alınmıştır. Robotun henüz yayınlanmış bilimsel çalışması olmadığı için donanım yapısı hakkında bilgi yoktur[7].



Şekil 1.3 : HRP-4C [7].

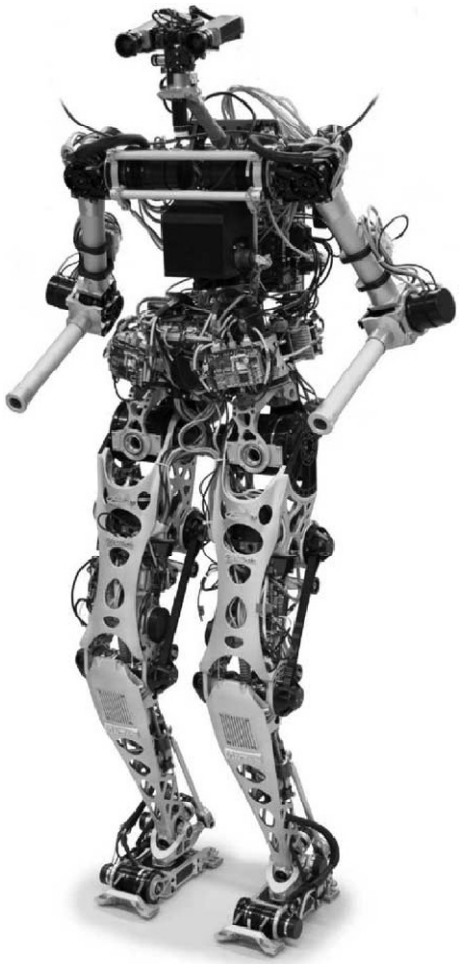
İnsan boyutlarındaki bir diğer insansı robot ise Münih Teknik Üniversitesi tarafından geliştirilmiş Johnnie'dir (Şekil 1.4) . Johnnie, bacaklarında 6'şar, üst gövdesinde 1 ve omuzlarında 2'şer serbestlik derecesine sahiptir. Robotun boyu, 1.80 m ağırlığı da 40 kg'dır. Robotun güç kaynağı dışardadır. Robotun bütün eklemleri sürücü kartları tarafından, DC motorlar ve bu motorların millerine bağlı 2000 çözünürlüklü artırımlı kodlayıcılar kullanılarak sürülmüştür. Ayrıca eklemlerin gerçek açıları ile artırımlı kodlayıcıların ölçtüğü açıları kalibre etmek için eklemin belirlenmiş açılarına 'ışık bariyerleri' yerleştirilmiştir. Robotun eklemine konulmuş optik alıcılar bu ışık bariyerlerinden geçtikleri zaman gerçek eklem açıları ile artırımlı kodlayıcılar arasında kalibrasyon yapılır. Johnnie'de sensörler, sürücü kartları ve bilgisayar

arasında CANbus iletişim protokolü kullanılmıştır. Robotun bilgisayarında gerçek zamanlı Linux işletim sistemi bulunmaktadır[8].



Şekil 1.4 : JOHNNIE [8].

Münih Teknik Üniversitesi tarafından Johnnie'den sonra tasarımına başlanan insansı robot Lola'dır (Şekil 1.5). Johnnie'den farklı olarak 25 adet serbestlik derecesine sahip Lola, 1.80 m boyunda ve 55 kg ağırlığındadır. Lola'nın hareket sisteminin yapısı Johnnie ile aynı olmasına rağmen eklemlerin gerçek açı değerlerini bulmak için Johnnie'deki gibi 'ışık bariyerleri' kullanmak yerine mutlak kodlayıcılar kullanılmıştır. Ayrıca bir diğer farklılık da donanımlar arası kullanılan iletişim protokolünde yapılmıştır. Lola'da iletişim protokülü olarak ethernet tabanlı gerçek zamanlı iletişim protokolü olan SERCOS III kullanılmıştır. Lola'nın üstündeki bilgisayarda, gerçek zamanlı QNX işletim sistemi kullanılmaktadır[9].



Şekil 1.5 : LOLA [9].

Kore İleri Teknoloji Enstitüsü KAIST ve Hanson Robotics firmasının ortak geliştirdiği Albert HUBO'da günümüzdeki gelişmiş insansı robotlardan bir tanesidir. Projeye 2000 yılında başlanmasına rağmen, Albert HUBO'dan önce birçok prototip başarıyla gerçekleştirilmiştir[10-12]. Albert HUBO, Albert Einstein'ın geliştirdiği özel görecelilik kuramının 100. yılını kutlamak adına imal edilmiştir. KHR-3 robotuna, Albert Einstein'ın kafasını model alınarak takılan 31 serbestlik dereceli yeni kafayla beraber Albert Hubo 66 serbestlik derecesine sahiptir. Albert Hubo 1.37 m boyunda ve 57 kg ağırlığındadır(Şekil 1.6). Robotun ayaklarında, 3 eksenli kuvvet/moment algılayıcısı ve eğimölçer, gövdesinde ise jiroskop ve eğimölçer bulunmaktadır. Robotta ana bilgisayar olarak Advantech firmasının geliştirdiği bir SBC (Tek kartta bilgisayar) kullanılmıştır. Bu ana bilgisayar diğer bilgisayarlara ve sürücü kartlarına CANbus iletişim protokolü üzerinden bağlanır. Ayrıca bu ana bilgisayarda, RTX HAL isimli bir yamanın üstüne kurulmasıyla beraber gerçek zamanlı hale getirilen Windows XP işletim sistemi kullanılmıştır[13].

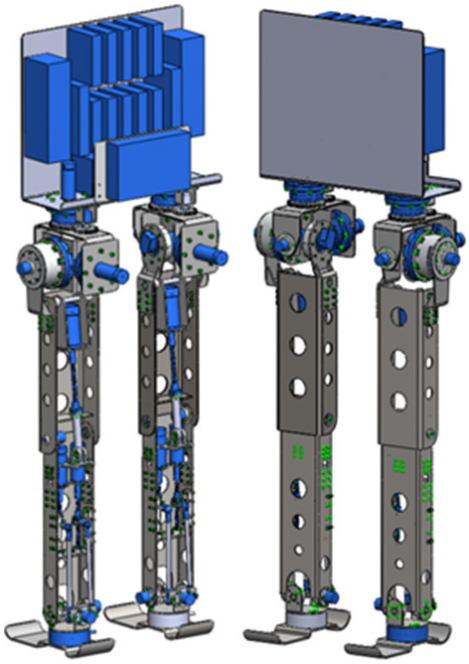


Şekil 1.6 : Albert HUBO [13].

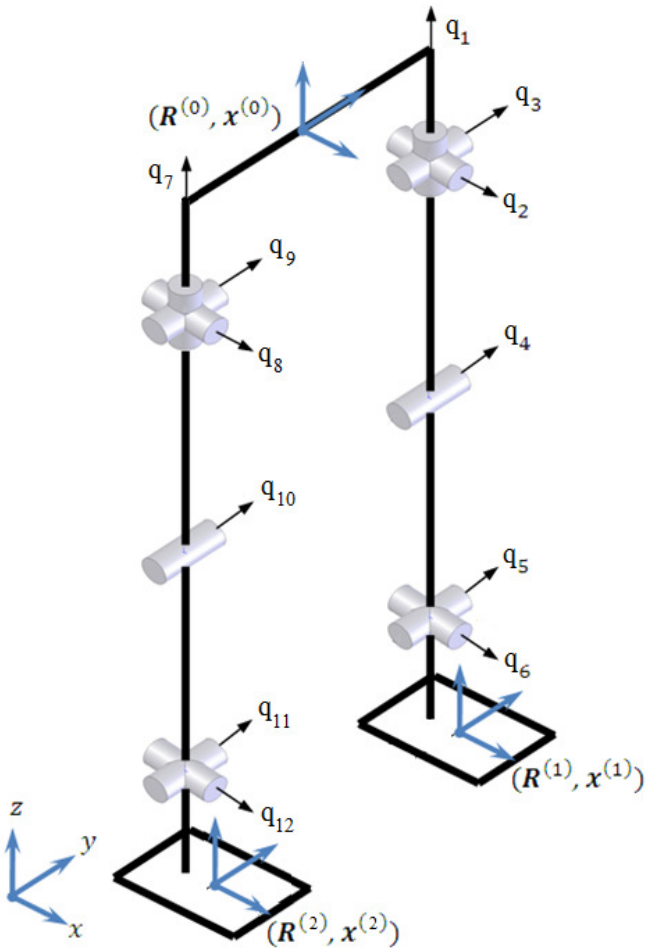
1.2 ITU İki Ayaklı Robotun Mekanik Yapısı

Bu bölümünde çalışmaya konu olan iki ayaklı robotun mekanik yapısı ile ilgili genel bilgiler verilecektir. Robotun mekanik tasarımı ile ilgili detaylı çalışmalar literatürde mevcuttur [14,15]. Mekanik tasarımın yanı sıra bu böllümde robotun üstünde bulunan serbestlik derecelerinden, eklemlerinin hareket erimlerinden ve üzerine monte edilmiş motorlarının eklemlerle bağlantısını sağlayan aktarma organlarının yapısından da bahsedilecektir.

İnşa edilen iki ayaklı yürüyen robotun daha önceki çalışmalarda[14] SOLIDWORKS© programı kullanılarak oluşturulmuş katı model görünümleri Şekil 1.7'de verilmiştir.



Şekil 1.7 : Robotun katı model görünümleri [15].



Şekil 1.8 : Robotun serbestlik dereceleri, eksen takımları ve eklem açıları.

Robot 12 serbestlik derecesine sahiptir. Bu serbestlik dereceleri kalça eklemlerinde 3 adet, diz eklemlerinde 1 adet, bilek eklemlerinde de 2 adet olacak biçimde dağılmaktadır (Şekil 1.8).

Şekil 1.8'de tüm eklem açıları sıfır konumlarındayken görülmektedir. Eklem açılarının pozitif yönleri şekilde verilen oklara sağ el kuralı uygulanarak bulunur.

Robotun üretimi gerçekleştikten sonra tüm eklemlerin hareket erimleri teker teker ölçülmüştür. Ölçülen erimler, Çizelge 1.1'de verilmiştir.

Çizelge 1.1 : Robotun eklemlerinin hareket erimleri

Eklem ismi	Simgesi	Hareket Erimi (˚)
Sol Kalça, Z	q_1	$-20 < q_1 < 20$
Sol Kalça, X	q_2	$-60 < q_2 < 60$
Sol Kalça, Y	q_3	$-90 < q_3 < 90$
Sol Diz, Y	q_4	$5 < q_4 < 82$
Sol Bilek, Y	q_5	$-53 < q_5 < -1$
Sol Bilek, X	q_6	$-30 < q_6 < 30$
Sağ Kalça, Z	q_7	$-20 < q_7 < 20$
Sağ Kalça, X	q_8	$-60 < q_8 < 60$
Sağ Kalça, Y	q_9	$-90 < q_9 < 90$
Sağ Diz, Y	q_{10}	$3 < q_{10} < 82$
Sağ Bilek, Y	q_{11}	$-52 < q_{11} < -3$
Sağ Bilek, X	q_{12}	$-30 < q_{12} < 30$

Hareket erimlerine bakıldığında bazı eklemlerin(q_4, q_5, q_{10}, q_{11},) sıfır değerine ulaşamadığı görülmüştür. Bu durum, üretimde ve montajda karşılaşılan sorunlardan kaynaklanmaktadır. Ancak, benzetim ortamında yapılan yürüme deneylerinde sıfır konumuna ulaşamayan bu eklemlerin hiçbir zaman bu konumuna ulaşması istenmediği görülmüştür. Dolayısıyla robotun yürüyebilmesi için herhangi bir erim sorunu bulunmadığı anlaşılmıştır. Üretilen robotun mekanik olarak yürümeyi başarabileceği öngörülmüştür.

Eklemlerin sürülebilmesi için robotun üstüne değişik güç kapasitelerinde 12 adet motor yerleştirilmiştir. Bu motorlarla eklemler arasında iki çeşit aktarma organı kullanılmıştır. Kalça eklemlerinde(q_1, q_2, q_3, q_7, q_8, q_9) aktarma organı olarak harmonik sürücüler bulunmaktadır. Diz(q_4, q_{10}) ve bilek(q_5, q_6, q_{11}, q_{12}) eklemlerindeki aktarma organları ise dikey olarak çalışan bilyalı vidalardan oluşmaktadır. Bilyalı vidalar, motorlardan aldıkları dairesel hareketi doğrusal harekete çevirirler(Çizelge 1.2).

Çizelge 1.2 : Motorlar, eklemler ve aktarma organları.

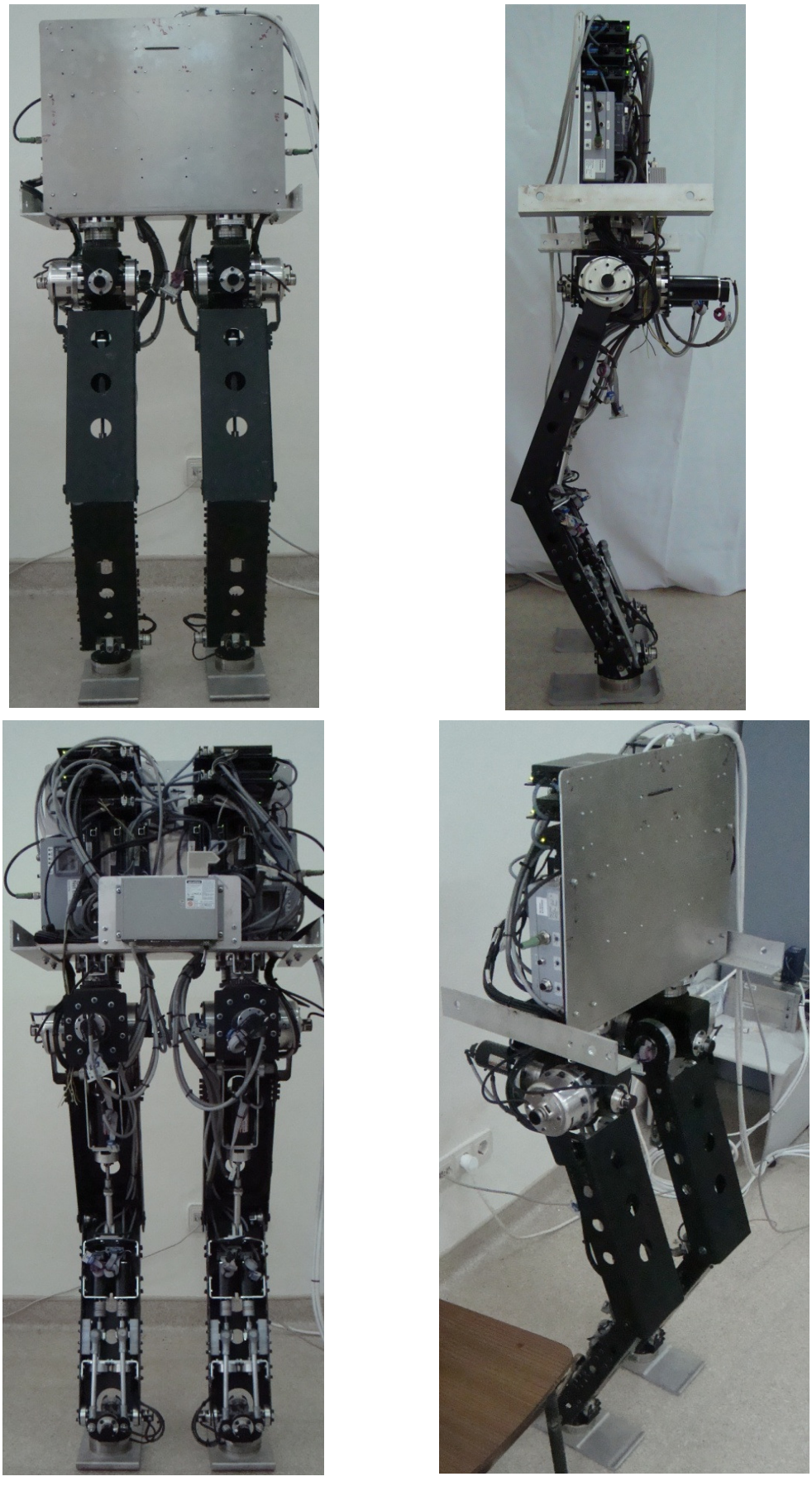
Motor ismi	Simgesi	Bağlı olduğu Eklem(ler)	Aktarma Organı
Sol Kalça, Z	Θ_1	q_1	Harmonik Sürücü (1:120)
Sol Kalça, X	Θ_2	q_2	Harmonik Sürücü (1:160)
Sol Kalça, Y	Θ_3	q_3	Harmonik Sürücü (1:160)
Sol Diz, Y	Θ_4	q_4	Rexroth Bilyalı Vida 1531-2
Sol Bilek, 1	Θ_5	q_5, q_6	Rexroth Bilyalı Vida 1531-1
Sol Bilek, 2	Θ_6	q_5, q_6	Rexroth Bilyalı Vida 1531-1
Sağ Kalça, Z	Θ_7	q_7	Harmonik Sürücü (1:120)
Sağ Kalça, X	Θ_8	q_8	Harmonik Sürücü (1:160)
Sağ Kalça, Y	Θ_9	q_9	Harmonik Sürücü (1:160)
Sağ Diz, Y	Θ_{10}	q_{10}	Rexroth Bilyalı Vida 1531-2
Sağ Bilek, 1	Θ_{11}	q_{11}, q_{12}	Rexroth Bilyalı Vida 1531-1
Sağ Bilek, 2	Θ_{12}	q_{11}, q_{12}	Rexroth Bilyalı Vida 1531-1

Bilek eklemlerinin sürülme şekli diğer eklemlerden farklıdır. Bilek motorlarının paralel çalışması veya ters yönde çalışması bilek eklemlerinin birbirinden bağımsız olarak sürülebilmesini sağlar. Aktarma organları ile ilgili daha ayrıntılı bilgiler 3.Bölümde verilecektir.

Diz ve bilek ekleminde bilyalı vida kullanılıp motorların robotun daha yukarısına alınmasının sebebi, robotun ağırlık merkezinin mümkün olduğu kadar yukarıya kaydırılmasının istenmesidir. Çünkü, iki ayaklı yürüyen robotlardaki ters sarkaç modeli yaklaşımına göre ağırlık merkezinin sürülen eklemlerden daha yüksekte olması gerekmektedir[16]. Bu nedenle motor sürücü kartları, kuvvet/moment algılayıcısının arayüzleri ve batarya kalça eklemlerinin yukarısına, robotun bel kısmına monte edilmiştir.

Mekanik tasarımı İstanbul Teknik Üniversitesi Makina Fakültesi'nde yapılan robotun 46 adet farklı mekanik bileşeni Altınay Robotik ve Otomasyon A.Ş. firması tarafından imal edilmiştir. Robotun mekanik ve elektrik montajı ise İstanbul Teknik Üniversitesi Makina Fakültesi ve Altınay Robotik ve Otomasyon A.Ş. firması işbirliği ile gerçekleştirilmiştir

Mekanik inşası tamamlanmış iki ayaklı yürüyen robotun boyu 142 cm uzunluğunda ve ağırlığı ise 55 kg'dır. Şekil 1.9'da üretilmiş bu robotun değişik açılardan görünümleri verilmiştir.



Şekil 1.9 : Robotun farklı açılardan görünümleri.

1.3 Sunuş Planı

İkinci bölümde çalışmaya konu olan iki ayaklı robotta kullanılan eyleyiciler, algılayıcılar ve gömülü bilgisayar gibi kontrol elemanlarının seçimi sunulmaktadır. Seçilen kontrol devresi bileşenlerinin gömülü bilgisayarla iletişimi için gerekli algoritmaların tasarımı ve yazılan bilgisayar kodları ile robotun hareket kontrolü için yazılan kodlar üçüncü bölümde yer almaktadır. Dördüncü bölümde robotla yapılan deneyler ve sonuçları sunulmaktadır. Son bölümde ise çalışmanın bir özeti ile ilerde yapılacak çalışmalarla ilgili yorumlar yer almaktadır.

2. DONANIM

Bu bölümde robotun eyleyicileri, algılayıcıları ve bilgisayarı gibi hazır olarak temin edilecek bileşenlerinin seçimi sunulmaktadır. Sonra da robotun seçilen bu donanımlarının birbirine nasıl bağlandığından bahsedilmektedir. Seçilen donanım, sistemin hareket kontrolü için tasarlanan kontrol devresinde kullanılmaktadır.

2.1 Hareket Sisteminin Seçimi

Elektrik motoruyla tahrik edilecek bir aktif serbestlik derecesinde en az bir elektrik motoru, bu motorun hareketini sağlayacak bir elektronik sürücü devresi ve motorun milinin açısal konumunu algılayacak bir konum algılayıcısı bulunması gerekir.

Hareket sistemin tasarımına, ilk olarak elektrik motorunun seçilmesiyle başlanılmıştır. Elektrik motorunun seçilmesi için robotun ilgili eklemine bağlanmış motorun sağlaması gereken momentlerin hesaplanması gerekmektedir. Bu çalışmada, öncelikle robotun katı modeli SolidWorks® yazılımında oluşturulmuştur. Katı model dosyası daha sonra Adams® yazılımına aktarılmıştır. Adams yazılımında robotun tasarım parametreleri kullanılarak dinamik hareket benzetimleri gerçekleştirilmiştir. Dinamik benzetimlerde, çeşitli yürüme senaryoları için oluşturulan referans yörüngeleri üzerindeki hareketler incelenmiştir [17]. Çizelge 2.1'de benzetim sonuçlarına göre, robotun 1 km/s hızla yürümesini sağlayacak motorların maksimum moment değerleri verilmiştir. Yürüme simetrik olduğu için iki bacağın aynı yerde bulunan motorlarının maksimum moment değerleri eşittir.

Çizelge 2.1: Motorların sağlaması gereken maksimum momentler.

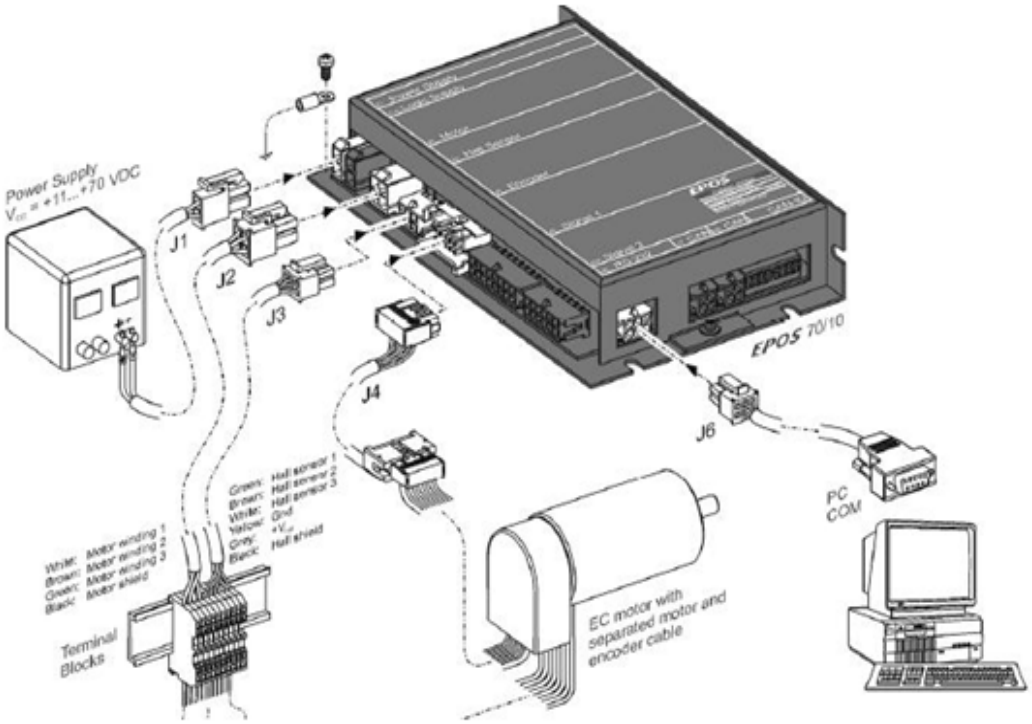
Motorlar	Maksimum Momentler (mNm)
Θ_1, Θ_7	100
Θ_2, Θ_8	300
Θ_3, Θ_9	250
Θ_4, Θ_{10}	400
Θ_5, Θ_{11}	150
Θ_6, Θ_{12}	150

Gereken maksimum momentler bulunduktan sonra bu momentleri sağlayabilecek motorlar herhangi bir motor üreticisi firmanın kataloğundan seçilebilir. Bunun için motor türüne de bu aşamada karar verilmesi gerekir. Bu çalışmada, robotun her eklemi için hassas konum kontrolü yapılması gerekmektedir. Bu yüzden, sistemde diğer motorlara göre kablolaması ve kontrolü daha kolay olan fırçalı doğru akım(DC) motoru seçilmiştir[12,13]. Belirlenen kuvvet değerlerine uygun olarak seçilecek DC motorlar için Maxon® firmasının motor kataloğundan faydalanılmıştır (EK A.1, EK A.2, EK A.3). Motorlar seçildikten sonra bu motorlara uygun motor sürücü devreleri de kolaylıkla belirlenebilir. Motor sürücülerinin seçimindeki önemli iki parametre motorların çalışma gerilimleri ve çekecekleri akımlardır. Benzetim ortamında her eklem için bulunan maksimum momentler, her eklem için seçilen DC motorların tork katsayılarına bölünerek, her motorun robotun hareketi sırasında çekeceği maksimum akım elde edilir. Robotun çalışma gerilimi akımların mertebesinin fazla yükselmemesi için 48V seçilmiştir. Akımların da hesaplanmasıyla birlikte Maxon firmasının sürücü kartları kataloğundan(EK A.4) uygun motor sürücü devreleri seçilmiştir (Çizelge 2.2).

Çizelge 2.2: Seçilen motorlar ve sürücü kartları.

Motorlar	Motorun Türü	Maksimum Akım (A)	Sürücü Kartı
Θ_1, Θ_7	RE-30	1.85	EPOS2 50/5
Θ_2, Θ_8	RE-50	3.18	EPOS2 70/10
Θ_3, Θ_9	RE-50	2.65	EPOS2 70/10
Θ_4, Θ_{10}	RE-50	4.25	EPOS2 70/10
Θ_5, Θ_{11}	RE-40	2.48	EPOS2 50/5
Θ_6, Θ_{12}	RE-40	2.48	EPOS2 50/5

Motor sürücü kartlarının seçilmesinden sonra robotun hareket sisteminin oluşturulması için geriye motor milinin açısal konumunu ölçecek algılayıcının seçimi kalır. Motor milinin açısal konumunu algılamak için yine Maxon Motors firmasının hareket sistemleri kataloğundan uygun arttırımlı kodlayıcılar seçilir. Bu çalışma kapsamında inşa edilmiş iki ayaklı yürüyen robotta kullanılmak üzere ilgili katalogdan motor milinin bir turunu 2000 artımla sayan manyetik artırımlı bir kodlayıcı seçilmiştir.



Şekil 2.1 : Hareket Sistemi [18].

İki ayaklı robotta kullanılmak üzere seçilen motor sürücü kartları, DC motorlar ve artırımlı kodlayıcılar ile birlikte servo motor sistemi oluştururlar (Şekil 2.1).

2.2 Bilgisayar Seçimi

İki ayaklı yürüyen robotun yürüyebilmesi için hareket sistemine eklemlerin açısal konum bilgilerini gönderecek bir bilgisayar gerekmektedir. Elektronik tasarım süresince bilgisayarın robotun üstüne monte edilmesine karar verilmiştir. Dolayısıyla seçilecek bilgisayar bir gömülü bilgisayar olacaktır. Bu gömülü bilgisayarda bulunması gerekecek önemli bir özellik CANBus iletişim portunun bulunmasıdır. Böylelikle, CANbus veri yolundan kolayca motor sürücü kartlarına yürüme için gereken açısal konum bilgileri gönderilebilir. Ayrıca gömülü bilgisayarın, öngörülen örnekleme periyodunda (10 ms) verilen komutları yerine getirebilmesi gerekmektedir. Günümüzdeki bilgisayarlar için 10 ms verilen komutların yetişmesi açısından uzun bir zaman dilimi olduğundan bu kriter göz ardı edilebilir. Son olarak da gömülü bilgisayara dışardan başka bir bilgisayarla müdahale etmek istenmektedir. Bunun için ise gömülü bilgisayarın bir LAN portunun olması yeterlidir.

Gömülü bilgisayarı seçerken bilgisayarın fiziksel özellikleri de önemlidir. Bilgisayar, robotun üstüne monte edileceği için bilgisayarın ufak, hafif, çarpmalara ve

sarsıntılara karşı dayanaklı olması gerekir. Bu sebeplerden dolayı bilgisayarın kompakt endüstriyel bilgisayarlar arasından seçilmesi önemlidir. Sarsıntılara karşı dayanıklı olması içinse gömülü bilgisayarda hareket eden bir disk veya fan bulunmaması gerekir.

Yukarıda sayılan özellikler düşünülerek, araştırmalar sonucu Advantech® firmasının geliştirmiş olduğu UNO 2052E (EK A.5) isimli gömülü endüstriyel bilgisayar seçilmiştir (Şekil 2.2).



Şekil 2.2 : UNO2052E endüstriyel bilgisayar.

Bu bilgisayarın standart bir kişisel bilgisayardan özellikleri açısından pek bir farkı yoktur (Çizelge 2.3).

Çizelge 2.3: UNO2052E bilgisayarının özellikleri.

Özellikler	Açıklama
İşlemci	AMD GX2-400 MHz
Bellek	256 MB DDR RAM
Sabit Bellek	4 GB kompakt flaş kart
Haberleşme Portları	1x USB, 1x RS232, 2xCANbus
Güç Kaynağı	9~36 VDC
Kasa Boyutu	188.8 x 106.5 x 35.5 mm

Seçilen bilgisayarda işletim sistemi olarak, çekirdeği "gerçek zamanlı" olarak derlenmiş Linux işletim sistemi kullanılmıştır. Bu sayede bilgisayarın kesme öncelikleri giriş çıkış portları öncelikli olarak yeniden düzenlenmiştir.

2.3 Mutlak Kodlayıcı Seçimi

Mutlak Kodlayıcılar, artırımlı kodlayıcılar gibi açısal konum ölçmek için kullanılan algılayıcılardır. Artırımlı kodlayıcılar açısal konum verilerini artımları sayarak elde ederken, mutlak kodlayıcılar açısal konumları mutlak olarak ölçerler. Artırımlı kodlayıcılardan mutlak bir veri alınamamasından dolayı, ölçüme başladığı ilk pozisyonu sıfır olarak kabul eden artırımlı kodlayıcı, onun üzerine yapılan artımları sayarak açısal konum verisine ulaşır. Bu da kontrol edilecek sistemde sıfırlama sorununa neden olur. Sıfırlama sorunu aşmak için eklemlerdeki servo hareket sisteminden bağımsız olarak her eklemde mutlak kodlayıcı kullanılması düşünülmüştür. Bu sayede robot çalışmaya başladığı zaman ilk açı değerleri mutlak olarak ölçülür ve hareket sistemine bu açı değerleri göz önünde bulundurularak açısal konum bilgileri gönderilir.

Sisteme mutlak kodlayıcı seçerken birkaç kısıtı göz önünde bulundurmak gerekir. Mutlak kodlayıcı seçilmeden önce hareket sistemi ve bilgisayar seçildiği için, mutlak kodlayıcının bu seçilen donanımlara uygun bir biçimde seçilmesi gerekir. Sisteme her ekleme birer tane olmak üzere toplam 12 adet eklenecek mutlak kodlayıcıların herhangi bir çevirici ya da arayüze gerek duymaması önem arz eder. Bu yüzden ilk olarak, mutlak kodlayıcı seçiminde iletişim için CANBus veri yolunu kullanan algılayıcılar ön planda tutulmuştur. Bu sayede, algılayıcıların mevcut olan CANbus veri yoluna kolayca eklemlenebileceği ön görülmüştür. Mutlak kodlayıcı seçimi için karşılaşılan diğer ve daha önemli bir kısıt ise boyut kısıtıdır. Robotun doğrudan eklemlerine yerleştirilmesi düşünülen mutlak kodlayıcıların boyutlarının eklemlere göre büyük olmaması gerekmektedir. Oluşturulan sisteme uygunluk ve ufak boyutlu olmak kısıtlarını birlikte düşünülerek yapılan araştırmalar sonucunda, sistemde kullanılabilecek birkaç mutlak kodlayıcı tespit edilebilmiştir (Çizelge 2.4).

Çizelge 2.4: Bulunan uygun mutlak kodlayıcılar.

Mutlak Kodlayıcı	Üretici Firma	Boyutlar (mm) (ÇapxUzunluk)	İletişim Protokolü
SCH24AB	Scancon	32x24.5	SSI
BMSH – MAGRES	Baum Electric	30x48	SSI
BMSK – MAGRES	Baum Electric	55x15.5	SSI
F3678	Kuebler	39x44.7	CANopen
MA-36	Encoder Technology	46x31	Grey Code
BMSV 42	Baumer	42x46	CANopen
MSC36	Lika	36x30	SSI

Veri transferi için CANbus iletişim protokolünü kullanan mutlak kodlayıcıların boyutlarını tahmin edilenden daha büyük çıkmasından dolayı araştırmalara diğer iletişim protokollerini kullanan algılayıcılar da dahil edilmiştir.

Tüm kısıtlar göz önünde bulundurulduğunda iletişim protokolünde taviz verilebilse dahi boyuttan taviz verilemeyeceği anlaşılmıştır. Böylelikle eklemlerin açısal konum bilgilerini mutlak olarak ölçebilmek için SCANCON firmasının ürettiği SCH24AB (EK A.6) isimli manyetik mutlak kodlayıcı sistemde kullanılmak üzere seçilmiştir (Şekil 2.3). q_1 ve q_7 hariç, seçilen mutlak kodlayıcı eklemlere doğrudan monte edilmiştir. q_1 ve q_7 'de ise monte edilen mutlak kodlayıcı ile eklem arasında mekanik kısıtlamalardan dolayı dişli sistemi bulunmaktadır.



Şekil 2.3 : Scancon SCH24AB mutlak kodlayıcı.

Mutlak kodlayıcıların sisteme eklenmesi ile eklem açıları doğrudan mutlak olarak ölçülebilir hale gelmiştir. Dolayısıyla motorun milinden ölçüm yapan artırımlı kodlayıcıların, aktarım organlarından kaynaklanabilecek olası hataları bu şekilde mutlak kodlayıcı tarafından telafi edilir. Bunun yanında, kullanılan mutlak kodlayıcı 12-bit çözünürlüğe sahip olduğu için eklem açıları çok hassas bir şekilde ölçülür.

Seçilen mutlak kodlayıcı CANbus iletişim protokolünü desteklememektedir. Çalışmanın yazılım bölümünde mutlak kodlayıcıdan veri okumak için gerekli algoritmanın nasıl oluşturulduğundan bahsedilecektir.

2.4 Kuvvet/Moment Algılayıcısının Seçimi

İki ayaklı yürüyen robot üzerinde yörünge planlamasına dayalı statik yürüme algoritmalarının yanı sıra dinamik yürüme kontrolünün de gerçekleştirilmesi düşünülmektedir. Dinamik yürüme kontrolünün gerçekleştirilebilmesi için robotun ayak tabanlarının etrafında oluşan sıfır moment noktasının (ZMP) konumunun bilinmesi gerekir. ZMP'nin anlık konumları, robotun ayak bileklerine monte edilen kuvvet/moment algılayıcıları ile hesaplanabilir. Bulunan ZMP noktasına göre de

tasarlanan bir denetleyici robotun eyleyicilerine gerekli açısal konumları yollayarak dinamik yürümeyi gerçekleştirir.

Bu sistemde ATI firması tarafından geliştirilmiş Mini85 isimli algılayıcı kullanılmıştır (Şekil 2.4) (EK A.7).



Şekil 2.4 : 6 eksenli kuvvet/moment algılayıcısı.

Bu kuvvet/moment algılayıcısı, ölçtüğü değerleri analog elektrik sinyallarine dönüştürmektedir. Algılayıcıdan gelen bu analog sinyallerin çeşitli filtrelerden geçirilmesi ve kullanılacak endüstriyel bilgisayarın kabul edeceği bir haberleşme protokolüne dönüştürülmesi için bir algılayıcı arayüzü gerekir. Sistemde, bahsedilen özellikli arayüz için yine aynı firmanın bu algılayıcılar için geliştirmiş olduğu F/T Net Box kullanılmıştır. Bu arayüz sayesinde kuvvet ve moment bilgileri CANbus aracılığıyla endüstriyel bilgisayarla kolayca okunur.

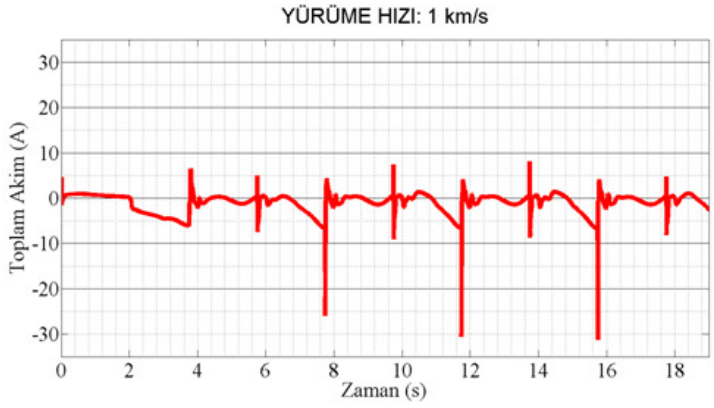
2.5 Batarya ve Güç Kaynağı Seçimi

Belli bir iş yapacak her mekanizma gibi iki ayaklı yürüyen robotun da harcayacağı enerjisini temin etmesi gereken bir enerji kaynağı gerekmektedir. Robotta kullanılan tüm bileşenler elektrik enerjisine ihtiyaç duymaktadır. Dolayısıyla, sisteme enerji beslemesi deneyler sırasında şebeke elektriğine bağlı bir doğru akım güç kaynağı tarafından yapılabilir. Ayrıca robotun otonom olarak hareket edebilmesi için istenildiğinde bir batarya da kullanabilmesi planlanmıştır. Sisteme enerji kaynağı olarak seçilmesi öngörülen doğru akım güç kaynağı ve bataryanın seçimi için sistemdeki bütün donanımın ve motorların çalışma gerilimleri ile çektiği akımların bilinmesi gerekmektedir. Sistemde, kuvvet/moment algılayıcılarının arayüzleri ile endüstriyel bilgisayarın çalışma gerilimi 24 V'tur. Sürücü kartlarının çalışma gerilimleri ise 48 V'tur. Bu uyumsuzluk sistemde bir de 48-24 Volt DC-DC çeviricisinin kullanılmasını zorunlu kılar. Seçilecek güç kaynağının çalışma

gerilimine karar verildikten sonra güç kaynağının verebileceği maksimum akım miktarının belirlenmesi gerekir.

Motorların çekeceği akımların mertebesi sistemde kullanılacak bileşenlerin çekeceği akımlardan çok fazla olduğu için bileşenlerin çekeceği akımlar güç kaynağı seçiminde ihmal edilebilir. Motorların çektiği akımlar, hareket sistemi seçimi yapılırken olduğu gibi yine benzetim ortamından alınmış verilerle hesaplanır. Tüm motorların aynı anda benzetim ortamına göre çekmesi gereken toplam akım bulunarak yürüme sırasında akımların nasıl değiştiği görülür. Hesaplanan maksimum toplam akım değeri de belirlenen bir güvenlik katsayısıyla çarpılarak seçilecek güç kaynağının vermesi istenen maksimum akım değeri belirlenir.

1 km/s'lik yürüme benzetim sonuçlarına göre yapılan hesaplamalarda robotun yürüme boyunca anlık olarak maksimum 30 A akım çektiği görülmüştür (Şekil 2.5).



Şekil 2.5 : Benzetim sırasında oluşan toplam akımlar.

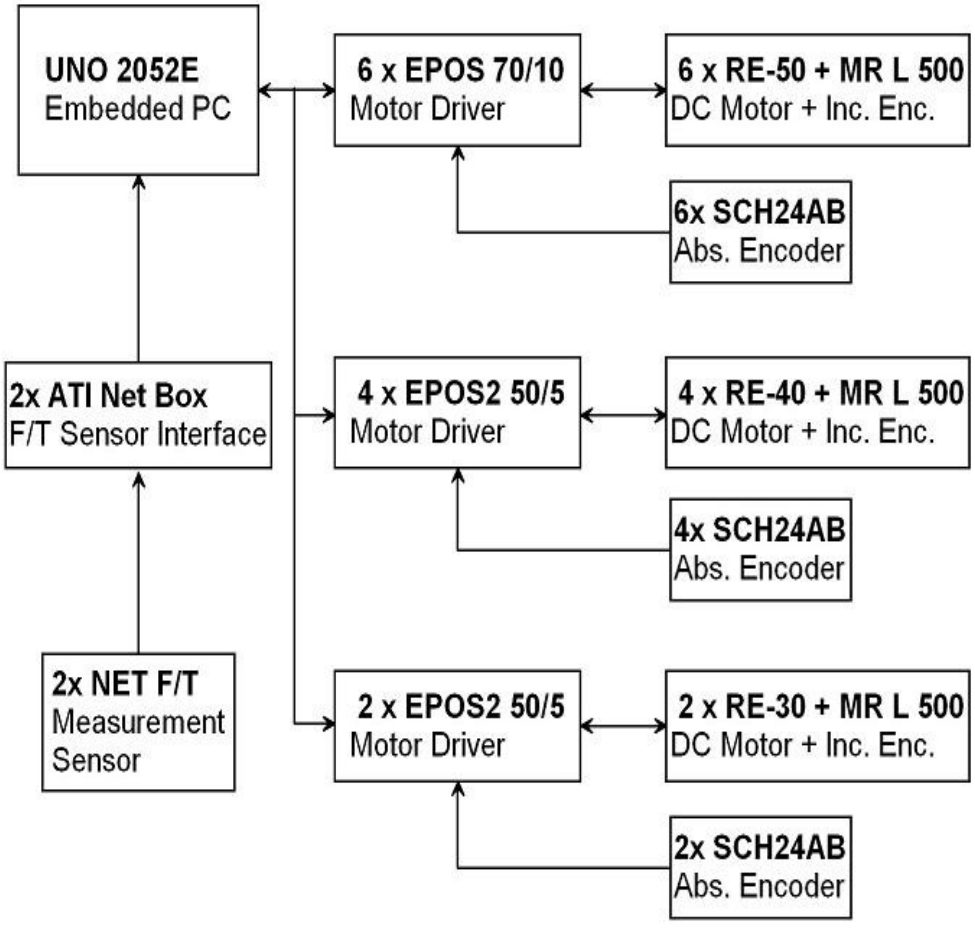
Buna göre de sistemde güç kaynağı olarak Lambda firmasının üretmiş olduğu SWS1000L-48 isimli güç kaynağından iki tanesi paralel olarak kullanılmıştır. Böylelikle güç kaynaklarının toplam verebileceği akım 44 A' e çıkarılmıştır. Batarya olarak da gerilimi 48 V olan, maksimum 32 A verebilen 4.2 Ah kapasiteli NiMh pil paketi seçilmiştir.

2.6 Donanımların Sisteme Eklemlenmesi

Elektronik tasarım boyunca üzerinde durulması gereken en önemli konu donanımlar arası iletişim olmuştur. Seçilecek donanımların çalışma gerilimleri ve iletişim protokollerinin birbirlerine uyumu büyük önem taşımaktadır. Robotun tasarımının ilk aşamalarında çalışma gerilimi 24 V olarak belirlenmiştir. Ancak motorların çektiği akımlar hesaplandığında, akım değerleri çok yüksek çıktığı için çalışma geriliminin

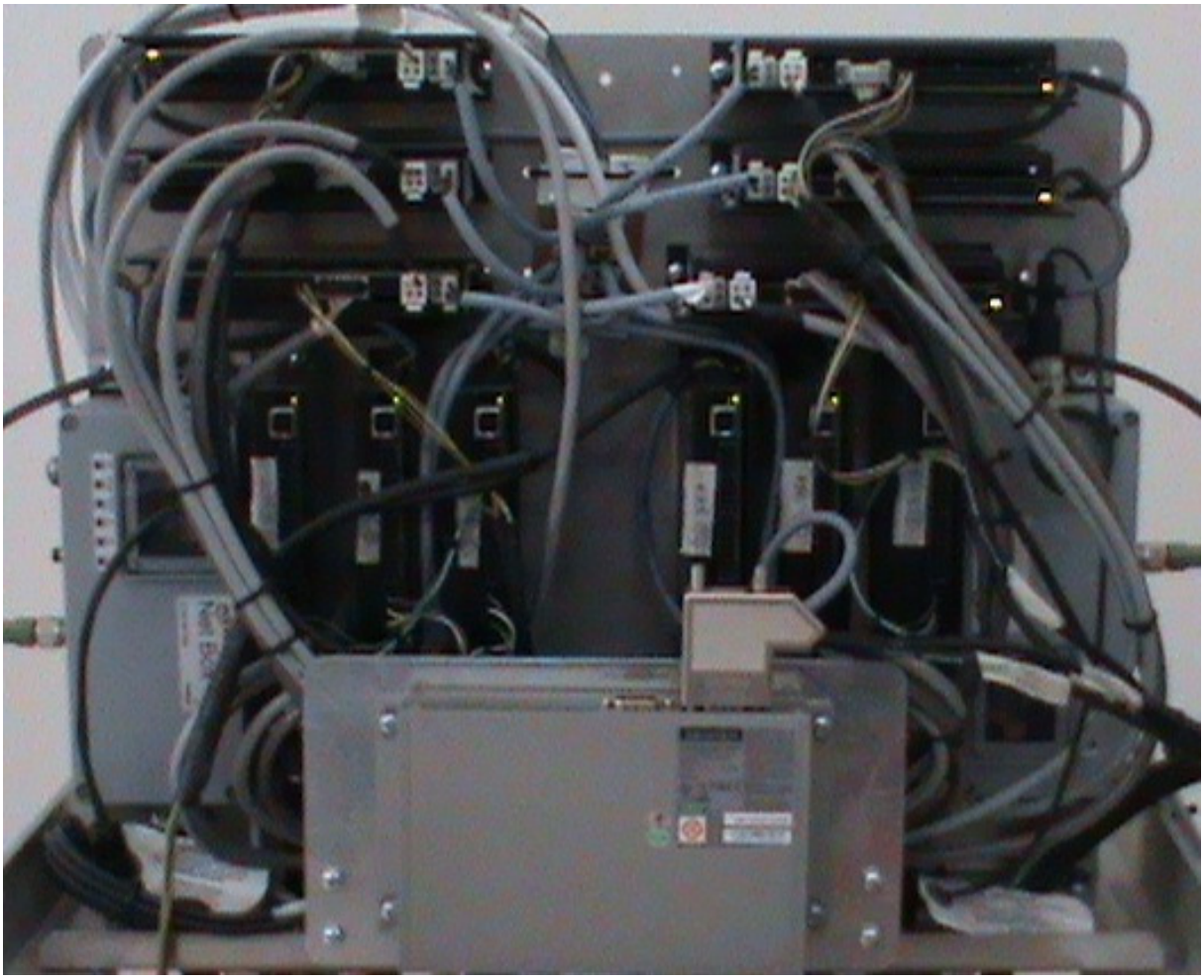
yükseltilmesi gerektiğine karar verilmiştir. Böylelikle çalışma gerilimi 48 V olarak değiştirilerek çekilen akımlar yarıya yarıya düşürülmüştür. Gerilim 48 V çıkarılmadan önce kuvvet/moment algılayıcısına ve endüstriyel bilgisayara karar verilmesinden dolayı bu iki bileşenin çalışma gerilimi 24 V olarak kalmıştır.

Robotta iletişim protokolü olarak donanımları birbirine seri bağlama imkanı veren CANbus kullanılmıştır (Şekil 2.6). Bu sayede hem tasarım hem de imalat süresince kablo karışıklığının da üstesinden gelinmiştir. Kablo sayısının ve uzunluğunun artması her robotta bir çok problem doğurur. Kablo sayısının artması demek konektör sayısının artması demektir. Bir elektronik sistemde konnektör sayısının fazla olması bir çok probleme yol açabilir. Ayrıca kablo uzunluklarının artması da robotun ağırlığını arttırır. Tüm bunlar göz önünde bulundurulduğunda donanımlar arası iletişim protokolü olarak CANbus gibi seri kablolama fırsatı tanıyan bir protokolün seçilmesi, bir robotun hem elektronik tasarımı hem de imalatı sürecinde büyük avantajlar sağlar.



Şekil 2.6 : Donanım şeması

Robotun elektronik tasarımı bitirildikten sonra imalat sırasında problem çıkartabileceği öngörülen kablo boylarının hareketli eklemler göz önüne alınarak ayarlanması, kabloların monte edilmesi ve konektörlerinin takılması robotun montajı esnasında yapılmıştır (Şekil 2.7).



Şekil 2.7 : Robotun üstündeki donanımlar.

3. YAZILIM

Robotun bileşenlerinin imalatı ve donanımların sisteme monte edilmesinden sonra robotun gerekli algoritmalarının oluşturulup, programlarının yazılmasına başlanmıştır. Bu bölümde, deneyler için gerekli programların yazılması ele alınacaktır.

3.1 Sürücü Kartlarıyla İletişim İçin Gerekli Algoritmalar

Robotlar, çevre ile etkileşim yapabilmek için bünyesinde birçok mekatronik bileşen barındırmaktadır. Bu bileşenlerin robotun karar verme organı olan bilgisayar ile iletişimleri önemlidir. Bu bölümde, iki ayaklı robotta kullanılan motor sürücü kartlarının robotun gömülü bilgisayarı ile nasıl iletişim kurduğu anlatılacaktır. Bu bölüm hazırlanırken Maxon Motors© firmasının uygulama notlarından faydalanılmıştır.[18]

İki ayaklı robotta kullanılan sürücü kartları ile bilgisayar arasındaki iletişim CANbus haberleşme protokolü ile sağlanmıştır. Sürücü kartlarının üretici firması, kullanıcının sürücü kartlarının içinde bulunan parametreleri okuyabilmesi ve gerekirse değiştirebilmesi için bir nesne sözlüğü geliştirmiştir. Bu nesne sözlüğünde her bir parametre bir nesne olarak tanımlanmıştır. Örneğin, sürücü kartının içinde bulunan PID kontrolcüsünün P, I ve D katsayıları ayrı ayrı hem değiştirilebilir hem de okunabilir nesnelerdir. Aynı şekilde artırımlı kodlayıcıdan okunan konum bilgisi, motorun dönmesi istenilen en yüksek hız, sürücü kartının dijital giriş/çıkışlarının durum bilgisi ve motorun o anda çektiği akım bilgisi de nesnelere örnektir. Ayrıca, üretici firma tarafından kullanıcının istediği nesneye ulaşabilmesi amacıyla her nesne için adres ve alt adres adı altında iki gösterici tanımlanmıştır. Nesne adresleri 2 baytlık ve alt adresleri 1 baytlık işaretsiz tamsayılardır. Bu adresler sayesinde kullanıcı hızlı bir şekilde istediği nesnenin değerini okuyabilmekte veya o nesnenin değerini değiştirebilmektedir. Sürücü kartlarında bulunan bazı nesneler adresleriyle, alt adresleriyle ve erişim izinleriyle beraber EK A.8'de verilmiştir.

Bilgisayarın istenen bir sürücü kartı ile haberleşme sağlayabilmesi için CANbus veri yoluna ilgili sürücü kartının ID'si(Kimlik Numarası) ile beraber 8 baytlık veri paketi aktarılmalıdır (Şekil 3.1). Bu veri paketinin içinde erişilmek istenen nesnenin adresi ve alt adresi, nesneye ne amaçla erişilmek istendiği ve nesnenin değeri değiştirilecekse nesnenin yeni değeri bulunmak zorundadır. Sürücü kartlarının ID'leri, sürücü kartlarının üzerlerinde bulunan minik anahtarlarla ayarlanır. İki sürücü kartına aynı ID verilmemesi gerekmektedir.

İstenen Sürücü kartının İstenen Nesnesini Okumak için Gereken Mesaj Yapısı:

MESAJ ID	VERİ [BAYT 0]	VERİ [BAYT 1]	VERİ [BAYT 2]	VERİ [BAYT 3]	VERİ [BAYT 4]	VERİ [BAYT 5]	VERİ [BAYT 6]	VERİ [BAYT 7]
0X600 + Sürücü ID	Bkz. Çizelge 3.1	Nesnenin Indeksi Bayt 0	Nesnenin Indeksi Bayt 1	Nesnenin Alt-Indeksi	REZERVE			

Sürücünün Cevabının Yapısı:

MESAJ ID	VERİ [BAYT 0]	VERİ [BAYT 1]	VERİ [BAYT 2]	VERİ [BAYT 3]	VERİ [BAYT 4]	VERİ [BAYT 5]	VERİ [BAYT 6]	VERİ [BAYT 7]
0X580 + Sürücü ID	Bkz. Çizelge 3.1	Nesnenin Indeksi Bayt 0	Nesnenin Indeksi Bayt 1	Nesnenin Alt-Indeksi	Nesne Bayt 0	Nesne Bayt 1	Nesne Bayt 2	Nesne Bayt 3

İstenen Sürücü kartının İstenen Nesnesine Değer Yazmak için Gönderilmesi Gereken Mesaj Yapısı:

MESAJ ID	VERİ [BAYT 0]	VERİ [BAYT 1]	VERİ [BAYT 2]	VERİ [BAYT 3]	VERİ [BAYT 4]	VERİ [BAYT 5]	VERİ [BAYT 6]	VERİ [BAYT 7]
0X600 + Sürücü ID	Bkz. Çizelge 3.1	Nesnenin Indeksi Bayt 0	Nesnenin Indeksi Bayt 1	Nesnenin Alt-Indeksi	Nesne Bayt 0	Nesne Bayt 1	Nesne Bayt 2	Nesne Bayt 3

Sürücünün Cevabının Yapısı:

MESAJ ID	VERİ [BAYT 0]	VERİ [BAYT 1]	VERİ [BAYT 2]	VERİ [BAYT 3]	VERİ [BAYT 4]	VERİ [BAYT 5]	VERİ [BAYT 6]	VERİ [BAYT 7]
0X580 + Sürücü ID	Bkz. Çizelge 3.1	Nesnenin Indeksi Bayt 0	Nesnenin Indeksi Bayt 1	Nesnenin Alt-Indeksi	REZERVE			

Şekil 3.1: Sürücü kartları ile iletişim için gereken CANbus mesajı [18].

CANbus mesajını oluşturan 8 baytlık verinin 0. baytı sürücü kartının ilgili nesnesine ne amaçla erişilmek istendiği ile ilgilidir. 0. baytın alabileceği değerler Çizelge 3.1'de verilmiştir.

Çizelge 3.1: Verinin 0. baytının alabileceği değerler

	Nesnenin Boyutu	Gönderilecek [Byte 0]	Alınacak [Byte 0]
Nesnenin Değerini Okurken	1 Bayt	0x40	0x4F
	2 Bayt	0x40	0x4B
	3 Bayt	0x40	0x43
Nesneye Değer Yazarken	1 Bayt	0x2F(veya 0x22)	0x60
	2 Bayt	0x2B(veya 0x22)	0x60
	3 Bayt	0x23(veya 0x22)	0x60
	Tanımlanmamış	0x22	0x60

Sürücü kartlarıyla haberleşebilmek için Şekil 3.1'deki ve Çizelge 3.1'deki bilgiler kullanılarak yazılacak programlarda kullanabilmek için "sendCan()" isimli bir C fonksiyonu yazılmıştır. Bu fonksiyon EK B.2'de verilen "Can4linux.h" kütüphanesini kullanarak bilgisayar üzerinden istenilen sürücü kartının istenilen nesnesine kolayca ulaşılabilmesini sağlar. Yazılan üç programda da kullanılmış olan bu fonksiyon EK B.1'deki kodun 223-259 , EK B.3'teki kodun 120-154 ve EK B.4'teki kodun 358-393 numaralı satırları arasında bulunmaktadır.

3.1.1 Motorlara açısal konum gönderilmesi

Kullanılan sürücü kartları kendisine bağlı bir adet DC motor ve artırımlı kodlayıcı ile birlikte bir kapalı çevrim kontrol sistemi oluştururlar. Bu sisteme, giriş olarak motor milinin açısal konumu verilir. Sürücü kartı yapısında barındırdığı PID kontrolcüsünü ve artırımlı kodlayıcıdan aldığı geri beslemeyi de kullanarak sisteme giriş olarak verilen referans açısal konum bilgisini DC motora besler.

Motorların hareket etmesi için sürücü kartlarına gönderilecek olan açısal konum bilgisi de sürücü kartlarının içinde bulunan diğer tüm parametreler gibi bir nesneye sahiptir. Bu nesneye ulaşılıp istenen yeni açısal konum bilgisi nesneye yazıldığında motorun mili, sürücü kartı tarafından bu yeni konumuna getirilir. Sürücü kartlarının, motorları harekete geçirmesinden önce bazı nesnelere değer ataması yapılmalıdır. İlk olarak işlem modunun belirlenmesi gerekir. Dolayısıyla, işlem modu nesnesine pozisyon kontrolü için gereken ilgili değer atanır(Çizelge 3.2).

Çizelge 3.2: İşlem modu nesnesinin yapısı.

Nesne İsmi	İŞLEM MODU
Nesne Tanımı	Kullanılan işlem modunun değiştirilmesini sağlar.
Nesne İndeksi	0x6060
Nesne Alt-İndeksi	0x00
Veri Tipi	İşaretli 8 bitlik tamsayı
Erişim İzni	Okuma/Yazma
İlk Değeri	1
Nesne Değeri	**Açıklaması**
7	İnterpolasyonlu Pozisyon Modu
6	Özgüdüm (Homing) Modu
3	Profilli Hız Modu
1	Profilli Pozisyon Modu
-1	Pozisyon Modu
-2	Hız Modu
-3	Akım Modu

İşlem modu nesnesine ilgili atama yapıldıktan sonra cihazın yazılımsal olarak kapatılıp yeniden açılması gerekir. Bunun için denetim nesnesinde bazı değişiklikler yapılmalıdır. Denetim nesnesine ilgili değerler sırasıyla atanarak cihaz kapatılır ve yeniden açılır(Çizelge 3.3).

Çizelge 3.3: Denetim nesnesinin yapısı.

Nesne İsmi	DENETİM NESNESİ
Nesne Tanımı	Cihazın denetim komutlarını içinde barındırır.
Nesne İndeksi	0x6040
Nesne Alt-İndeksi	0x00
Veri Tipi	İşaretsiz 16 bitlik tamsayı
Erişim İzni	Okuma/Yazma
İlk Değeri	-
Nesne Değeri	**Açıklaması**
0x0006	Cihazın kapatılması
0x000F	Cihazın açılması
0x0080	Cihazın hata durumunun sıfırlanması
0x0100	Cihazın tüm operayonlarının durdurulması

Denetim nesnesinde de ilgili değişiklikler yapıldıktan sonra cihazın poziyon modu için referans değerini barındıran nesnesine istenilen atama yapıldığında, sürücü kartı bu değeri motora uygulatır(Çizelge 3.4).

Çizelge 3.4: Poziyon modu – referans değeri nesnesinin yapısı.

Nesne İsmi	POZİSYON MODU – REFERANS DEĞERİ
Nesne Tanımı	Motor milinin ulaşması istenen açısal konum artım cinsinden bu nesneye yazılır.
Nesne İndeksi	0x2062
Nesne Alt-İndeksi	0x00
Veri Tipi	İşaretli 32 bitlik tamsayı
Erişim İzni	Okuma/Yazma
İlk Değeri	0
Nesne Değeri	**Açıklaması**
En düşük	-2 147 483 648
En yüksek	2 147 483 647

İşlem modu ve denetim nesnesinde yapılan değişiklikler motorlar hareket etmeden önce bir defa gerçekleştirilir. Motor, sırasıyla verilen referans değerlerini gerçekleştirirken işlem modu ve denetim nesnesinin değeri tekrar değiştirilmez. EK B.3'te bulunan kodun 96-106 numaralı satırları arasında bulunan "motorInit()" fonksiyonu bu iki nesnenin değerini değiştirirken 86-95 satırları arasında bulunan "motorDon()" fonksiyonu pozisyon modu referans değeri nesnesine atama yapar.

3.1.2 Artırımlı Kodlayıcının değerinin okunması

Motorlar verilen konum referanslarına göre sürülürken artırımlı kodlayıcının değerinin bilinmesine gerek yoktur. Ancak ileriki aşamalarda robotun eklem referanslarını motor referanslarına çevirirken motorların o anki açısal konumunun bilinmesi gerekmektedir. Bunun için sürücü kartının ilgili nesnesi okunarak bu değere ulaşılabilir (Çizelge 3.5).

Çizelge 3.5: Anlık konum bilgisi nesnesinin yapısı.

Nesne İsmi	ANLIK KONUM BİLGİSİ
Nesne Tanımı	Motor milinin artım cinsinden anlık konum bilgisini barındırır.
Nesne İndeksi	0x6064
Nesne Alt-İndeksi	0x00
Veri Tipi	İşaretli 32 bitlik tamsayı
Erişim İzni	Salt okunur
İlk Değeri	-

EK B-1'de bulunan eklem referanslarını motor referanslarına çeviren programın 205-222 numaralı satırları arasında bulunan "incEncoderRead()" fonksiyonu bu ilgili nesneden okuma yaparak anlık konum bilgisini alır.

3.1.3 Motorların çektiği akım değerinin okunması

Robotun üstünde bulunan tüm motorlarının, denemeler boyunca çektiği akımlar gözlenmek istenmektedir. Motorların çektiği akımları gözlemlemek, bağlı bulundukları eklemlere etkiyen kuvvet/momentlerin bilinmesini sağlar. Ayrıca denemeler sırasında motorların çektiği maksimum akımların belirlenmesi de seçilen motor sürücülerinin yeterli olup olmadığını gösterir. Motorların o anda çektiği akım değeri sürücü kartının ilgili nesnesinden okunur (Çizelge 3.6).

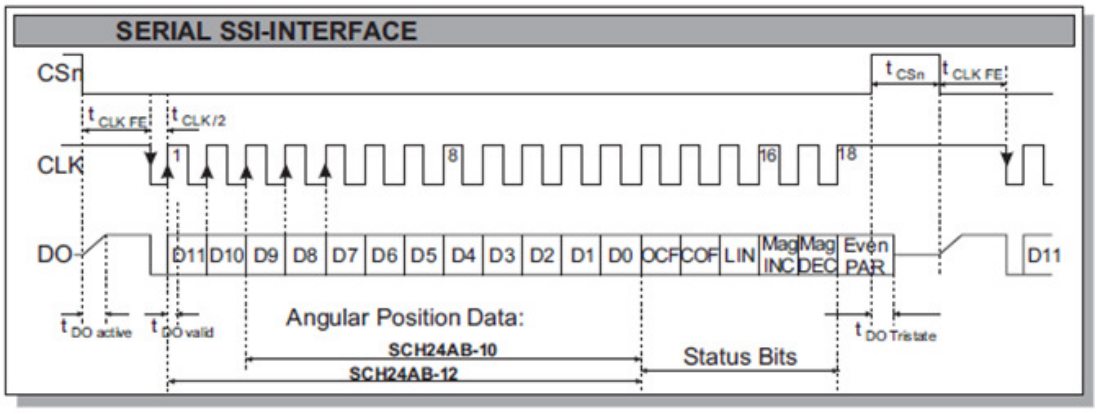
Çizelge 3.6: Anlık akım değeri nesnesinin yapısı.

Nesne İsmi	ANLIK AKIM DEĞERİ
Nesne Tanımı	Motorun mili Amper cinsinden anlık çektiği akım değerini barındırır.
Nesne İndeksi	0x6078
Nesne Alt-İndeksi	0x00
Veri Tipi	İşaretli 16 bitlik tamsayı
Erişim İzni	Salt okunur
İlk Değeri	-

EK B-3'te bulunan motor referanslarını uygulayan programın 224-237 numaralı satırları arasında bulunan "readCurrent()" fonksiyonu bu ilgili nesneden okuma yaparak anlık akım değerlerini okur.

3.1.4 Mutlak kodlayıcılarla eklem açılarının okunması

Seçilen mutlak kodlayıcılar ile iletişim üç sinyal hattı üzerinden sağlanmaktadır. Bu hatlardan iki tanesi giriş, kalan bir tanesini de çıkış sinyalidir. Kodlayıcıya, giriş hatlarından uygun sinyaller gönderildiği takdirde, kodlayıcı çıkış hattından seri olarak istenen açı ve iletişim durum bilgisini göndermektedir (Şekil 3.2). Seçilen mutlak kodlayıcılarla iletişime geçecek olan donanımda iki adet dijital çıkış ve bir adet de dijital giriş bulunması gerekmektedir.



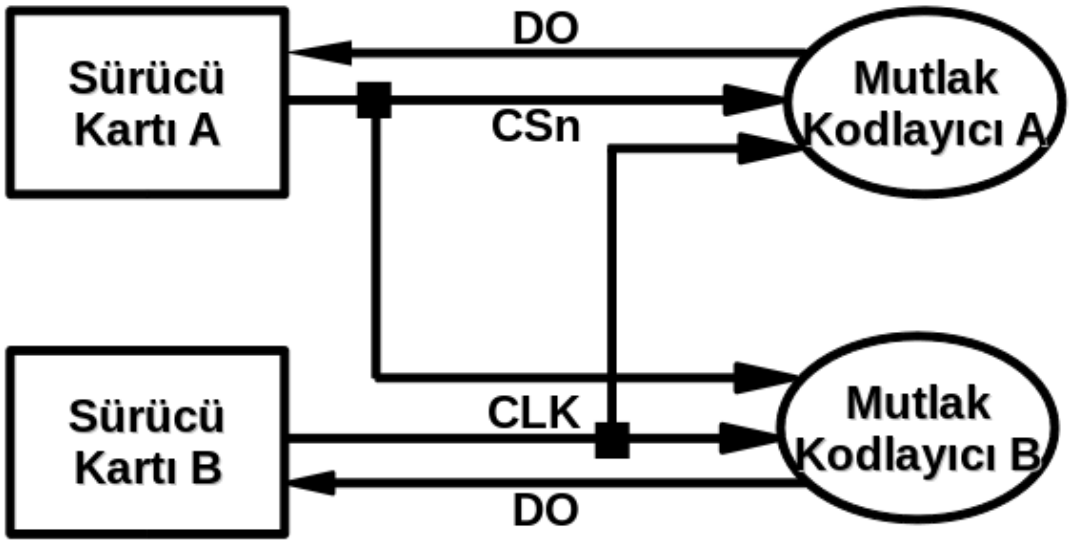
Şekil 3.2: Mutlak kodlayıcının iletişim yapısı.

Sürücü kartlarının üzerinde bulunan dijital giriş ve çıkışlar kullanılarak mutlak kodlayıcılardan veri alımı gerçekleştirilebilir. Kodlayıcıların kullandığı iletişim protokolüne uygun olarak CSn ve CLK bacaklarına sırasıyla uygun gerilimler sağlanarak DO bacağından verinin bitleri sırasıyla teker teker okunabilir (Çizelge 3.7).

Çizelge 3.7: Mutlak kodlayıcıdan veri okuma algoritması.

İşlem no	İşlem	Açıklama
1	CSn: 0	Algılayıcının CSn bacağına mantık-0 gerilimi gönder.
2	Bekle	En az 500 ns bekle.
	DÖNGÜ:	Algılayıcının DO bacağından döngü her döndüğünde verinin bir bitini al.
3	CLK: 0	Algılayıcının CLK bacağına mantık-0 gerilimi gönder.
4	Bekle	En az 500 ns bekle.
5	CLK: 1	Algılayıcının CLK bacağına mantık-1 gerilimi gönder.
6	Bekle	En az 375 ns bekle.
7	DO: ?	Algılayıcının DO bacağının mantık değerini oku.
8	Bekle	En az 500 ns bekle.
9	**→DÖNGÜ**	DÖNGÜ 'yü 16 kere tekrar et.
10	CSn: 1	Algılayıcının CSn bacağına mantık-1 gerilimi gönder.
11	Bekle	En az 500 ns bekle.

Sürücü kartlarının 0-5V gerilim aralığında çalışan ve aynı konektörle ulaşılabilen 1 adet girişi ve 1 adet çıkışı bulunmaktadır. Dolayısıyla, 1 mutlak kodlayıcıyı 1 sürücü kartına bağlayarak kodlayıcılardan veri alma işlemi gerçekleştirilemez. Ancak, sürücü kartları ve kodlayıcılar ikişerli gruplara ayrılırsa bu iletişim mümkün olabilir (Şekil 3.3). Böylelikle iki mutlak kodlayıcıya aynı iki çıkış verilir ama veriler yine ayrı ayrı okunur.



Şekil 3.3: Sürücü kartları ile mutlak kodlayıcıların bağlantısı.

EK B-4'te bulunan programın 133-343 numaralı satırları arasında bulunan "absEncoderRead()" fonksiyonu yukarda bahsedilen bütün algoritmayı içinde barındırır ve tüm eklemlerin açı değerini mutlak kodlayıcılardan okur.

3.2 Eklem Yörünge Referanslarının Üretilmesi

Robotun eklem açılarının istenen değerlere ulaşması için her eklemine ayrı ayrı yörünge oluşturmak gerekmektedir. Eklem yörüngeleri oluşturulurken çeşitli kriterler göz önüne alınmalıdır.

3.2.1 Robotun belirlenen konfigürasyonu için eklem yörüngeleri oluşturma

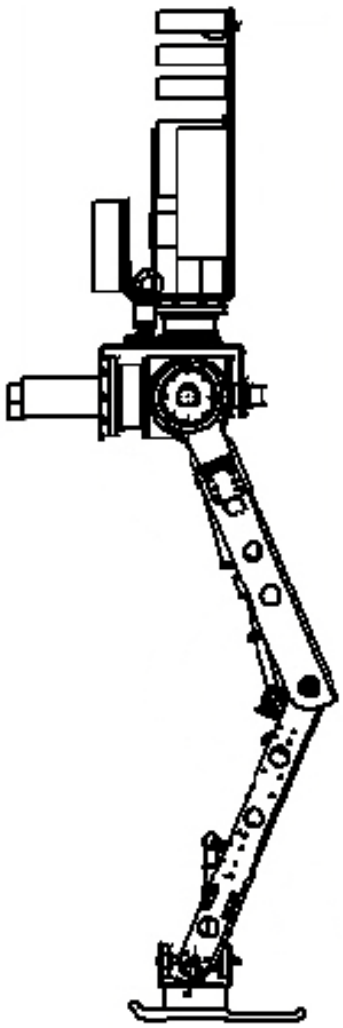
Robotun eklemlerinde bulunan servo sürüş sistemlerinde mutlak kodlayıcı bulunmadığı için motorlara gönderilecek referanslar mutlak değildir. Eklem referanslarının mutlak, motor referanslarının ise göreceli olması sistemde gerçek açı ölçme problemini doğurur. Bu yüzden robotu denemelerde aynı konfigürasyonda başlatmak düşünülmüştür. Böylelikle motor referansları bu konfigürasyondaki referans eklem açıları düşünülerek gönderilir. Çalışmanın bu kısmında, robotu denemelerden önce belirlenmiş konfigürasyona getirecek bir program yazılmıştır.

İlk olarak robotun başlangıç konfigürasyonunu belirlemek gerekmektedir. İstenen herhangi bir konfigürasyon başlangıç olarak kabul edilebilir. Buradaki önemli nokta robotun referans eklem açılarının mutlak olarak bilinmesidir. Daha sonra robot için oluşturulacak eklem referansları belirlenirken o anki eklem konumları ve başlangıç konfigürasyonu düşünülerek oluşturulacaktır. Sonra bu eklem referansları gömülü bilgisayara gönderilecek ve eklem referansları ile motor referansları arasındaki dönüşüm de buna göre yapılacaktır(Çizelge 3.8).

Çizelge 3.8: Belirlenen robot konfigürasyonu için eklem açıları.

Eklem	Başlangıç açı değeri (˚)
q_1, q_7	0
q_2, q_8	0
q_3, q_9	-21.4
q_4, q_{10}	44.1
q_5, q_{11}	-22.7
q_6, q_{12}	0

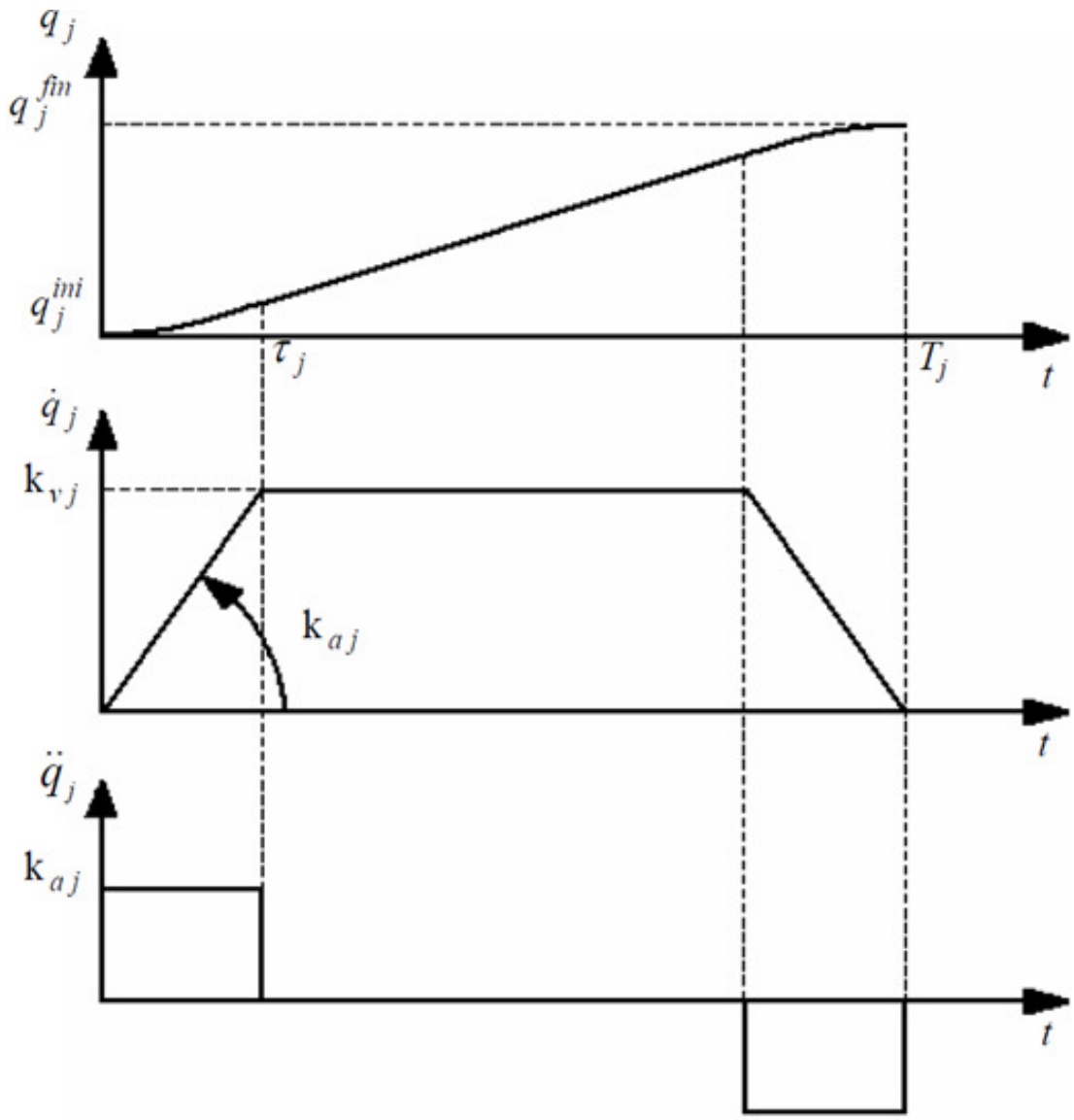
Bu çalışmada, iki ayaklı robot için belirlenen başlangıç konfigürasyonu "squad" pozisyonudur. Bu konumda robotun dizleri kırık, bilekleri ve gövdesi yere paraleldir. Bu pozisyon robotun yürümeye başlamadan önceki ilk pozisyonu olarak düşünülmüştür (Şekil 3.4).



Şekil 3.4: Robotun başlangıç konfigürasyonu.

Robotu herhangi bir konfigürasyondan "squad" pozisyonuna geçirmek için her eklemine yörünge oluşturmak gerekmektedir. Öncelikle, eklem yörüngesinin şekline karar verilmelidir. Robotun eklemlerinin "squad" pozisyonu için gereken açısal konumlara ulaşması için her eklem için yamuk hız profili kullanılmıştır.

Yamuk hız profili kullanılarak oluşturulmuş bir yörünge üç fazdan oluşmaktadır. Bu fazlar: sabit pozitif ivme fazı, sabit hız fazı ve sabit negatif ivme fazlarıdır. Yörüngede ilk hız ve son hız sıfırdır. Böylece yörüngenin konum ve hız profilleri sürekli, ivme profili ise süreksizdir. Kinematik kısıtlar düşünüldüğünde pozitif ve negatif ivmelenme fazlarının simetrik olması daha uygundur[19] (Şekil 3.5).



Şekil 3.5: Oluşturulan konum, hız ve ivme profilleri [19].

Yörünge oluşturmak için, her eklemin o anki açısal konumu (q_j^{ilk}), izin verilen maksimum hızı (k_{v_j}) ve ivmesinin (k_{a_j}) bilinmesi gerekir. Her eklemin o anki açısal konumu, robotun eklemlerine monte edilmiş olan mutlak kodlayıcılar tarafından okunur. Eklemlerin maksimum hız ve ivme değerleri ise mekanik kısıtlar göz önüne alınarak belirlenmiştir. Yörünge oluşturmaya başlamak için, her eklemin (j) **(3.1)**'deki koşulu sağlaması gerekir.

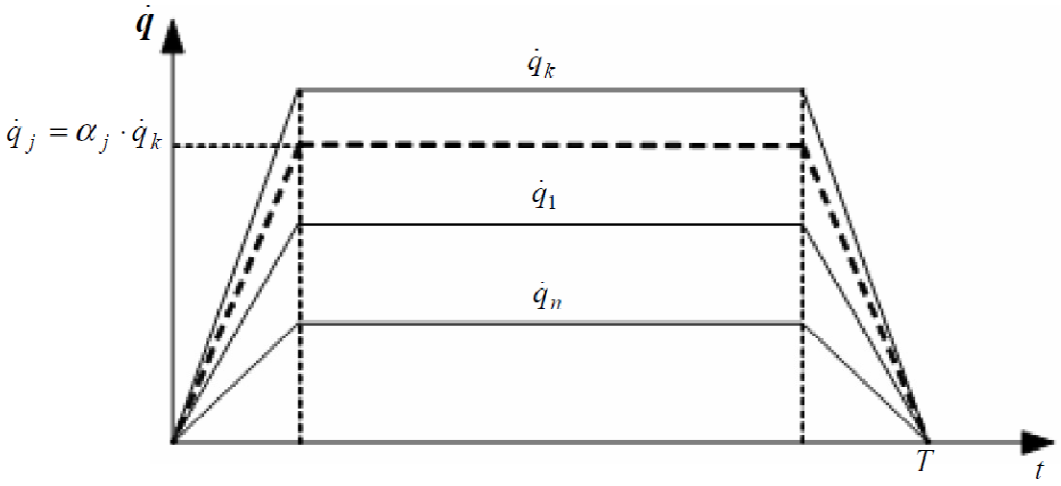
$$|D_j| > \frac{k_{v_j}^2}{k_{a_j}} \qquad \textbf{(3.1)}$$

Eğer, **(3.1)**'deki koşul sağlanamazsa yani maksimum hızların aşağıdaki denkleme göre yeniden düzenlenir.

$$k_{v_j} = \sqrt{|D_j| k_{a_j}} \tag{3.2}$$

Sonuç olarak, **(3.1)**'deki koşulu sağlayamayan eklemlerin maksimum hızları **(3.2)**'deki eşitliğe göre yeniden düzenlenir.

Bu çalışmadaki gibi birçok serbestlik derecesi barındıran mekanizmalarda, hareketin eşgüdümü açısından yörüngelerin homotetik olarak oluşturulması gerekir. Böylelikle ivmelenme, yavaşlama süreleri ve toplam hareket süreleri tüm eklemlerde aynı olur (Şekil 3.6).



Şekil 3.6: Homotetik hız profilleri [19].

Yörünge oluşturmaya başlamadan önce, ortak hareket zamanının(T_{ort}) belirlenmesi gerekir. Bunun için her eklemin ayrı ayrı maksimum hızlarına ve ivmelerine göre hareket zamanları(T_j) bulunur **(3.3)**.

$$T_j = \tau_j + \frac{|D_j|}{k_{v_j}} \tag{3.3}$$

Bu denklemdeki (τ_j) ifadesi **(3.4)**'deki denklemle tanımlanmıştır.

$$\tau_j = \frac{k_{v_j}}{k_{a_j}} \tag{3.4}$$

Her eklem için bulunan hareket zamanlarının en büyüğü ortak hareket zamanı olarak seçilir**(3.5)**.

$$T_{ort} = \max\ (T_j) \tag{3.5}$$

Ortak hareket zamanı belirlendikten sonra geriye tüm eklemlerin hareket sırasındaki hızları($k_{v_j}^*$) ve ivmelerinin($k_{a_j}^*$) bulunması gerekir. Bu değerleri bulurken her eklem için hız katsayısı(λ_j) ve ivme katsayısı(υ_j) tanımlanır **(3.6,3.7)**.

$$k_{v_j}^* = \lambda_j \,.\, k_{v_j} \tag{3.6}$$

$$k_{a_j}^* = \upsilon_j \,.\, k_{a_j} \tag{3.7}$$

Her eklemin hız katsayısı ve ivme katsayısı o eklemlerin maksimum hızlarının ne kadarının kullanıldığını gösterir. Bu katsayıların değerlerinin bulunması ise **(3.8)**'de verilmiştir.

$$\begin{cases} \lambda_1 = min\left[1,\ \dfrac{k_{v_j}\,.\,|D_1|}{k_{v_1}\,.\,|D_j|}\right] \\ \upsilon_1 = min\left[1,\ \dfrac{k_{a_j}\,.\,|D_1|}{k_{a_1}\,.\,|D_j|}\right] \end{cases} \tag{3.8}$$

Katsayı değerlerinin bulunmasıyla oluşturulacak yörüngelerin tüm parametreleri belirlenmiştir. Yörüngeler **(3.9)**'daki gibi oluşturulur.

$$q_j(t) = \begin{cases} q_j^{ilk} + D_j \dfrac{t^2}{2\tau_{ort}(T_{ort} - \tau_{ort})} & for\ 0 \le t \le \tau_{ort} \\ q_j^{ilk} + D_j \dfrac{(2t - \tau_{ort})}{2(T_{ort} - \tau_{ort})} & for\ \tau_{ort} \le t \le T_{ort} - \tau \\ q_j^{ilk} + D_j \dfrac{(T - \tau_{ort})^2}{2\tau_{ort}(T_{ort} - \tau_{ort})} & for\ T_{ort} - \tau \le t \le T_{ort} \end{cases} \tag{3.9}$$

Buradaki ortak hızlanma ve yavaşlama zamanı (τ_{ort}) ise **(3.10)**'daki gibi tanımlanır.

$$\tau_{ort} = \frac{\lambda_j \,.\, k_{v_j}}{\upsilon_j \,.\, k_{a_j}} \tag{3.10}$$

Böylelikle, anlık eklem açıları mutlak kodlayıcılardan alarak robotun başlangıç konfigürasyonundaki referans eklem açılarına eklemlerin sürülmesi için yamuk hız profili kullanılarak yörünge oluşturulmuştur. Oluşturulan bu eklem yörüngeleri de robotun örnekleme zamanına göre ayrıklaştırılarak bir dosya içine kaydedilmiştir. Şekil 3.7'de homotetik yamuk hız profilleri kullanarak robotun başlangıç konfigürasyonu için eklem yörüngesi oluşturmak için yazılmış programın akış şeması verilmiştir.



Şekil 3.7: Eklemlere başlangıç pozisyonu için yörüngeler oluşturulması.

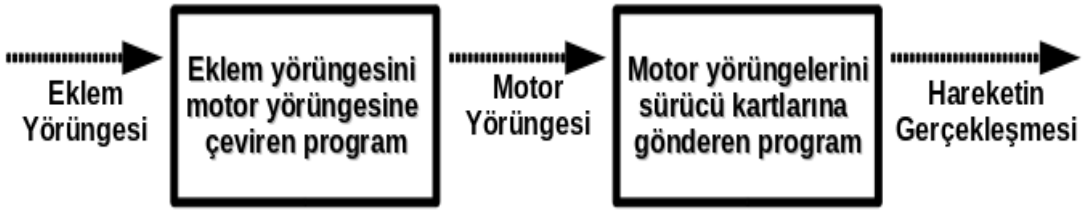
Robotun başlangıç konfigürasyonu için eklem yörüngeleri oluşturan bu programın da kodları C programa dilinde yazılmış olup, EK B.4'te bu program çeşitli açıklamalarıyla verilmiştir.

3.2.2 Robotun ters kinematiği kullanılarak eklem referansları oluşturulması

Robotun yürüme hareketini gerçekleştirebilmesi için robotun bilgisayarına gereken eklem yörüngesi bilgisi yüklenmelidir. Yürüme için gerekli eklem yörüngelerini oluşturan program, robotun üstünde daha önce yapılmış olan çalışmalarda yazılmıştır. MATLAB'ta yazılmış olan bu program, ADAMS benzetim ortamında test edilmiş ve başarılı olduğu görülmüştür[17]. MATLAB'ta yazılmış olan program bu programa atılacak toplam adım sayısı, adımlar arası mesafe ve adımın süresi gibi parametreler girilir. Program girilen bu bilgileri kullanarak robotun ayakuçlarında ve ağırlık merkezinde bulunan eksen takımları için yörünge üretir. Sonra program robotun daha önceki çalışmalarda[17] çıkarılmış olan kinematiğini kullanarak ayakların ve ağırlık merkezinin yörüngelerini alır ve bunları eklem yörüngelerine çevirir. Daha sonra oluşturulan eklem yörüngeleri belirlenmiş örnekleme periyoduna göre ayrıklaştırılır ve tablo halinde bir dosyaya yazılır. Oluşturulan bu dosya robotun bilgisayarına yüklenir ve istenen hareket sağlanır.

3.3 Eklem Yörünge Referanslarının Uygulanması

Robotun eklemlerine hareket yaptırabilmek için robotun üstündeki bilgisayarda iki adet program yazılmıştır. Birinci program, robota verilen referans eklem açısal konum değerlerini ilgili motorların açısal konumlarına çevirecek programdır. Bunun için bu ilk programın içinde eklemlerle motorlar arasında bulunan aktarım organlarının modelleri bulunacaktır. İkinci program ise ilk programla elde edilecek motorların açısal konum değerlerinin motorlara beslemek için sürücü kartlarına gönderecektir (Şekil 3.8).



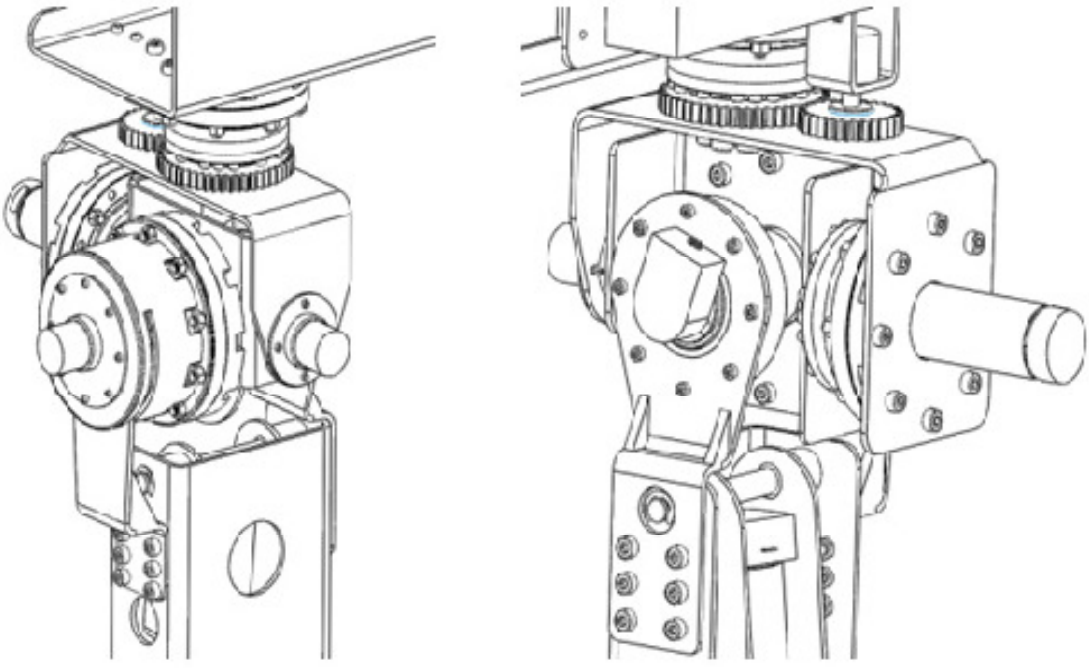
Şekil 3.8: Eklem yörüngeleri uygulama şeması.

3.3.1 Eklem referanslarının motor referanslarına dönüştürülmesi

Gönderilen eklem açılarını motor açılarına dönüştüren program, aslında genel olarak eklemler ile motorlar arasında bulunan aktarım organlarının modelidir. Programda bunun yanı sıra birkaç güvenlik algoritması da bulunmaktadır. Programdaki en önemli güvenlik algoritması, robota verilecek eklem yörüngelerinin eklemlerin dönebileceği maksimum ve minimum açılarının dışına çıkıp çıkmadığını kontrol eden kısımdır. Bu sayede robota, sadece mekanik olarak başarabileceği referans eklem açıları gönderilmiş olur. Bunun yanı sıra bu programda, kullanıcıya maksimum eklem hızlarını ve toplam deney zamanını gösteren kısımlar da bulunmaktadır.

Eklemlerin açıları ile motorların mil açıları arasındaki ilişkinin modelini üç farklı aktarma organı kullanıldığı için üçe ayırabiliriz. Mekanik tasarım kısmında da bahsedildiği üzere, aktarma organı olarak kalça eklemlerinde harmonik sürücüler, diz eklemlerinde bilyalı vidalar, bilek eklemlerinde ise bilyalı vidalar ile beraber kardan milleri kullanılmıştır.

Kalçada aktarma organı olarak kullanılan harmonik sürücüler, çalışma şekilleri bakımından diğer dişli sistemlerden farklıdır. Ayrıca diğer dişli sistemlerinden çok daha hassas ve kompakt yapıdadır(Şekil 3.9).



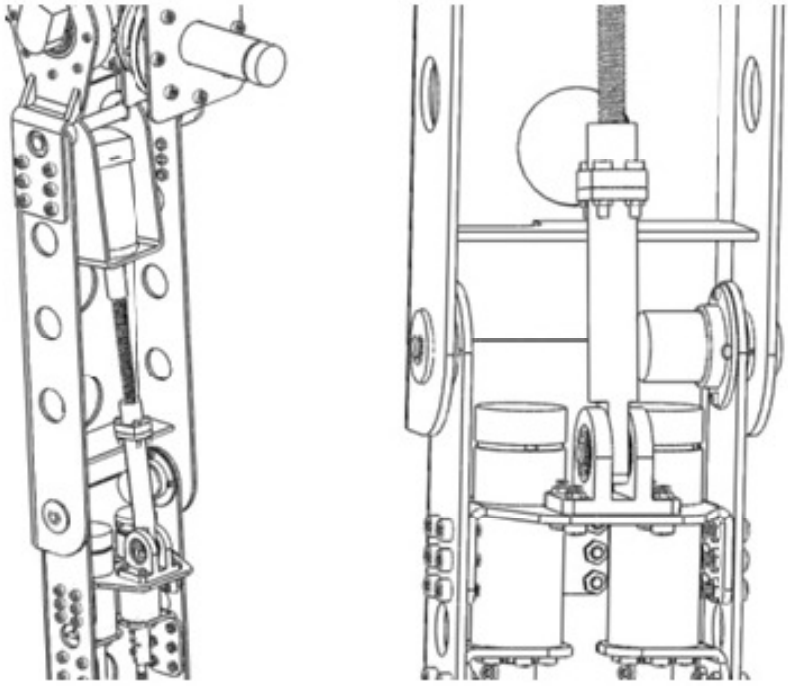
Şekil 3.9 : Kalça eklemlerinde bulunan aktarma organları.

Dolayısıyla kalça eklemlerinin açıları ile bağlı oldukları motorların millerinin açıları arasında çevirme oranları bulunmaktadır (Çizelge 3.9). Gerçekleşmesi istenen kalça eklemlerinin açısal konumlarından motorların açısal konumlarını bulmak için istenen eklem açısını ilgili eklemin çevirme oranıyla çarpmak yeterlidir.

Çizelge 3.9 : Kalça eklemlerindeki aktarma organlarının çevirme oranları.

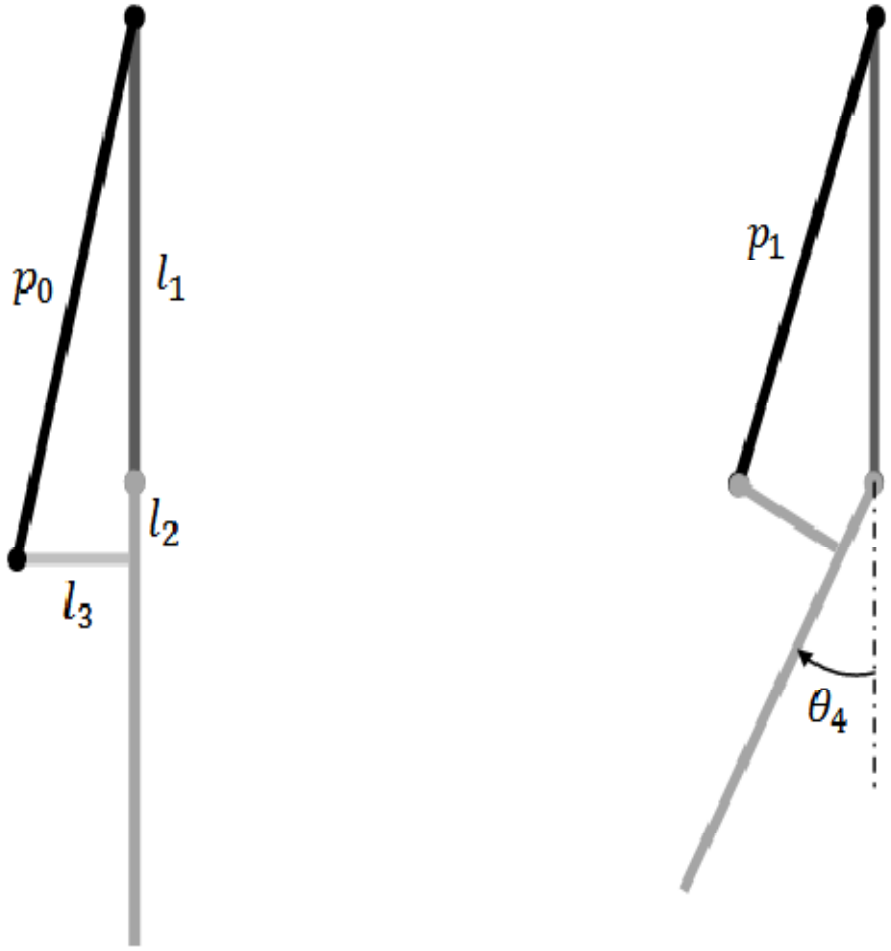
Eklem	Çevirme Oranı
q_1	120
q_2	160
q_3	160

Dizdeki aktarım organı için daha önceden de bahsedilen bilyalı vida sistemi kullanılmıştır(Şekil 3.10).



Şekil 3.10 : Diz Ekleminde bulunan aktarma organı.

Dizdeki bilyalı vidanın vida adımı 2 mm'dir. Motorun tam turunda oluşacak 2 mm'lik doğrusal hareket, bağlı olduğu diz ekleminin dairesel hareketine sebep olmaktadır. Eklemin belirli bir açısındaki vidanın hareket etme miktarı ile başka açısındaki vidanın hareket etme miktarı, eklem açısında aynı derecede değişikliğe sebep olmaz. Dolayısıyla diz ekleminin modeli doğrusal değildir(Şekil 3.11).



Şekil 3.11 : Dizdeki aktarma organının modeli.

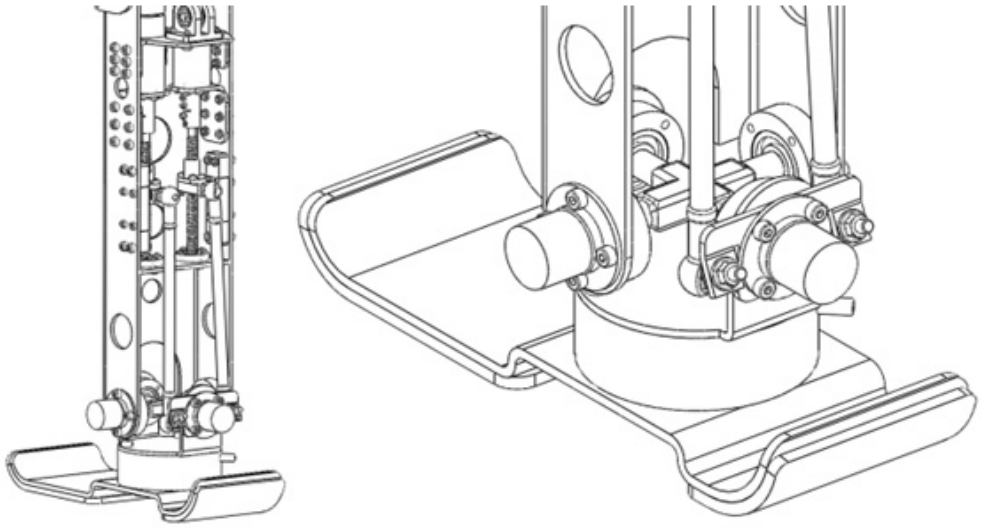
$$\Delta p = p_0 - p_1$$
$$= \sqrt{{l_3}^2 + (l_2 + l_1)^2}$$
$$- \sqrt{{l_2}^2 + {l_3}^2 + {l_1}^2 - 2l_1\sqrt{{l_2}^2 + {l_3}^2}cos(\pi - \tan^{-1}(l_3/l_2) - q_4)} \quad \textbf{(3.11)}$$

(3.11) kullanılarak q_4 için bilyalı vidada gereken uzama miktarı bulunmuştur.

$$\Theta_4 = \Delta p.\pi \quad \textbf{(3.12)}$$

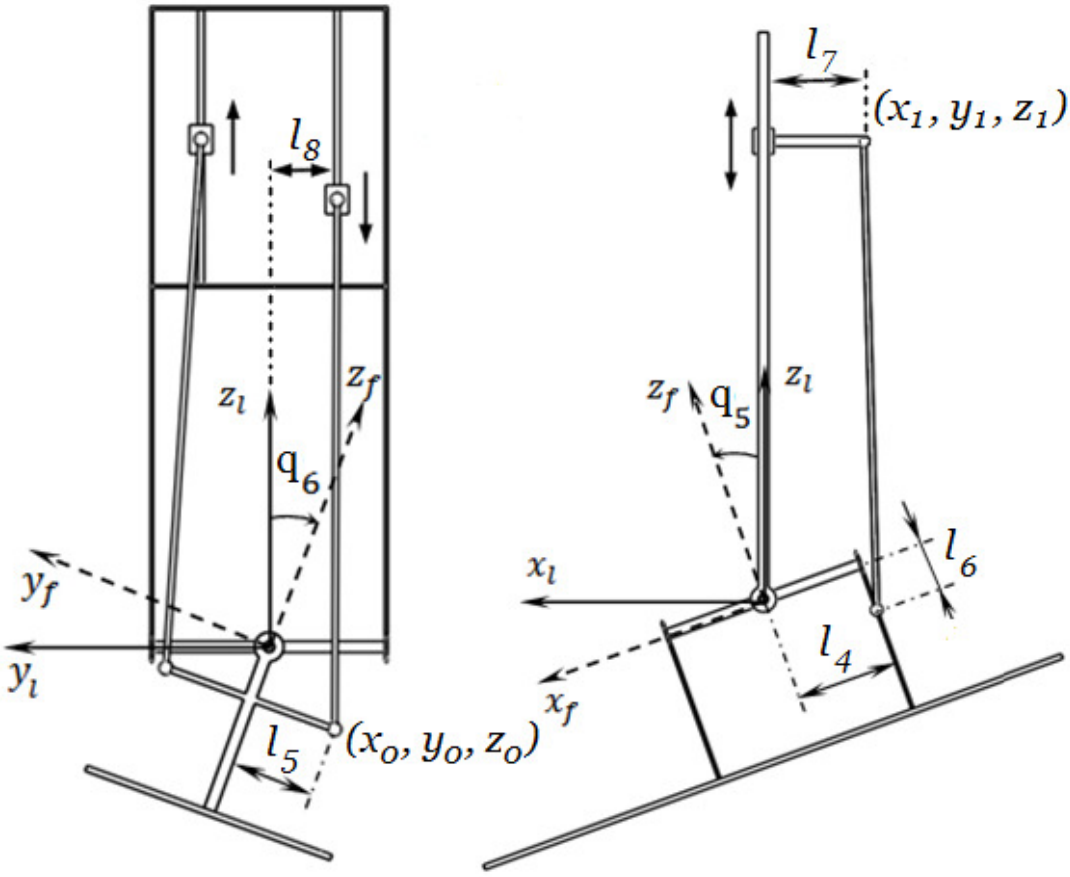
Motorun her bir turu 2 mm'lik uzama miktarına denk geldiği için Θ_4 açısı **(3.12)** ile hesaplanmıştır.

Bilekteki aktarım organı ise dizden daha karmaşık bir yapıdadır. Çünkü bilekteki iki serbestlik derecesi, bilekteki iki motorla da bağlantılıdır(Şekil 3.12).



Şekil 3.12 : Bilek eklemlerinde bulunan aktarma organları.

Bir motorun milinde gerçekleşen açısal yer değiştirme, iki eklem açısında da yer değiştirmeye sebep olur. Θ_5 ve Θ_6'ın birlikte aynı yöne doğru eşit olarak döndürülmesi sadece q_5'i, zıt yönlere doğru eşit olarak döndürülmesi ise sadece q_6'yı etkiler.



Şekil 3.13 : Bilekteki aktarma organlarının modeli.

Şekil 3.13'te görüldüğü üzere q_5 ve q_6 eksen takımlarının kesiştiği noktaya orjinleri gelicek şekilde iki adet eksen takımı yerleştirilmiştir. Bu eksen takımlarından biri bacakla, diğeri de bilekle beraber hareket etmektedir. Bilyalı vidaların uzaması gereken miktarı bulmadan önce aşağıda bulunan küresel mafsalın bacaktaki eksen takımına göre tanımlanması gerekmektedir. Bunun için rotasyon matrisi hesaplanmıştır.

$$\begin{bmatrix} {}^{l}x_0 \\ {}^{l}y_0 \\ {}^{l}z_0 \end{bmatrix} = \begin{bmatrix} Cq_5 & Sq_5Sq_6 & Sq_5Cq_6 \\ 0 & Cq_6 & -Sq_6 \\ -Cq_5 & Cq_5Sq_6 & Cq_5Cq_6 \end{bmatrix} \begin{bmatrix} {}^{f}x_0 \\ {}^{f}y_0 \\ {}^{f}z_0 \end{bmatrix} \quad \textbf{(3.13)}$$

(3.13) kullanılarak alttaki küresel mafsalın koordinatları bacak eksen takımında tanımlanmıştır. Daha sonra da alttaki küresel mafsalın koordinatları kullanılarak üstteki mafsalın koordinatları hesaplanmış ve bilyalı vidanın uzama miktarı **(3.14)** ile belirlenmiştir.

$${}^{l}z_1 = \sqrt{{l_9}^2 - \left({}^{l}x_1 - {}^{l}x_0\right)^2 - \left({}^{l}y_1 - {}^{l}y_0\right)^2} + {}^{l}z_0 \quad \textbf{(3.14)}$$

X ve Y eksenlerinde bulunan küresel mafsallar sabit olduğu için mafsallar sadece Z ekseni doğrultusunda hareket eder. Dolayısıyla (3.14)'te bu dikkate alınarak sadece z ekseni için hesap yapılmıştır. Z ekseni doğrultusunda uzama miktarı bulunduktan sonra Θ_5 açısı, motorun her bir turu 1 mm'lik uzama miktarına denk geldiği için **(3.15)** ile hesaplanmıştır.

$$\Theta_5 = {}^{l}z_1 . \pi/2 \quad \textbf{(3.15)}$$

Θ_5 için çıkartılan denklemler Θ_6 için de aynıdır. Ancak bazı uzunlukların değerleri farklıdır(Çizelge 3.10).

Çizelge 3.10 : Bilek aktarma organı modelinde kullanılan parametreler

Parametreler	Θ_5 için	Θ_6 için
${}^{f}x_0$	l_4	l_4
${}^{f}y_0$	$-l_5$	l_5
${}^{f}z_0$	l_6	l_6
${}^{l}y_1$	l_7	l_7
${}^{l}x_1$	$-l_8$	l_8

Robotun tüm eklemleri için daha önceki çalışmalarda çıkarılmış olan bu modeller[20], verilen eklem açılarının motor açılarına dönüşümü için yazılan bu programda kullanılmıştır. **(3.16)**'da bu dönüşüm denklemleri bulunmaktadır.

$$\begin{bmatrix} \theta_1 = 120\,.\,q_1 \\ \theta_2 = 160\,.\,q_2 \\ \theta_3 = 160\,.\,q_3 \\ \theta_4 = \pi\left[\sqrt{{l_3}^2 + (l_2 + l_1)^2} - \sqrt{{l_2}^2 + {l_3}^2 + {l_1}^2 - 2l_1\sqrt{{l_2}^2 + {l_3}^2}\cos(\pi - \tan^{-1}(l_3/l_2) - q_4)}\right] \\ \theta_5 = \frac{\pi}{2}\left[\sqrt{{l_9}^2 - \left((-l_8) - {}^{l}x_0\right)^2 - \left(l_7 - {}^{l}y_0\right)^2} + {}^{l}z_0\right] \\ \theta_6 = \frac{\pi}{2}\left[\sqrt{{l_9}^2 - \left(l_8 - {}^{l}x_0\right)^2 - \left(l_7 - {}^{l}y_0\right)^2} + {}^{l}z_0\right] \end{bmatrix} \quad \textbf{(3.16)}$$

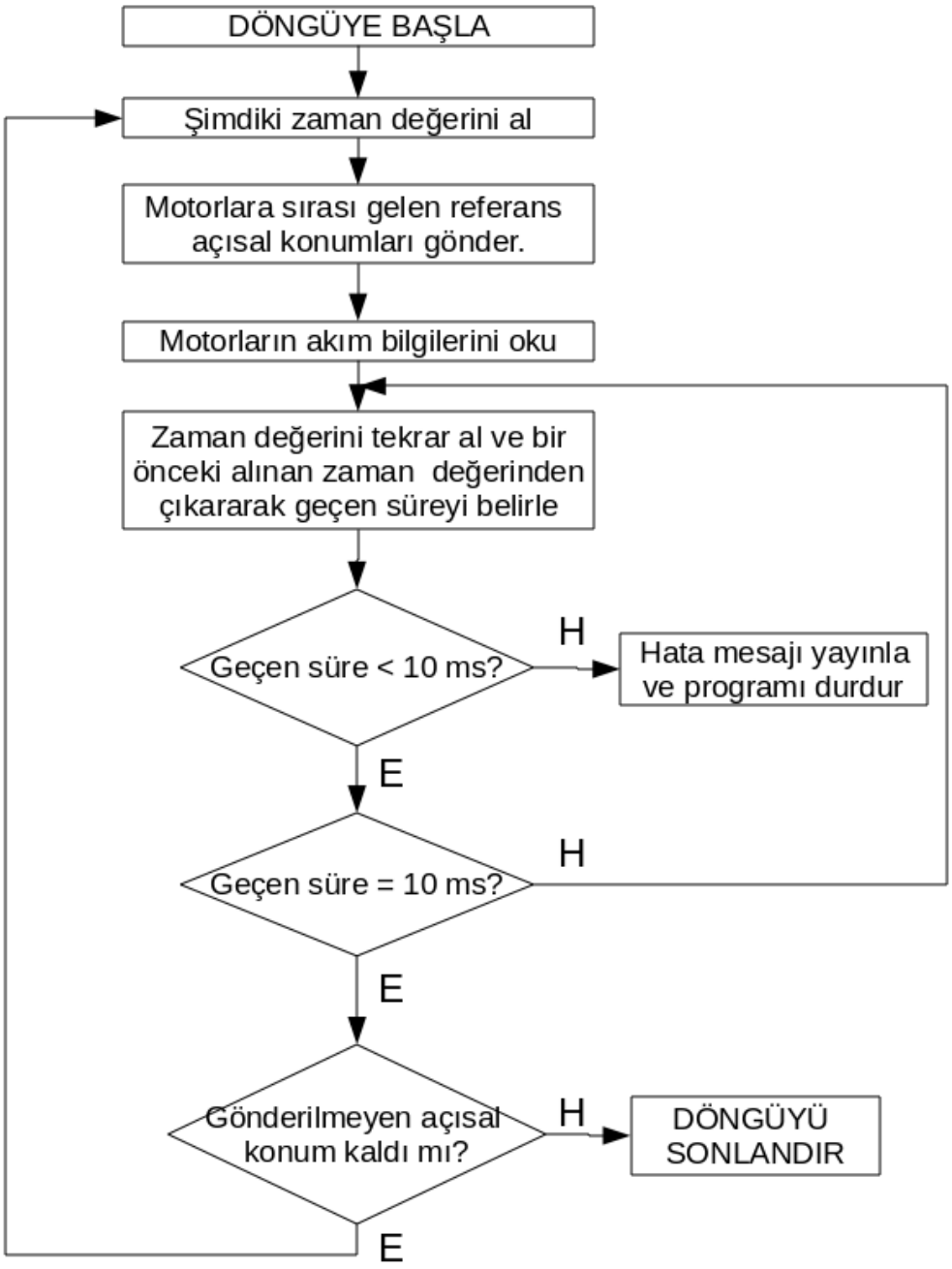
Deneylerde robot hareket başlamadan önce bu program çalıştırılarak verilen eklem referanslarından motor referansları oluşturulmaktadır. Yazılan bu program ayrıca, artırımlı kodlayıcılardan da motor millerinin ilk konumunu okumaktadır. Okunan bu değerler, ofset olarak bulunan bütün motor yörünge bilgilerine eklenmektedir. C dilinde yazılmış olan bu programın kodları EK B.1'de mevcuttur.

3.3.2 Motor referanslarının uygulanması

Robota verilen eklem yörüngeleri motor yörüngelerine çevrildikten sonra bu motor yörüngeleri motorlara uygulatılmıştır. Motor millerini verilen referanslara göre sürmek için yazılmış bu program, robotun bütün hareketini sağlayan ana programdır. Bu programda, sürücü kartları ile CANbus üzerinden iletişim kurulacağı için CANbus iletişim protokolünü içinde barındıran "can4linux.h" kütüphanesi de kullanılmıştır. Bu kütüphanenin kodları EK B.2'de verilmiştir. Ayrıca bu program sürekli olarak motorlara komut veren algoritmadan başka, içinde çeşitli güvenlik algoritmaları ile durum bildiren algoritmalar da barındırır.

Programda kullanılan güvenlik algoritmaları, robotun hareketini belirlenen güvenli sınırlar dâhilinde yapmasını sağlamak içindir. Robota uygulatılacak yörüngenin süresi ve tüm motor millerinde oluşacak maksimum hızlar, robot harekete başlamadan önce ekran çıktısı olarak sunulur. Böylelikle, gerekli kontroller yapılarak, uygun görülürse hareketin başlamasına onay verilir.

Motorların hareketine izin verildikten sonra, program belirlenen örnekleme zamanında (10 ms) sürekli olarak sürücü kartlarına sırasıyla motor yörüngelerinin bulunduğu dosyadan alınan yörünge bilgilerini gönderir. Bu aşamada örnekleme zamanı kontrolü için de ayrı bir algoritma yazılmıştır (Şekil 3.14).

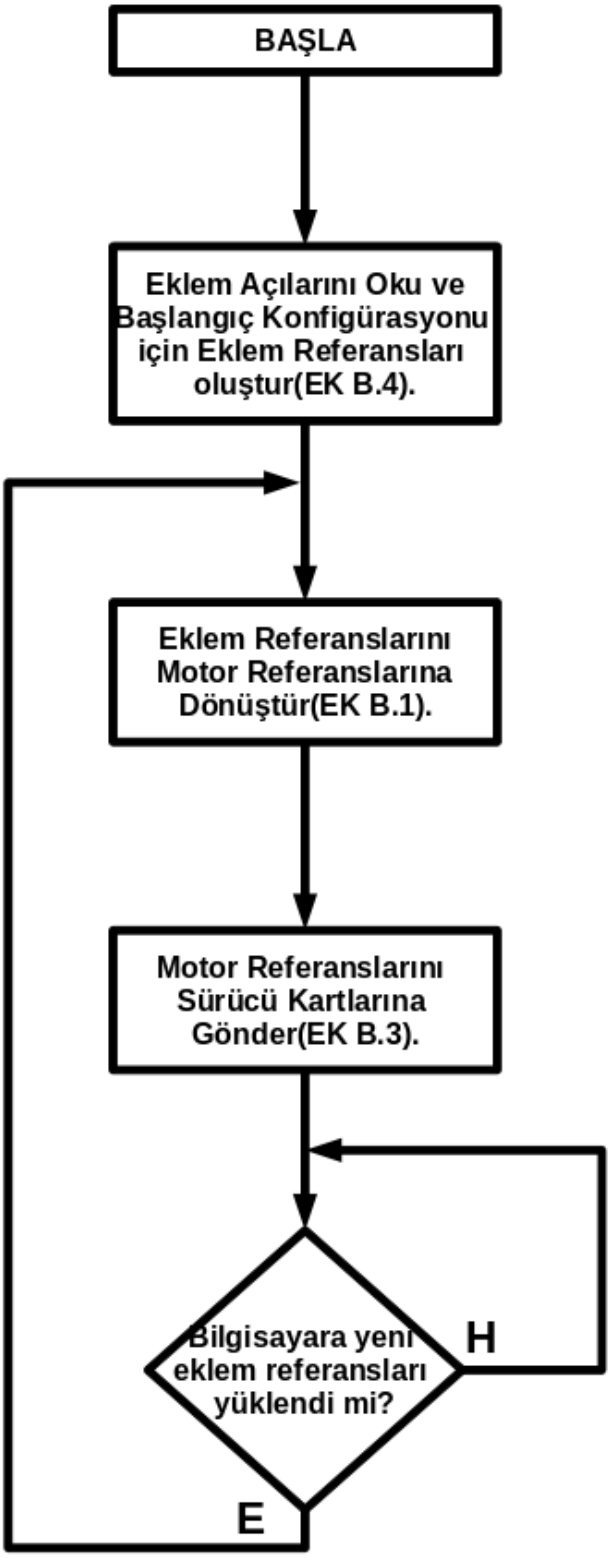


Şekil 3.14: Sürücü kartlarına konum referanslarının gönderilmesi.

Program, sürücü kartlarına bilgileri gönderdikten sonra geçen zamanı ölçer ve örnekleme zamanı geçilmişse hata verir. Eğer örnekleme zamanına ulaşılmamışsa bir sonraki yörünge bilgilerini göndermek için örnekleme zamanına ulaşılıncaya kadar bekler.

Analiz yapabilmek için robotun hareketi boyunca çeşitli durumların gözlemlenmesi gerekmektedir. Deneyler sırasında motorların açısal konumları ve çektiği akımlar bir dosyaya kaydedilmiş ve 5. Bölüm'de bunlar yorumlanmıştır. Bu programın da kodları C dilinde yazılmış olup EK B.3'te çeşitli açıklamalarıyla beraber verilmiştir.

Sonuç olarak, bilgisayarın içinde derlenip çalışabilir hale getirilen bütün kodlar, işletim sistemi üzerinden Şekil 3.15'teki akış diyagramına uygun olarak çalıştırılmıştır.



Şekil 3.15: Programların genel akış şeması.

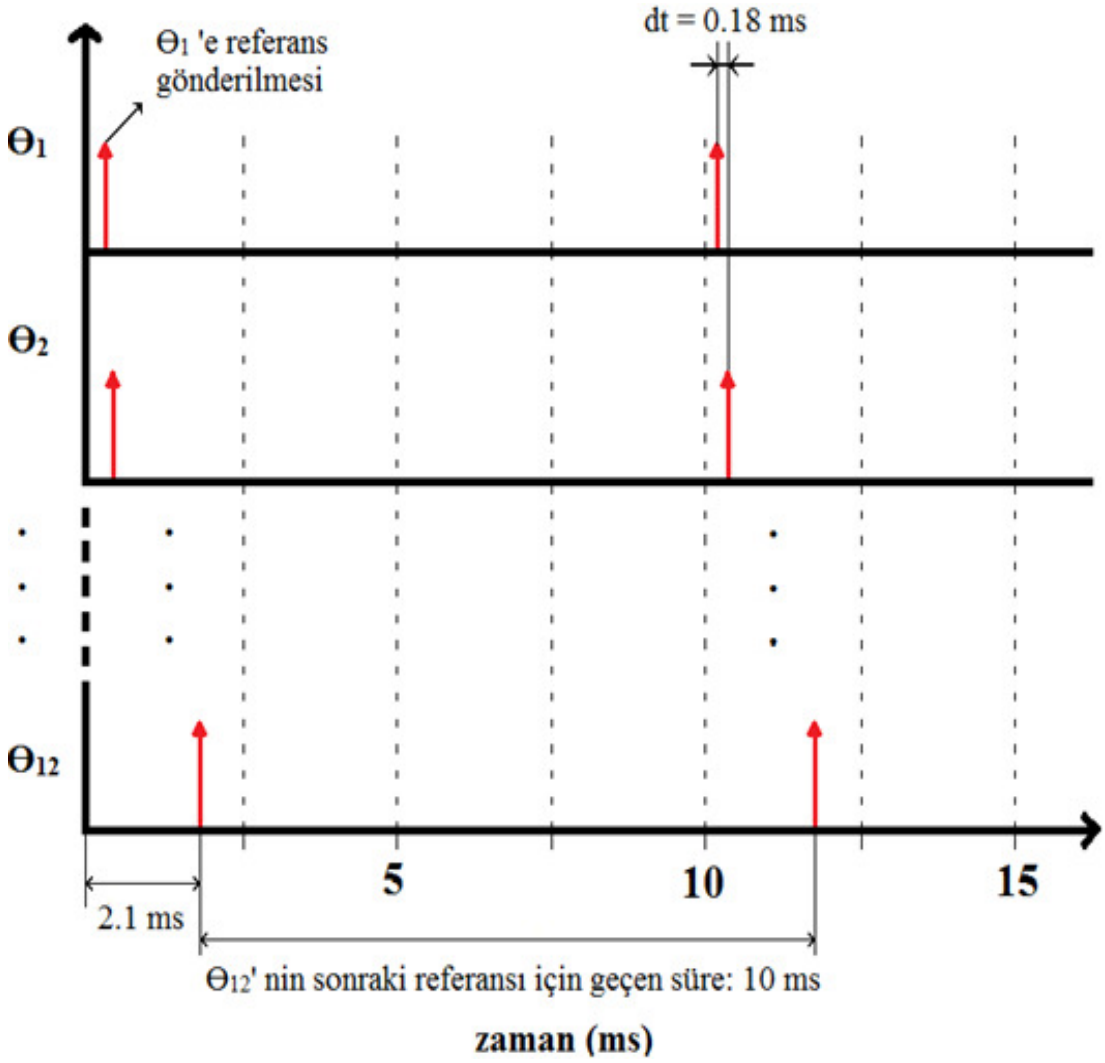
4. DENEYSEL ÇALIŞMALAR

Robotun donanımı ve yazılımı ile ilgili çalışmalar tamamlandıktan sonra robotun üstünde deneyler gerçekleştirilmiştir. Deneylere başlamadan önce örnekleme zamanı seçilmiş ve eklem kontrolcülerinin parametreleri ayarlanmıştır. Ayrıca bu bölümde, deneyleri yorumlayabilmek adına çeşitli değişkenlerin ölçülme ve hesaplanma şekilleri de açıklanmıştır.

4.1 Örnekleme Zamanı Seçimi

Robotun örnekleme zamanı seçilirken birçok kısıt göz önünde bulundurulmalıdır. İlk kısıt, sürücü kartlarının örnekleme zamanıdır. Seçeceğimiz örnekleme zamanı, sürücü kartlarının örnekleme zamanından büyük olmalıdır. Aksi takdirde sürücü kartlarına örnekleme zamanı içinde birden çok referans bilgisi gönderilebilir. Sürücü kartlarının örnekleme zamanı Maxon Motors tarafından 1 ms seçildiği için [18], robotun örnekleme zamanı bu değerden büyük bir süre seçilmelidir. İkinci önemli kısıt ise, bir örnekleme zamanı içinde gereken bütün CANbus haberleşmesinin eksiksiz tamamlanabilmesidir. Seçilen CANbus hızının 1 Mb/s olduğu düşünüldüğünde bir örnekleme zamanı içinde yapılması gereken CANbus haberleşmesinin ne kadar süreceği teorik olarak hesaplanabilir. Bu hesaplamayı yapabilmek için bir örnekleme zamanı içinde 12 sürücü kartına da yeni referans konum bilgisi gönderileceği, referans göndermek için bir CAN mesajı gönderip bir tane de almak gerekeceği ve de bir CAN mesajının toplamda 11 bayt olacağı göz önünde bulundurulmalıdır. Gereken hesaplama yapıldığında bir sürücü kartına referans bilgisi göndermek için CANbus hattından 176 bit transfer gerçekleştirmek gerekeceği bulunur. Transfer hızı saniyede 1 megabit olduğu göz önüne alındığında bu süre bir sürücü kartı için 176 mikro saniye olarak bulunur. Referanslar 12 sürücü kartına da seri olarak gönderileceği için bir örnekleme zamanı içinde yapacağımız bit transferinin süresi yaklaşık 2.1 mili saniye bulunur. Bir örnekleme zamanı içinde CANbus haberleşmesinin başarıyla tamamlanabilmesi için seçilecek zamanın bulunan bu değerden büyük olması gerekir.

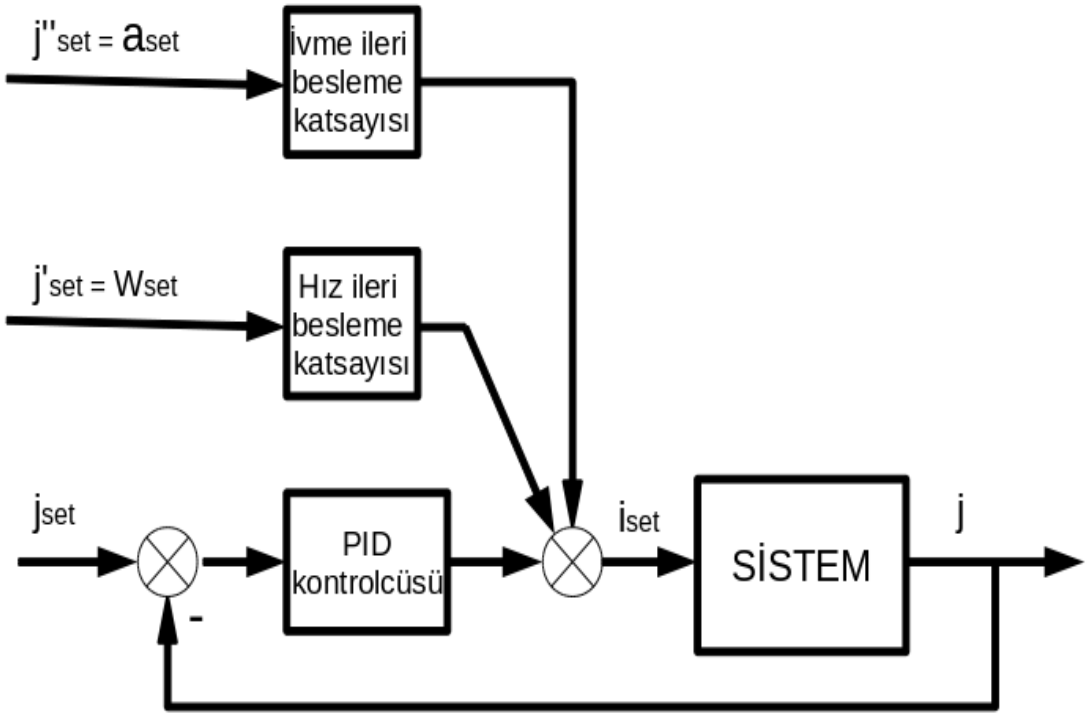
Sürücü kartlarına seri olarak referanslar gönderildikten sonra motorların da bu referansları seçilecek örnekleme zamanında gerçekleştirmesi gerekmektedir. Motorların referansları gerçekleştirme zamanı, verilecek referanslara, motor şaftında oluşan sürtünme kuvvetine ve motorun bağlı bulunduğu eklemin atalet momentine bağlıdır. Bu etkenleri gerçek robotun üstünde gerçek zamanlı olarak kestirmek zordur. Dolayısıyla, robottan beklenen hareketleri yaparken motorlarda oluşan hızlardan bir öngörü yapmak daha uygundur. Örnekleme zamanının 10 mili saniye seçilmesi öngörülmüştür. Örnekleme zamanı 10 ms olarak yapılan deneylerde de CANbus haberleşmesinin tamamlanamaması veya motorların verilen referansları takip edememesi gibi bir durum ortaya çıkmamıştır. Şekil 4.1'de, seçilen 10 milisaniye örnekleme zamanında sürücü kartlarına hangi anlarda motor referansları gönderildiği görülmektedir.



Şekil 4.1 : Sürücü kartlarına motor referanslarının gönderilmesi.

4.2 Eklem Kontrocülerinin Parametrelerinin Ayarlanması

Sürücü kartlarına CANbus haberleşme protokolü kullanılarak motor referansları iletilmektedir. Sürücü kartları da aldığı bu konum bilgilerini gerçekleştirmek için artırımlı kodlayıcıdan geri besleme alarak, bünyesinde barındırdığı kontrolcüsü ile motorları hareket ettirmektedir. Sürücü kartlarında bulunan kontrolcü, ivme ve hız ileri beslemesi eklenerek modifiye edilmiş bir PID kontrolcüsüdür(Şekil 4.2).



Şekil 4.2 : Eklemlerde bulunan kontrol çevrimi [18].

Üretici firma, sürücü kartlarında bulunan kontrolcünün parametrelerinin ayarlanması için bir otomatik parametre kestirim programı geliştirmiştir. Sürücü kartlarının kontrolcü parametreleri bu otomatik parametre kestirim programı tarafından bulunmuştur.

Program 3 aşamadan oluşmaktadır. Bunlar: tanılama, haritalama ve test etme aşamalarıdır. Program öncelikle sürücü kartına bağlı motora harmonik giriş uygular ve bu girişlerin sonucunda sistemin frekans cevabına bakarak sistemi tanılar. Tanılama bittikten sonra haritalama aşamasına geçilir. Bu aşamada kendi içinde bulunan katalogdan tanıladığı sisteme en uygun kontrolcü parametrelerini seçer. Son aşamada ise bu parametreleri kullanarak motora çeşitli konum referansları gönderir ve parametrelerin uygunluğunu test eder.

Robotun eklem kontrolcülerinin parametre kestirimi için Maxon firmasının geliştirmiş olduğu yukarda bahsedilen otomatik parametre kestirim programı kullanılmıştır. Robot vinç yardımıyla havada asılı tutulurken kestirilen parametreler, robot yere indiğinde de kullanılmıştır.

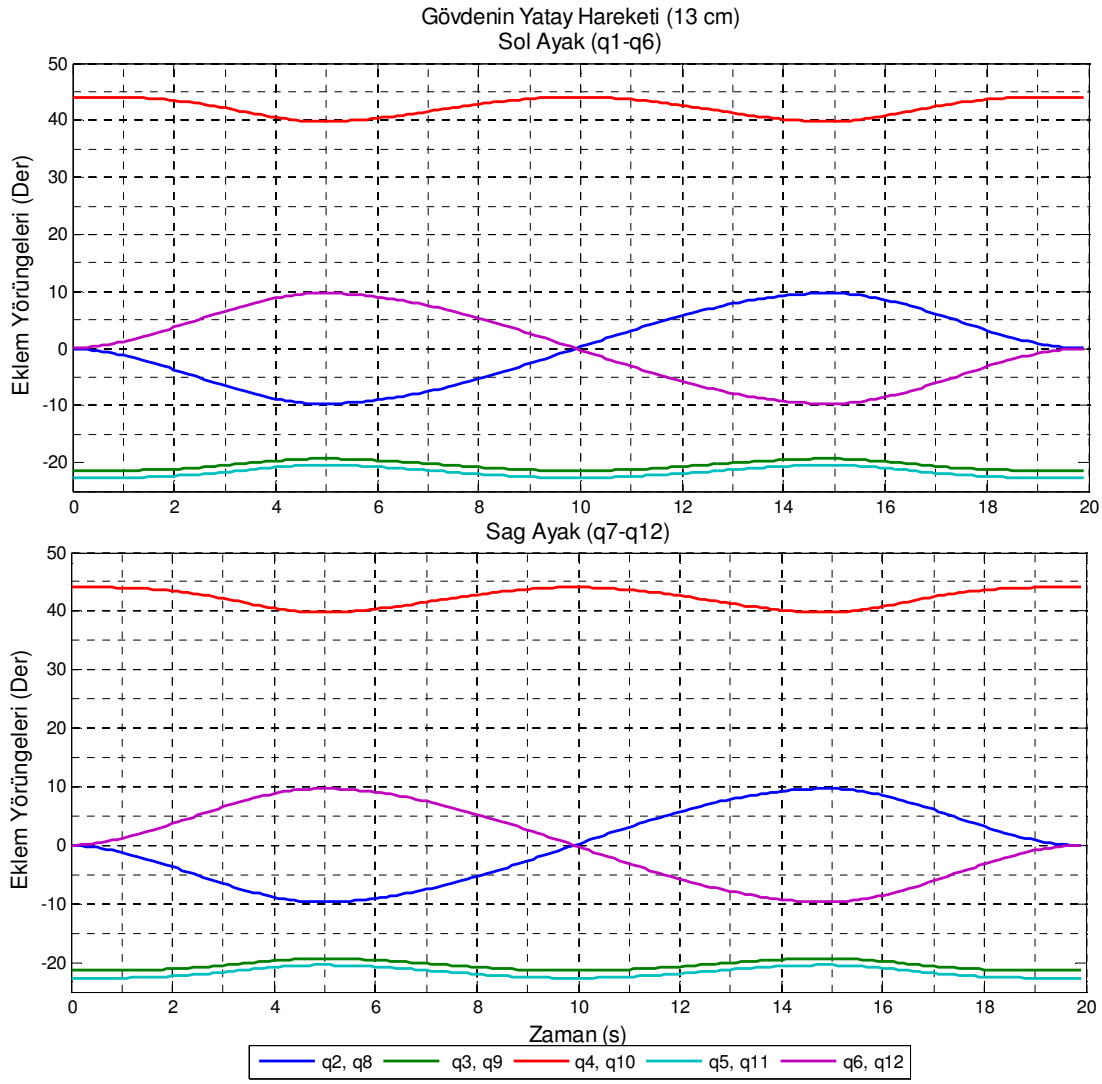
4.3 Robotun Verilen Referans Eklem Yörüngelerini Takip etmesi

Bu tez kapsamında robota 5 farklı hareket gerçekleştirmesi için referans eklem yörüngeleri oluşturulmuştur. Benzetim ortamında da test edilmiş bu yörüngeler, robotun elektronik yapısının ve iletişiminin test edilmesi amacıyla seçilmiştir. Yörüngeler, robotun iki ayağı yerdeyken gövdenin yatay ve düşey hareket etmesini kapsamaktadır.

Robotun gövdesi ilk olarak Z ekseninde hareket ettirilmiştir. Robotun gövdesini, Z ekseninde 5 cm ve 10 cm hareket ettiren iki deney yapılmıştır. Ayakların konumu ve yönelimi sabitken gövdenin yönelimi de sabit kalacak şekilde gövde aşağıya doğru hareket ettirilmiştir. Bu deneylerin sonuçları EK C.1-12'de verilmiştir.

Daha sonra robotun gövdesi Y ekseni doğrultusunda hareket ettirilmiştir. Ancak, gövdenin Y ekseni doğrultusundaki yatay hareketi düşey hareketten daha kapsamlıdır. Çünkü Z ekseni doğrultusundaki düşey hareket sırasında sadece q_3, q_4, q_5, q_9, q_{10}, q_{11} eklemleri hareket etmiştir. Fakat gövde Y ekseni doğrultusunda yatay olarak hareket ederken q_1 ve q_7 hariç tüm eklemler hareket etmiştir. Robotun gövdesini Y ekseni doğrultusunda 7 cm, 10 cm ve 13 cm hareket ettiren 3 farklı deney yapılmıştır. Bu hareketlerden 7cm'lik ve 10 cm'lik olanların sonuçları EK C.13-24'te verilmiştir. Gövdenin 13 cm'lik yatay hareketi, robotun yürümesi sırasında da kullanılacağı için önemlidir. Robot yürüme sırasında adım atmadan önce gövdesi yatay olarak 13 cm hareket ettirmesi gerekmektedir. Bu hareketle robotun ağırlık merkezinin izdüşümü destek ayağının ortasına düşer ve robot bu sayede diğer ayağını kaldırıp adım atabilir.

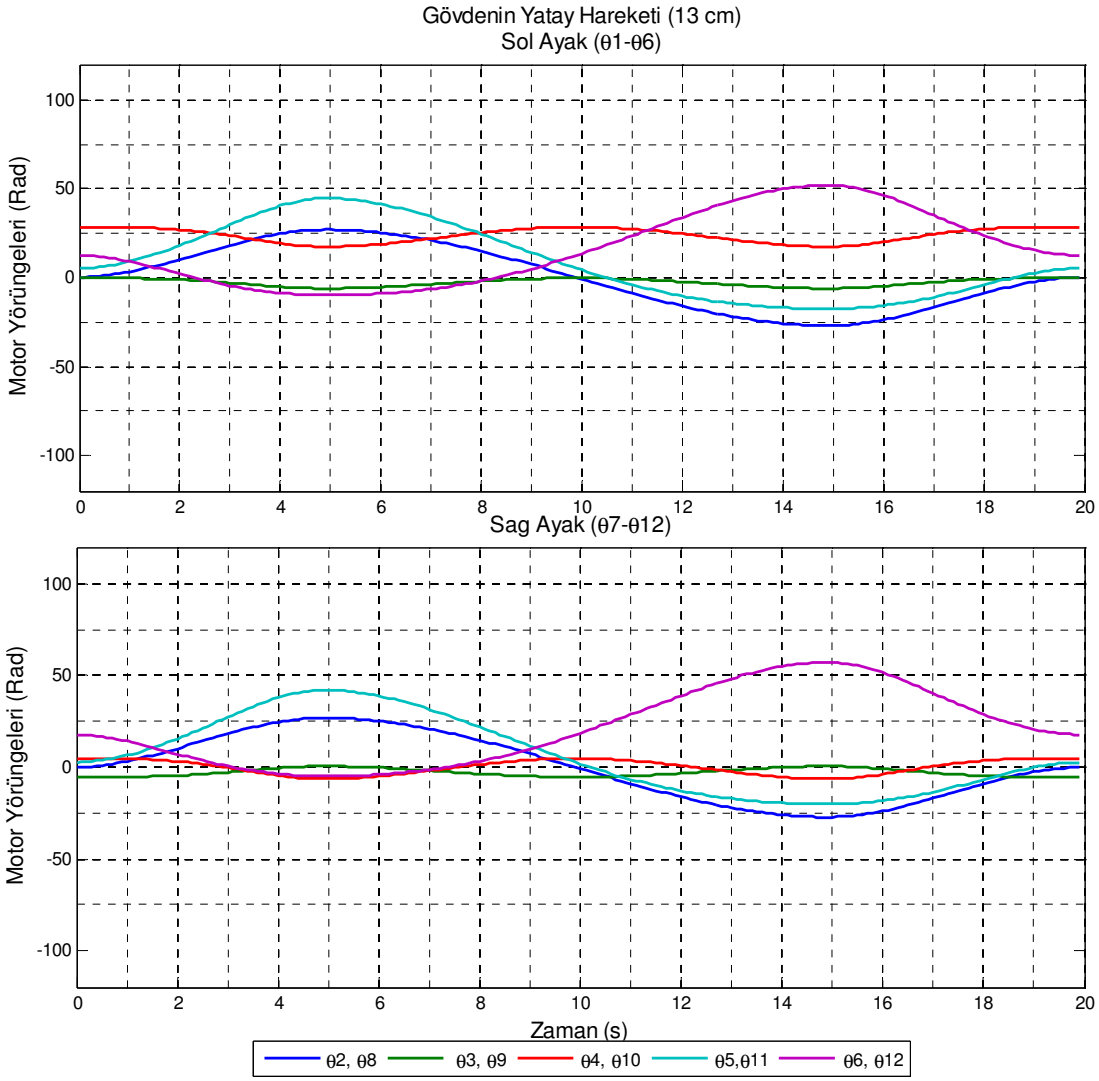
Gövdenin Y ekseni doğrultusundaki yatay hareketi için yapılan deneylerin her biri 20 saniyedir. Bu sürenin ilk 5 saniyesinde robotun gövdesi bahsi geçen mesafe kadar pozitif yönde hareket eder. Diğer 5 saniyede ise başlangıç konumuna geri döner. Sonraki 5 saniyede de negatif yönde bahsi geçen mesafe kadar ilerleyip, son 5 saniyede başlangıç konumuna tekrar geri döner.



Şekil 4.3 : Gövdenin 13 cm'lik yatay hareketi için referans eklem yörüngeleri.

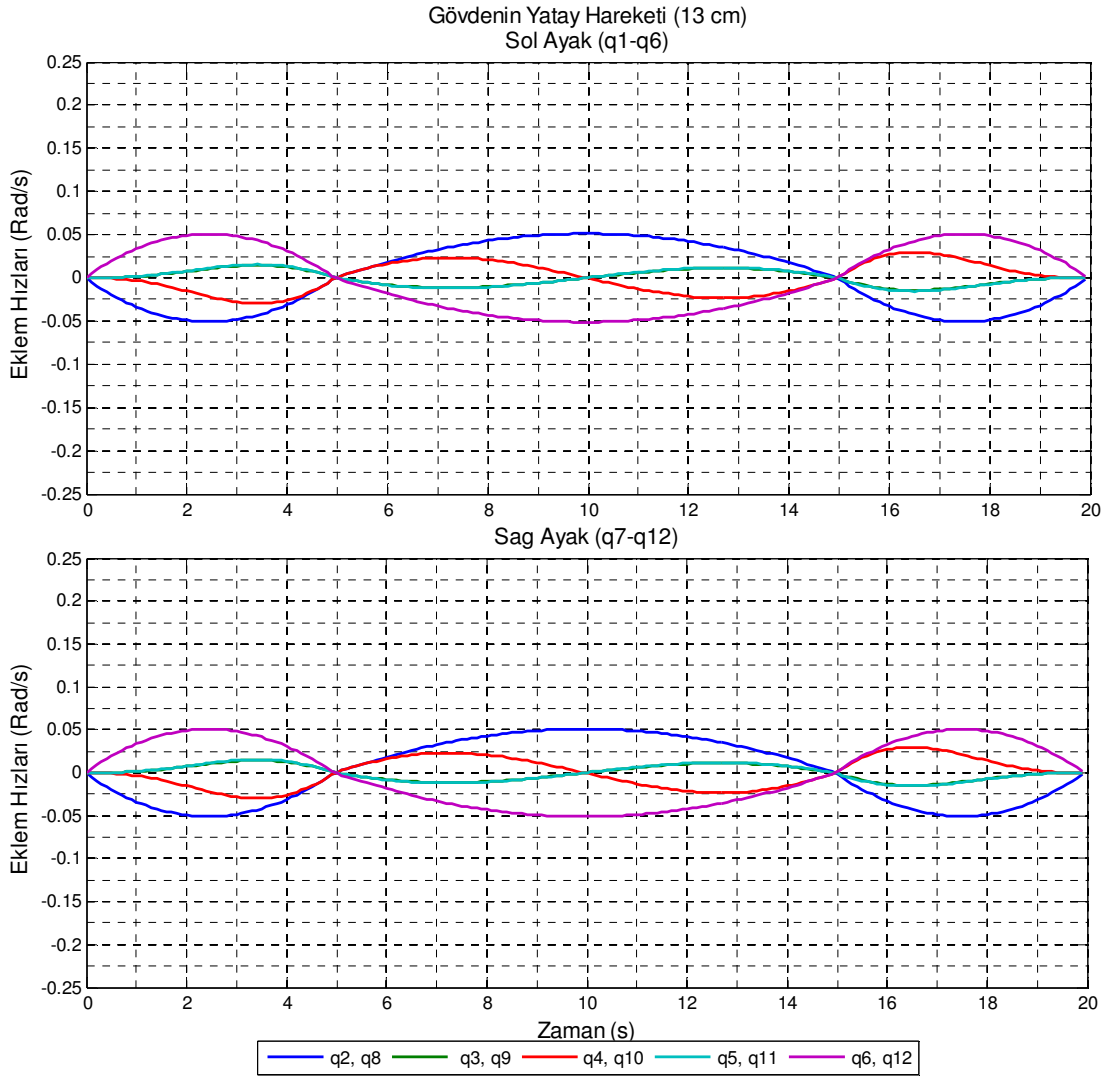
Robotun gövdesinin Y ekseni doğrultusunda 13 cm'lik yatay hareketini gerçekleştirmesi amacıyla önceki çalışmalarda[17] MATLAB programı kullanılarak oluşturulmuş referans eklem yörüngeleri Şekil 4.3'te verilmiştir. Bu yörüngelere bakıldığında eklem hareketlerinin robotun ölçülen eklem erimlerinin içinde olduğu görülmektedir. Bu yörüngeler daha sonra belirlenmiş örnekleme zamanına göre ayrıklaştırılmış ve bir tablo haline getirilmiştir. Tablo haline getirilmiş referans konum bilgileri son olarak da bir dosyaya yüklenerek robotun gömülü bilgisayarına aktarılmıştır.

Şekil 4.3'te verilen referans eklem yörüngeleri gömülü bilgisayara yüklendikten sonra, bu referans eklem yörüngeleri robot harekete başlamadan önce gömülü bilgisayar tarafından referans motor yörüngelerine çevrilmiştir. Bu referans motor yörüngeleri de Şekil 4.4 verilmiştir.



Şekil 4.4 : Gövdenin 13 cm'lik yatay hareketi için referans motor yörüngeleri.

Şekil 4.4'te dikkat edilmesi gereken nokta referans eklem yörüngelerinin iki bacakta da aynı olmasına rağmen referans motor yörüngelerinin farklı olmasıdır. Özellikle Θ_4 ve Θ_{10} motorlarının referanslarına bakıldığında bu farklılık görülebilir. Bu, tezin 2. bölümünde de bahsedilen artırımlı kodlayıcıların mutlak açı değerlerini okuyamamasından kaynaklanır. Mutlak açı değerleri eklemlere monte edilmiş mutlak kodlayıcılarla okunduğu için Θ_4 ve Θ_{10} motorlarına bu mutlak açı değerlerine uyması için farklı referanslar gönderilmiştir Bu motorlara gönderilen referanslar farklı olsa da eklem uzayından bakıldığında q_4 ve q_{10} eklemlerinin aynı şekilde hareket ettikleri görülür. Motor millerinin başlangıçta farklı konumlarda bulunmasından dolayı Θ_4 ve Θ_{10} motorlarının konum referansları arasında bir öteleme farkı bulunmaktadır. Ayrıca her eklemle motorlar arasında aktarma organları bulunduğu için motorlara verilen açısal konum referansları mertebe olarak daha büyüktür.

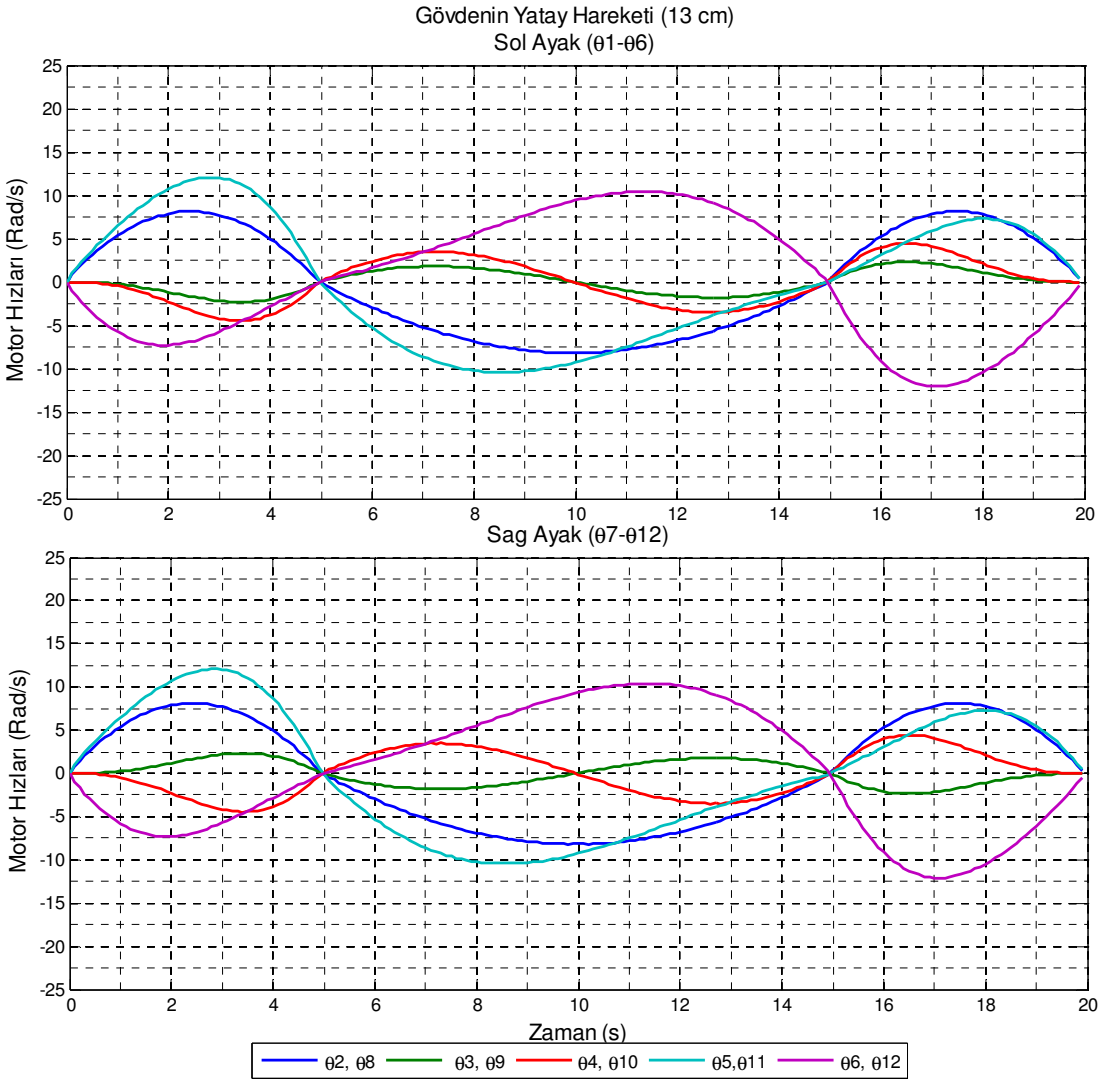


Şekil 4.5 : Gövdenin 13 cm'lik yatay hareketi için referans eklem hızları.

Referans eklem yörüngelerinin örnekleme zamanına göre ayrık türevi alınarak eklemlerin hızları hesaplanmıştır. Bu referans hızlarının hesaplanmasının nedeni eklemlerin hız profillerini incelemek ve sürekli olup olmadığını kontrol etmektir. Hesaplanan bu eklem hızları Şekil 4.5.'te verilmiştir.

Bu tez kapsamında robot üstünde yapılan denemeler, dinamik kuvvetlere maruz kalmamak için Şekil 4.4'te de görüldüğü gibi düşük hızlarda gerçekleştirilmiştir. Ayrıca Şekil 4.4'e bakıldığında tüm eklemlerin hızlarının sürekli olduğu da görülmektedir.

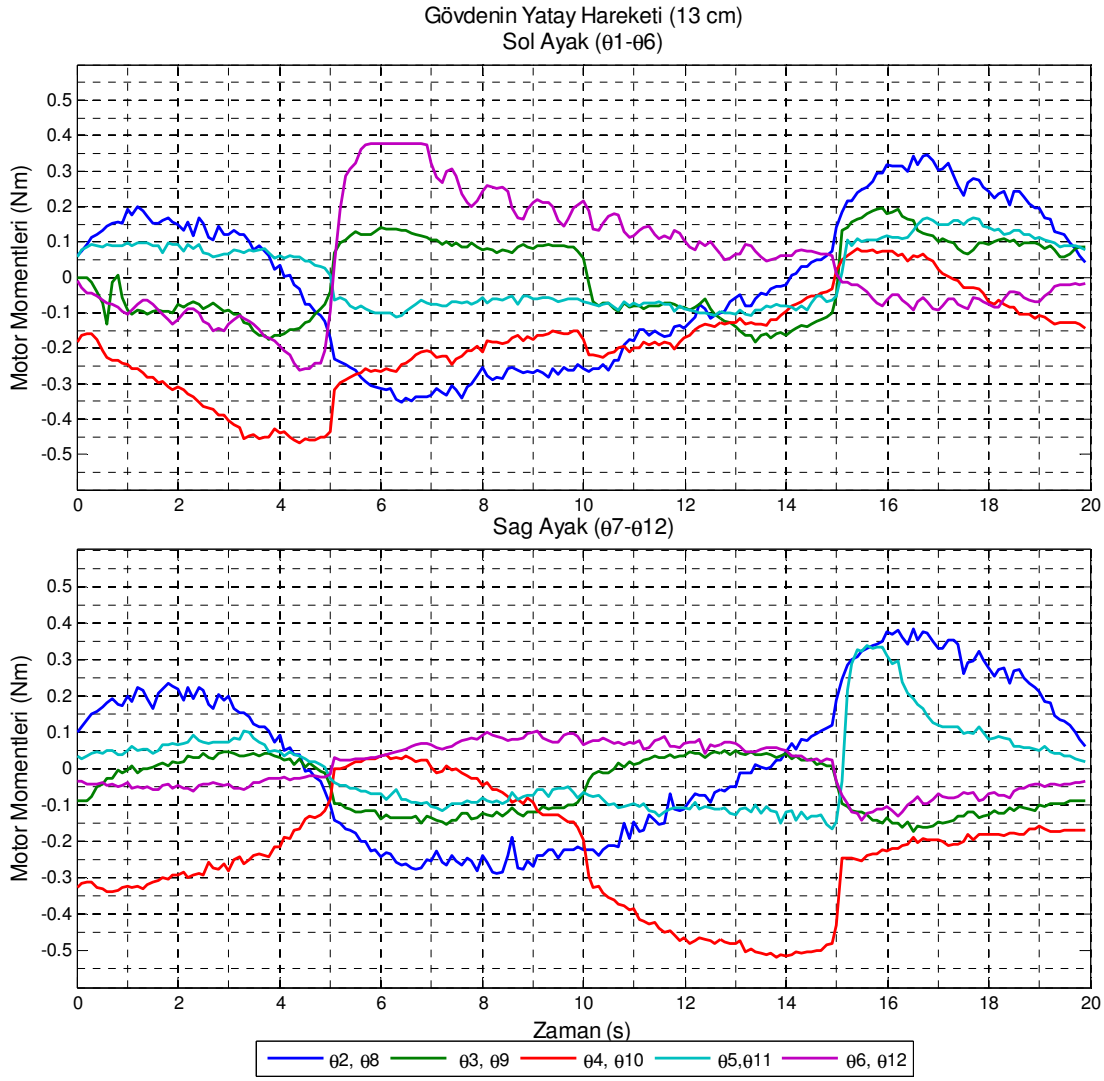
Eklem yörüngelerine yapıldığı gibi referans motor yörüngelerinin de örnekleme zamanına göre ayrık türevi alınarak motor hızları hesaplanmıştır. Hesaplanan bu eklem hızları Şekil 4.6.'da verilmiştir.



Şekil 4.6 : Gövdenin 13 cm'lik yatay hareketi için referans motor hızları.

Motorların hızlarının eklem hızlarına göre çok daha büyük olduğu Şekil 4.5'te ve Şekil 4.6'da görülmektedir. Aktarma organları, motorların üstünde oluşan momentleri azaltmaya yönelik mekanizmalar oldukları için motor hızları da buna bağlı olarak artar.

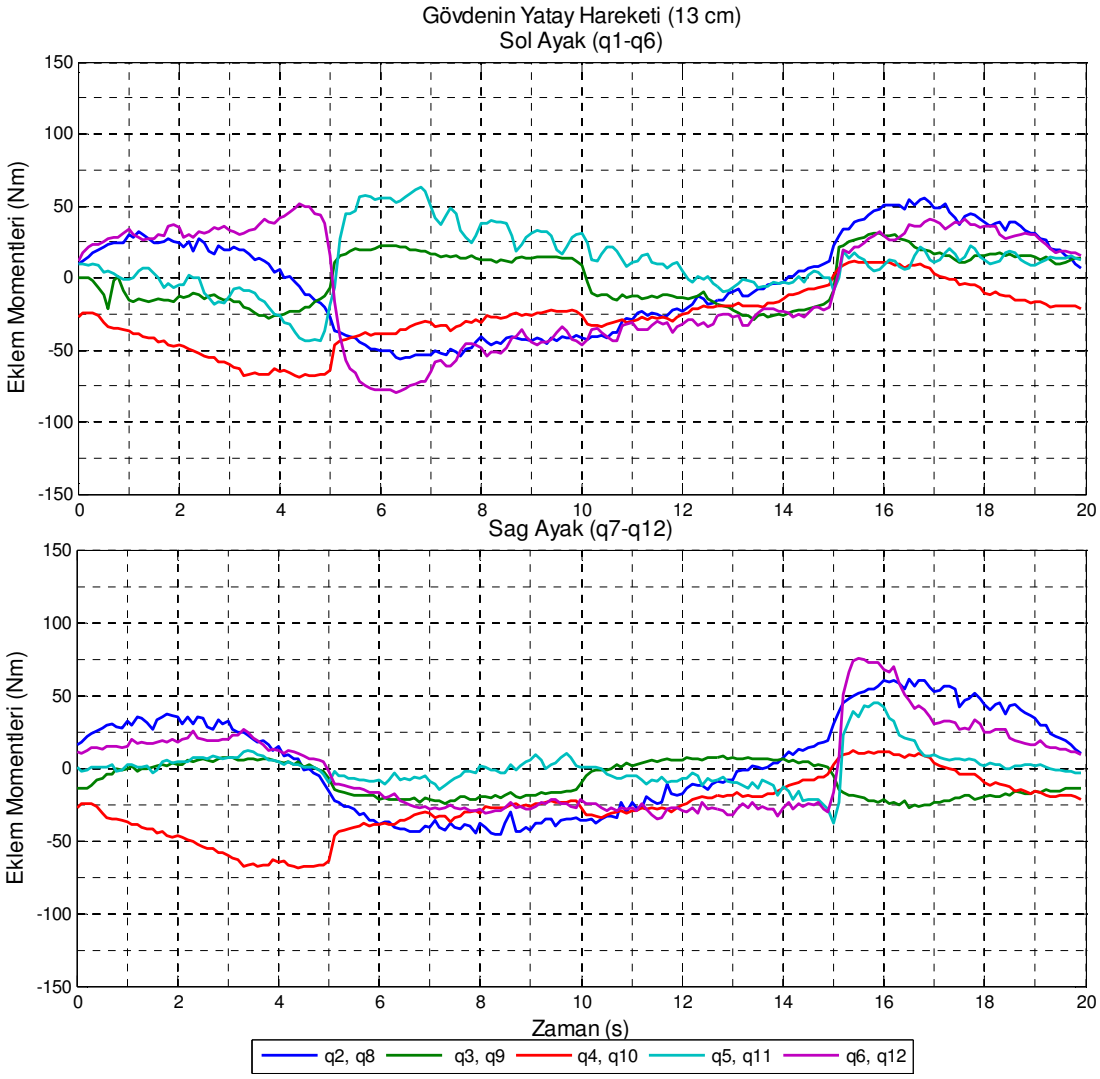
Deneyler sırasında robotun eklemlerinde ve motorlarında oluşan momentlerinin hesaplanması için motorların çektiği akımlardan yararlanılmıştır. Yazılım bölümünde anlatıldığı gibi sürücü kartları anlık akım bilgisini bir nesnesinde saklar.Motorların deneyler sırasında çektiği akımlar bu nesneden okunmuş ve bir dosyaya yazılmıştır. Daha sonra bu dosya bir masaüstü bilgisayara aktarılmış ve akım verileri MATLAB'ta motor momentlerine çevrilmiştir. Ölçülen akımlar motorların tork katsayısılarına bölünerek deneyler sırasında motor millerinde oluşan momentler bulunmuştur. Bulunan bu motor momentleri Şekil 4.7'te verilmiştir.



Şekil 4.7 : Gövdenin 13 cm'lik yatay hareketinde ölçülen motor momentleri.

Hareket sırasında Şekil 4.7'de görüldüğü gibi bilek motorlarında tahmin edilenden fazla zorlanma meydana gelmiştir. Dikkatli bakıldığında Θ_6'ın deneyin 6. ve 7. saniyeleri arasında beslenebileceği maksimum akım değerine ulaşarak doyuma ulaştığı görülmektedir. Doyum süresince motor gerekli momenti karşılayamadığından dolayı referans motor yörüngesi bu motor tarafından doğru olarak takip edilememiştir.

Robotun motor momentleri sürücü kartları tarafından ölçülen akımlardan bulunmuştur. Eklem momentlerini bulmak için ise robotun önceden oluşturulmuş kinematik denklemleri kullanılmıştır[21]. Bulunan motor momentleri robotun aktarım organları için çıkartılan kinematik denklemleri kullanılarak eklem momentlerine çevrilmiştir. Bu şekilde hesaplanan eklem momentleri Şekil 4.8'de verilmiştir.



Şekil 4.8 : Gövdenin 13 cm'lik yatay hareketindeki eklem momentleri.

Şekil 4.7'de verilen motor momentlerinde gözlemlenen doyum etkisi Şekil 4.8'de verilen eklem momentlerinde de etkisini göstermektedir. Bilek eklemlerinin momentlerinin tahmin edilenden bu kadar yüksek çıkması bilekteki aktarma organının yapısından kaynaklandığı görülmüştür. Bilek aktarma organlarında kullanılan bazı parçaların bükülerek imal edilmesi aynı bacaktaki bilek motorlarının tam paralel monte edilememesine sebep olmuştur. İki motorun tam paralel olmamasının motorlarda fazladan moment oluşturacağı ön görülmüştür. Ayrıca en büyük etken olarak da motor millerinin yataklanmamasından dolayı motorların eksenel yüke maruz kaldığı anlaşılmıştır. Bu sebeplerden dolayı bileklerdeki aktarma organlarının bağlantı parçalarının yeniden bükmeden imal edilmesine ve motor millerinin yataklanmasına karar verilmiştir. Bilek eklemlerindeki aktarma organları değişene kadar deneylere de ara verilmiştir.

Çizelge 4.1: Deneylerdeki maksimum referans eklem hızları

	Hesaplanan Maksimum Eklem Hızları (rad/s)				
	Gövdenin düşey hareketi		Gövdenin yatay hareketi		
Eklemler	5 cm	10 cm	7 cm	10 cm	13 cm
q_1	0	0	0	0	0
q_2	0	0	0.0276	0.0394	0.0512
q_3	0.0403	0.0725	0.0041	0.0084	0.0145
q_4	0.0834	0.1502	0.0084	0.0173	0.0299
q_5	0.0431	0.0777	0.0043	0.0089	0.0154
q_6	0	0	0.0275	0.0392	0.0508
q_7	0	0	0	0	0
q_8	0	0	0.0276	0.0394	0.0512
q_9	0.0403	0.0725	0.0041	0.0084	0.0145
q_{10}	0.0834	0.1502	0.0084	0.0173	0.0299
q_{11}	0.0431	0.0777	0.0043	0.0089	0.0154
q_{12}	0	0	0.0275	0.0392	0.0508

Çizelge 4.1'de robotun referans eklem yörüngelerinin ayrık türevi alınarak hesaplanan maksimum eklem hızları verilmiştir. Dinamik kuvvetlere maruz kalmamak için robotun tüm eklemlerinin düşük hızlarla hareket ettirildiği Çizelge 4.1'den de görülmektedir. Çizelge 4.2'de ise robotun referans motor yörüngelerinin ayrık türevi alınarak hasaplanan maksimum motor hızları verilmiştir. Görüldüğü gibi hareketlerin genliği arttıkça süre sabit kaldığı için motor hızları da artmaktadır. Ayrıca, gövdenin düşey hareketi boyunca diz motorlarının, gövdenin yatay hareketi boyunca da bilek motorlarının diğer motorlara göre daha hızlı hareket ettikleri görülmektedir.

Çizelge 4.2: Deneylerdeki maksimum referans motor hızları

	Hesaplanan Maksimum Motor Hızları (rad/s)				
	Gövdenin düşey hareketi		Gövdenin yatay hareketi		
Motorlar	5 cm	10 cm	7 cm	10 cm	13 cm
Θ_1	0	0	0	0	0
Θ_2	0	0	4.39	6.28	8.19
Θ_3	6.44	11.59	0.65	1.35	2.32
Θ_4	12.59	22.61	1.25	2.57	4.46
Θ_5	8.48	14.92	5.78	8.76	12.06
Θ_6	8.48	14.92	5.78	8.73	12.06
Θ_7	0	0	0	0	0
Θ_8	0	0	4.39	6.28	8.19
Θ_9	6.44	11.59	0.65	1.35	2.32
Θ_{10}	12.59	22.61	1.25	2.57	4.46
Θ_{11}	8.48	14.92	5.78	8.76	12.06
Θ_{12}	8.48	14.92	5.78	8.73	12.06

Çizelge 4.3: Deneylerdeki maksimum motor momentleri.

	Ölçülen Maksimum Motor Momentleri (Nm)				
	Gövdenin düşey hareketi		Gövdenin yatay hareketi		
Motorlar	5 cm	10 cm	7 cm	10 cm	13 cm
Θ_1	0.0011	0.0275	0.0342	0.0270	0.0341
Θ_2	0.0059	0.0786	0.2062	0.2810	0.3550
Θ_3	0.3242	0.4202	0.1499	0.1477	0.2022
Θ_4	0.4738	0.5950	0.3408	0.3801	0.4743
Θ_5	0.1520	0.3772	0.2276	0.1755	0.1742
Θ_6	0.2575	0.3778	0.3256	0.3561	0.3775
Θ_7	0.0168	0.0324	0.0147	0.0250	0.0294
Θ_8	0.0620	0.0511	0.2448	0.3177	0.3828
Θ_9	0.2774	0.3839	0.1439	0.1363	0.1730
Θ_{10}	0.5451	0.6090	0.5497	0.5370	0.5240
Θ_{11}	0.1120	0.3633	0.0975	0.1312	0.3616
Θ_{12}	0.1129	0.3859	0.0802	0.1056	0.1401

Çizelge 4.3'te motorlarda oluşan maksimum motor momentleri verilmiştir. Çizelgeden de anlaşıldığı gibi hareketin genliği artıkça momentler de yükselmektedir. Gövdenin 10 cm'lik düşey hareketi, 10 cm'lik ve 13 cm'lik yatay hareketleri sırasında bilek motorlarının momentleri kritik seviyelere ulaşmıştır. Çizelge 4.4'te ise bu motor momentlerinden ileri kinematik denklemler kullanılarak hesaplanan eklem momentleri verilmiştir.

Çizelge 4.4: Deneylerdeki maksimum eklem momentleri.

	Hesaplanan Maksimum Eklem Momentleri (Nm)				
	Gövdenin düşey hareketi		Gövdenin yatay hareketi		
Eklemler	5 cm	10 cm	7 cm	10 cm	13 cm
q_1	0.13	3.30	4.09	3.23	4.09
q_2	0.94	12.57	32.98	44.95	56.80
q_3	51.86	67.22	23.98	23.63	32.35
q_4	70.78	89.79	50.75	56.36	69.94
q_5	78.84	148.74	57.04	67.36	63.21
q_6	20.88	22.39	65.31	63.41	79.58
q_7	2.02	3.88	1.76	3.00	3.52
q_8	9.92	8.17	39.16	50.82	61.24
q_9	44.38	61.42	23.02	21.80	27.67
q_{10}	70.78	89.79	50.75	56.36	69.94
q_{11}	45.37	144.36	9.13	15.08	49.98
q_{12}	7.87	78.29	26.96	37.55	78.43

5. SONUÇ VE ÖNERİLER

12 serbestlik dereceli ITU Biped robotun, katı modeli bu tez çalışmasından önce oluşturulmuş ve benzetim ortamında robota istenen hareketler yaptırılmıştır. Bu tezde ise, benzetimlerdeki robotun prototipinde kullanılmak üzere elektronik donanımlarının seçimi, birbiriyle iletişimi, programlanması ve robota başlangıç konumu için eklem yörüngesi oluşturulması konuları ele alınmıştır. Robotun daha önceden benzetim ortamında gerçekleştirdiği yatay ve düşey doğrultuda gövdesinin hareketi ve yürümesi için planlanan eklem yörüngeleri, belirlenmiş örnekleme zamanında ayrıklaştırılarak gömülü bilgisayara yüklenmiş ve bu verilerle gerçek prototiple aynı hareketler denenmiştir.

Robot askıda ayakları yere temas etmediği zaman verilen tüm eklem referanslarını takip etmiştir. Ancak robot yere indirildiğinde bilek eklemlerindeki mekanik problemlerden dolayı yörünge gerçekleştirmede sorunlarla karşılaşılmıştır. Robotun ayakları yere temas ederken düşük genlikli hareketlerde başarı sağlanmıştır. Fakat yapılan bazı deneylerde ise bilek eklemlerinde oluşan momentlerin motorların karşılayabileceği moment değerlerini geçtiği görülmüştür. Robotun gövdesi düşeyde 10 cm ve yatayda 13 cm hareket ederken motorların doyuma ulaştığı görülmüş ve deneylere ara verilmiştir. Robotun bilek eklemlerindeki yataklama sorunu giderilince deneyler tekrar başlayacaktır.

KAYNAKLAR

[1] **Lee, C. H., Kim, S. H., Kang, S. C., Kim, M. S. and Kwak, Y. K.,** 2003: Double-track mobile robot for hazardous environment applications, Advanced Robotics, Vol. **17**, no. 5, pp. 447-459.

[2] **Vukobratovic, M. and Juricic, D.,** 1968: A contribution to the synthesis of biped gait, *IFAC Symp. Technical and Biological Problem of Control,* Yerevan, USSR.

[3] **Kato, I., Ohteru, S., Kobayashi, H., Shirai, K. and Uchiyama, A.,** 1973: Information-power machine with senses and limbs, *Proc. CISM-IFToMM Symp. Theory and Practice of Robots and Manipulators*, Udine, Italy, 12-24.

[4] **Ogura, Y., Aikawa, H., Shimomura, K., Morishima, A., Hun-ok L. and Takanishi, A.,** 2006. Development of a new humaid robot WABIAN-2, *In Proceedings of IEEE International Conference on Robotics and Automation,* Orlando, FL, USA, 76 – 81.

[5] **Sakagami, Y., Watanabe, R., Aoyama, C., Matsunaga, S., Higaki N. and Fujimura, K.,** 2002. The intelligent ASIMO: System overview and integration, Intelligent Robots and Systems, Vol. **3**, pp. 2478 – 2483.

[6] **American Honda Motor Co., Inc.,** 2011. ASIMO Photo Gallery. Retrieved May 2, 2011, from http://asimo.honda.com.

[7] **Advanced Industrial Science and Technology,** 2009. Successful Development of a Robot with Appearance and Performance Similar to Humans, Retrieved April 28, 2011, from http://www.aist.go.jp .

[8] **Pfeiffer, F., Loffler, K., Gienger, M. and Ulbrich, H.,** 2004. Sensor and control aspects of biped robot Johnnie, International Journal of Humanoid Robotics (IJHR), Vol. **1-3**, pp. 481–496.

[9] **Lohmeier, S., Buschmann, T. and Ulbrich, H.,** 2009. System Design and Control of Antropomorphic Walking Robot LOLA, Proc. IEEE/ASME Transactions on Mechatronics, Vol. **14**, no:6, pp. 658 – 666.

[10] **Kim, J. and Oh, J.** 2004. Realization of dynamic walking for the humanoid robot platform KHR-1, Advanced Robotics, Vol. **18**, no:7, pp. 749-768

[11] **Kim, J., Park, I., Lee, J., Kim, M., Cho, B. and Oh, J.,** 2005. System Design and Dynamic Walking of Humanoid Robot KHR-2, Robotics and Automation, pp. 1431 – 1436.

[12] **Park, I., Kim, J., Lee, J. and Oh, J.,** 2005. Mechanical design of humanoid robot platform KHR-3, *Humanoid Robots, 5th IEEE-RAS International Conference on*, Tsukuba, Japan, 321 – 326.

[13] **Oh J., Hanson, D., Kim W., Han, Y., Kim, J. and Park, I.,** 2006. Design of Android type Humanoid Robot Albert HUBO, *Intelligent Robots and Systems, IEEE/RSJ International Conference on,* Beijing, China, 1428 – 1433.

[14] **Gerçek, A., Acar, M., Kocabaş, H. ve Bayraktaroğlu, Z. Y.,** 2009. İki Ayaklı Yürüyen Robot Tasarımı*, TOK'09,* İstanbul, Turkey.

[15] **Acar, M., Gercek A., Tasasız, B., Bayraktaroglu, Z. Y., Kocabaş, H.,** 2010. Design of a 12-Dof Biped Robot, *MACRo'2010- International Conference on Recent Achievements in Mechatronics, Automation, Computer Science and Robotics*, Targu Mureş, Romania, 215 - 227.

[16] **Kajita, S., Kanehiro, F., Kaneko, K., Fujiwara, K., Harada, K., Yokoi, K. and Hirukawa, H.,** 2003. Biped walking pattern generation by using preview control of zero-moment point, *Proc. of IEEE International Conference on Robotics and Automation,* 2, 1620-1626.

[17] **Acar, M.,** 2007. İki ayaklı yürüme hareketinin modellenmesi ve kontrolü, Yüksek Lisans Tezi, İstanbul Teknik Üniversitesi Fen Bilimleri Enstitüsü, İstanbul.

[18] **Maxon Motor AG,** 2010. EPOS 2 Positioning Controllers Application Notes Collection, Retrieved April 28, 2011, http://www.maxonmotor.com/ .

[19] **Haddad, M., Chettibi, T., Khalil, W. and Lehtihet, H.,** 2007. Trajectory Generation. in Modeling, Performance Analysis and Control of Robot Manipulators, p. 193-198, Eds. Dombre, E. and Wisama Khalil W., Antony Rowe Ltd, Chippenham, Wiltshire.

[20] **Bayraktaroglu, Z. Y., Acar, M., Gerçek, A. and Tasasız, B.,** 2011. Design and Construction of I.T.U. Biped Robot. Manuscript submitted for publication.

[21] **106M340 Nolu TUBİTAK Projesi,** 2011. İki Ayaklı Yürüyen Robot Tasarımı, Prototip İmalatı ve Kontrolü.

EKLER

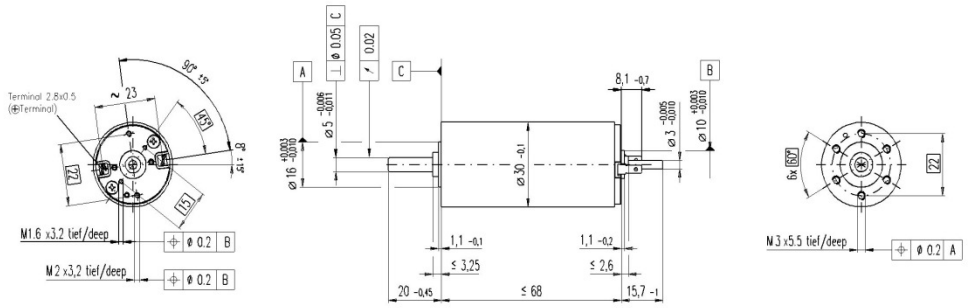
EK A.1 : RE-30 doğru akım motorunun veri kâğıdı.
EK A.2 : RE-40 doğru akım motorunun veri kâğıdı.
EK A.3 : RE-50 doğru akım motorunun veri kâğıdı.
EK A.4 : Kullanılan sürücü kartlarının veri kâğıdı.
EK A.5 : Kullanılan endüstriyel bilgisayarın veri kâğıdı.
EK A.6 : Kullanılan mutlak kodlayıcının veri kâğıdı.
EK A.7 : K/M algılayıcısının veri kâğıdı.
EK A.8 : Sürücü kartlarının içinde bulunan nesne sözlüğü.
EK B.1 : Eklem açılarını motor açılarına dönüştüren program.
EK B.2 : Can4linux.h kütüphanesi.
EK B.3 : Motorların hareketini sağlayan program.
EK B.4 : Başlangıç konumuna gitmek için eklem yörüngesi oluşturan program.
EK C.1 : Gövdenin 5 cm'lik düşey hareketi için referans eklem yörüngeleri.
EK C.2 : Gövdenin 5 cm'lik düşey hareketi için referans motor yörüngeleri.
EK C.3 : Gövdenin 5 cm'lik düşey hareketi için referans eklem hızları.
EK C.4 : Gövdenin 5 cm'lik düşey hareketi için referans motor hızları.
EK C.5 : Gövdenin 5 cm'lik düşey hareketinde ölçülen motor momentleri.
EK C.6 : Gövdenin 5 cm'lik düşey hareketindeki eklem momentleri.
EK C.7 : Gövdenin 10 cm'lik düşey hareketi için referans eklem yörüngeleri.
EK C.8 : Gövdenin 10 cm'lik düşey hareketi için referans motor yörüngeleri.
EK C.9 : Gövdenin 10 cm'lik düşey hareketi için referans eklem hızları.
EK C.10 : Gövdenin 10 cm'lik düşey hareketi için referans motor hızları.
EK C.11 : Gövdenin 10 cm'lik düşey hareketinde ölçülen motor momentleri.
EK C.12 : Gövdenin 10 cm'lik düşey hareketindeki eklem momentleri.
EK C.13 : Gövdenin 7 cm'lik yatay hareketi için referans eklem yörüngeleri.
EK C.14 : Gövdenin 7 cm'lik yatay hareketi için referans motor yörüngeleri.
EK C.15 : Gövdenin 7 cm'lik yatay hareketi için referans eklem hızları.
EK C.16 : Gövdenin 7 cm'lik yatay hareketi için referans motor hızları.
EK C.17 : Gövdenin 7 cm'lik yatay hareketinde ölçülen motor momentleri.
EK C.18 : Gövdenin 7 cm'lik yatay hareketindeki eklem momentleri.
EK C.19 : Gövdenin 10 cm'lik yatay hareketi için referans eklem yörüngeleri.
EK C.20 : Gövdenin 10 cm'lik yatay hareketi için referans motor yörüngeleri.
EK C.21 : Gövdenin 10 cm'lik yatay hareketi için referans eklem hızları.
EK C.22 : Gövdenin 10 cm'lik yatay hareketi için referans motor hızları.
EK C.23 : Gövdenin 10 cm'lik yatay hareketinde ölçülen motor momentleri.
EK C.24 : Gövdenin 10 cm'lik yatay hareketindeki eklem momentleri.

EK A.1

Website: http://www.allimagetool.com

maxon DC motor

RE 30 ∅30 mm, Graphite Brushes, 60 Watt



Stock program
Standard program
Special program (on request)

			Order Number								
		according to dimensional drawing	310005	310006	310007	310008	310009				
		shaft length 15.7 shortened to 8.7 mm	268193	268213	268214	268215	268216				
Motor Data											
	Values at nominal voltage										
1	Nominal voltage	V	12.0	18.0	24.0	36.0	48.0				
2	No load speed	rpm	8170	8590	8810	8590	8490				
3	No load current	mA	300	212	164	106	78.5				
4	Nominal speed	rpm	7630	7900	8050	7810	7750				
5	Nominal torque (max. continuous torque)	mNm	51.7	75.5	85.0	83.4	88.2				
6	Nominal current (max. continuous current)	A	4.00	4.00	3.44	2.20	1.72				
7	Stall torque	mNm	844	991	1020	936	1020				
8	Starting current	A	60.5	49.8	39.3	23.5	19.0				
9	Max. efficiency	%	86	87	87	87	88				
	Characteristics										
10	Terminal resistance	Ω	0.198	0.362	0.611	1.53	2.52				
11	Terminal inductance	mH	0.0345	0.0703	0.119	0.281	0.513				
12	Torque constant	mNm / A	13.9	19.9	25.9	39.8	53.8				
13	Speed constant	rpm / V	685	479	369	240	178				
14	Speed / torque gradient	rpm / mNm	9.74	8.71	8.69	9.22	8.33				
15	Mechanical time constant	ms	3.42	3.25	3.03	3.17	3.01				
16	Rotor inertia	gcm²	33.5	35.7	33.3	32.9	34.5				

Specifications

Thermal data

17	Thermal resistance housing-ambient	6.0 K / W
18	Thermal resistance winding-housing	1.7 K / W
19	Thermal time constant winding	16.2 s
20	Thermal time constant motor	714 s
21	Ambient temperature	-30 ... +100°C
22	Max. permissible winding temperature	+125°C

Mechanical data (ball bearings)

23	Max. permissible speed	12000 rpm
24	Axial play	0.05 - 0.15 mm
25	Radial play	0.025 mm
26	Max. axial load (dynamic)	5.6 N
27	Max. force for press fits (static)	110 N
	(static, shaft supported)	1200 N
28	Max. radial loading, 5 mm from flange	28 N

Other specifications

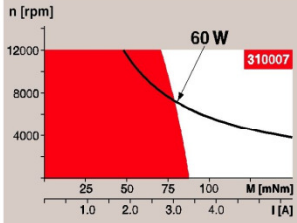
29	Number of pole pairs	1
30	Number of commutator segments	13
31	Weight of motor	238 g

Values listed in the table are nominal.
Explanation of the figures on page 49.

⚠ Tolerances may vary from the standard specification.

Option
Preloaded ball bearings

Operating Range



Comments

Continuous operation
In observation of above listed thermal resistance (lines 17 and 18) the maximum permissible winding temperature will be reached during continuous operation at 25°C ambient.
= Thermal limit.

Short term operation
The motor may be briefly overloaded (recurring).

Assigned power rating

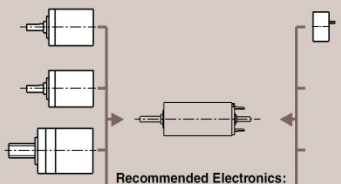
maxon Modular System

Overview on page 16 - 21

Planetary Gearhead
∅32 mm
0.75 - 6.0 Nm
Page 232 / 234 / 235

Koaxdrive
∅32 mm
1.0 - 4.5 Nm
Page 238

Spindle Drive
∅32 mm
Page 251 / 252 / 253



Encoder MR
256 - 1024 CPT
3 channels
Page 265

Recommended Electronics:

ADS 50/5	Page 282
ADS_E 50/5	283
EPOS2 Module 36/2	302
EPOS2 24/5	303
EPOS2 50/5	303
EPOS P 24/5	306
Notes	**18**

May 2009 edition / subject to change

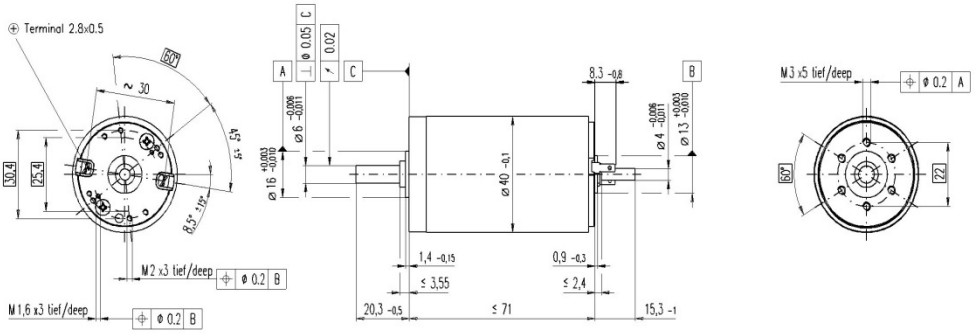
Şekil A.1 : RE-30 doğru akım motorunun veri kâğıdı.

EK A.2

Website: http://www.allimagetool.com

maxon DC motor

RE 40 ∅40 mm, Graphite Brushes, 150 Watt



M 1:2

Stock program
Standard program
Special program (on request)

Order Number

			148866	148867	148877	218008	218009	218010	218011	218012	218013	218014
Motor Data												
	Values at nominal voltage											
1	Nominal voltage	V	12.0	24.0	48.0	48.0	48.0	48.0	48.0	48.0	48.0	48.0
2	No load speed	rpm	6920	7580	7580	6420	5560	3330	2690	2130	1710	1420
3	No load current	mA	241	137	68.6	53.7	43.7	21.9	16.7	12.5	9.67	7.77
4	Nominal speed	rpm	6370	6930	7000	5810	4920	2700	2050	1500	1080	774
5	Nominal torque (max. continuous torque)	mNm	94.9	170	184	183	177	187	187	189	189	188
6	Nominal current (max. continuous current)	A	6.00	5.77	3.12	2.62	2.20	1.38	1.12	0.898	0.721	0.593
7	Stall torque	mNm	1680	2280	2500	1990	1580	995	796	641	512	415
8	Starting current	A	102	75.7	41.4	28.0	19.2	7.26	4.68	3.00	1.92	1.29
9	Max. efficiency	%	88	91	92	91	91	89	88	87	86	85
	Characteristics											
10	Terminal resistance	Ω	0.117	0.317	1.16	1.72	2.50	6.61	10.2	16.0	24.9	37.1
11	Terminal inductance	mH	0.0245	0.0823	0.329	0.460	0.612	1.70	2.62	4.14	6.40	9.31
12	Torque constant	mNm / A	16.4	30.2	60.3	71.3	82.2	137	170	214	266	321
13	Speed constant	rpm / V	581	317	158	134	116	69.7	56.2	44.7	35.9	29.8
14	Speed / torque gradient	rpm / mNm	4.15	3.33	3.04	3.23	3.53	3.36	3.39	3.35	3.37	3.44
15	Mechanical time constant	ms	6.03	4.81	4.39	4.36	4.35	4.31	4.31	4.31	4.31	4.32
16	Rotor inertia	gcm²	139	138	138	129	118	123	121	123	122	120

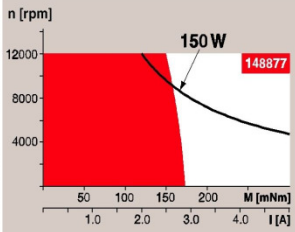
Specifications

	Thermal data	
17	Thermal resistance housing-ambient	4.65 K / W
18	Thermal resistance winding-housing	1.93 K / W
19	Thermal time constant winding	41.6 s
20	Thermal time constant motor	1120 s
21	Ambient temperature	-30 ... +100°C
22	Max. permissible winding temperature	+155°C
	Mechanical data (ball bearings)	
23	Max. permissible speed	12000 rpm
24	Axial play	0.05 - 0.15 mm
25	Radial play	0.025 mm
26	Max. axial load (dynamic)	5.6 N
27	Max. force for press fits (static)	110 N
	(static, shaft supported)	1200 N
28	Max. radial loading, 5 mm from flange	28 N
	Other specifications	
29	Number of pole pairs	1
30	Number of commutator segments	13
31	Weight of motor	480 g

Values listed in the table are nominal.
Explanation of the figures on page 49.

Option
Preloaded ball bearings

Operating Range



Comments

Continuous operation
In observation of above listed thermal resistance (lines 17 and 18) the maximum permissible winding temperature will be reached during continuous operation at 25°C ambient.
= Thermal limit.

Short term operation
The motor may be briefly overloaded (recurring).

Assigned power rating

maxon Modular System Overview on page 16 - 21

Planetary Gearhead
∅42 mm
3 - 15 Nm
Page 240

Planetary Gearhead
∅52 mm
4 - 30 Nm
Page 243

Encoder MR
256 - 1024 CPT,
3 channels
Page 265

Encoder HED_ 5540
500 CPT,
3 channels
Page 268 / 270

Brake AB 28
∅45 mm
24 VDC, 0.4 Nm
Page 316

Industrial Version
Encoder HEDL 9140
Page 273
Brake AB 28
Page 317

Recommended Electronics:

ADS 50/5	Page 282
ADS 50/10	283
ADS_E 50/5	283
ADS_E 50/10	283
EPOS2 24/5	303
EPOS2 50/5	303
EPOS 70/10	303
EPOS P 24/5	306
Notes	**18**

May 2009 edition / subject to change

Şekil A.2 : RE-40 doğru akım motorunun veri kâğıdı.

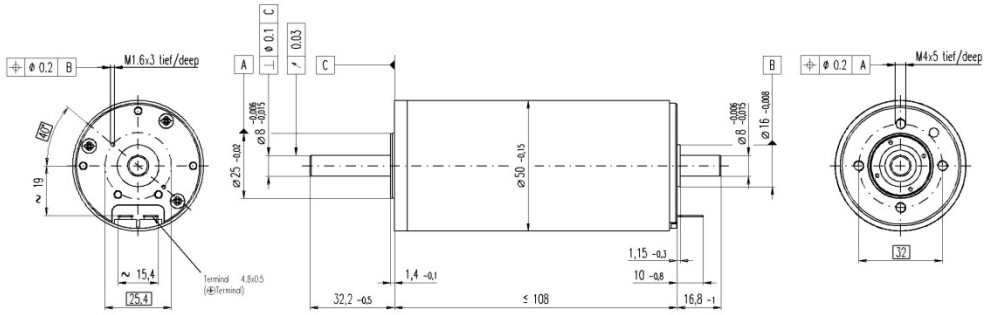
EK A.3

Website: http://www.allimagetool.com

RE 50 ∅50 mm, Graphite Brushes, 200 Watt

NEW

maxon DC motor



M 1:2

Stock program
Standard program
Special program (on request)

Order Number	370354	370355	370356	370357
Motor Data (provisional)				
Values at nominal voltage				
1 Nominal voltage V	24	36	48	70
2 No load speed rpm	5778	5778	4865	2806
3 No load current mA	248	166	100	35
4 Nominal speed rpm	5538	5538	4615	2545
5 Nominal torque (max. continuous torque) mNm	349	349	356	380
6 Nominal current (max. continuous current) A	9.04	6.03	3.88	1.63
7 Stall torque mNm	8416	8416	6956	4091
8 Starting current A	212	142	74	17
9 Max. efficiency %	93	93	93	91
Characteristics				
10 Terminal resistance Ω	0.113	0.254	0.649	4.068
11 Terminal inductance mH	0.094	0.211	0.528	3.371
12 Torque constant mNm / A	39.6	59.4	94.1	238
13 Speed constant rpm / V	241	161	102	40.2
14 Speed / torque gradient rpm / mNm	0.687	0.687	0.700	0.687
15 Mechanical time constant ms	4.20	4.20	4.24	4.20
16 Rotor inertia gcm²	584	584	578	584

Specifications

Thermal data

17	Thermal resistance housing-ambient	3.81 K / W
18	Thermal resistance winding-housing	2.27 K / W
19	Thermal time constant winding	137 s
20	Thermal time constant motor	406 s
21	Ambient temperature	-30 ... +100°C
22	Max. permissible winding temperature	+125°C

Mechanical data (preloaded ball bearings)

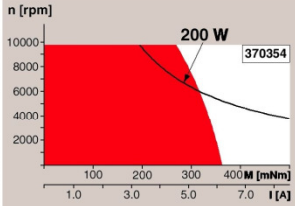
23	Max. permissible speed	9800 rpm
24	Axial play at axial load < 12 N	0 mm
	> 12 N	0.1 mm
25	Radial play	0.03 mm
26	Max. axial load (dynamic)	30 N
27	Max. force for press fits (static)	150 N
	(static, shaft supported)	6000 N
28	Max. radial loading, 15 mm from flange	110 N

Other specifications

29	Number of pole pairs	1
30	Number of commutator segments	15
31	Weight of motor	1100 g

Values listed in the table are nominal.
Explanation of the figures on page 49.

Operating Range



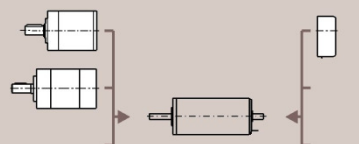
Comments

Continuous operation
In observation of above listed thermal resistance (lines 17 and 18) the maximum permissible winding temperature will be reached during continuous operation at 25°C ambient.
= Thermal limit.

Short term operation
The motor may be briefly overloaded (recurring).

Assigned power rating

maxon Modular System Overview on page 16 - 21



Planetary Gearhead
∅52 mm
4 - 30 Nm
Page 243

Planetary Gearhead
∅62 mm
8 - 50 Nm
Page 245

Encoder HED_ 5540
500 CPT,
3 channels
Page 269 / 271

Recommended Electronics:

ADS 50/5	Page 282
ADS 50/10	283
ADS_E 50/5	283
ADS_E 50/10	283
EPOS2 50/5	303
EPOS 70/10	303
Notes	**18**

Şekil A.3 : RE-50 doğru akım motorunun veri kâğıdı.

EK A.4

EPOS2 24/5
Slave-Version

USB CANopen RS232 GUI



Matched with DC brush motors with encoder or brushless EC motors with Hall sensors or encoder up to 120 watts.

Characteristics
- DC and EC motors up to 120 watts
- Point to point control unit (1 axis)
- Interpolated position mode (PVT)
- Combination of several drives via CAN Bus
- CANopen
- 6 digital inputs (TTL and PLC level)
- 4 digital outputs
- 2 analogue inputs (12-bit ADC)
- Compact module design

Slave version (online commanded) using CAN Master (EPOS2 P, PC, PLC, SoftPLC, etc.) or PC via USB or RS232 interface

Typical applications
- Tool building
- Production equipment
- System automation tasks

Electrical Data	EPOS2 24/5
Operating voltage V_{cc}	11–24 VDC
Max. output voltage	0.9 x V_{cc}
Max. output current I_{max}	10 A
Continuous output current I_{cont}	5 A
Mechanical Data	
Weight	approx. 170 g
Dimensions (L x B x H)	105 x 83 x 24 mm
Mounting	Flange for M3-screws
Order Number	
367676 EPOS2 24/5	
Accessories	
309687 DSR 50/5 Shunt regulator	

EPOS2 50/5, EPOS2 70/10
Slave-Version

USB CANopen RS232 GUI



Matched with DC brush motors with encoder or brushless EC motors with Hall sensors or encoder up to 700 watts.

Characteristics
- DC and EC motors up to 700 watts
- Point to point control unit (1 axis)
- Interpolated position mode (PVT)
- Combination of several drives via CAN Bus
- CANopen
- EPOS2 50/5 — 11 digital inputs (optically isolated); 5 digital outputs and 1 analogue output
- EPOS2 70/10 — 7 digital inputs (optically isolated); 3 digital inputs (differential); 3 digital outputs (optically isolated); 1 digital output (differential); 1 digital output
- 2 analogue inputs (12-bit ADC, differential)
- Compact module design

Slave version (online commanded) using CAN Master (EPOS2 P, PC, PLC, SoftPLC,etc.) or PC via USB or RS232 interface

Typical applications
- Tool building
- Production equipment
- System automation tasks
- Plant construction

Electrical Data	EPOS2 50/5	EPOS2 70/10
Operating voltage V_{cc}	11–50 VDC	11–70 VDC
Max. output voltage	0.9 x V_{cc}	0.9 x V_{cc}
Max. output current I_{max}	10 A	25 A
Continuous output current I_{cont}	5 A	10 A
Mechanical Data		
Weight	approx. 240 g	approx. 330 g
Dimensions (LxBxH) in mm	120x93.5x27	150x93x27
Mounting	Flange for M3-screws	
Order Numbers		
347717 EPOS2 50/5		
375711 EPOS2 70/10		
Accessories		
309687 DSR 50/5 Shunt regulator to EPOS2 50/5		
235811 DSR 70/30 Shunt regulator to EPOS2 70/10		

Şekil A.4 : Kullanılan sürücü kartlarının veri kâğıdı.

EK A.5

Website: http://www.allimagetool.com

UNO-2052E

AMD GX2 UNO with 2 x CAN, LAN



Features

- Onboard GX2 400 MHz
- Provides two CAN interfaces
- Provides one 10/100Base-T RJ-45 port and one USB port
- Windows® CE 5.0 & 6.0, Windows XP Embedded SP2, and Linux ready solution
- Fanless and no internal cabling design

Introduction

The Advantech UNO-2052E is a X86-grade platform that offers dual CAN 2.0B interfaces. Combined with CAN 2.0B interfaces, the UNO-2052E is an ideal solution for automobile and logistics applications. UNO-2052E is available with a pre-configured Windows CE image with optimized onboard device drivers. Microsoft Windows CE is a compact, highly efficient, real-time operating system designed for embedded systems without mechanical HDD limitations.

To expand storage capability, the UNO-2052E allows the addition of an external 2.5" HDD using Advantech's UNO HDD expansion kit. It can be used for large data backup requirements and popular OS installations such as Microsoft Windows and Linux OS. Significant anti-vibration is maintained even with the mechanical HDD inside. (1 G)

UNO-2052E is the perfect embedded application-ready-platform to shorten development time and offer a rich networking interface to fulfill diverse application requirements.

Specifications

General

Dimensions (W x D x H)	188.8 x 106.5 x 35.5 mm (7.5" x 4.2" x 1.4")
Enclosure	Aluminum
Mounting	Wall, DIN 35 rail
Power Consumption	15 W (Typical)
Power Input	9 ~ 36 V_{DC} (e.g +24 V @ 1 A) (Min. 24 W), AT
Weight	0.8 kg
OS Support	Windows XP Embedded, Windows 2000 & XP, Windows CE 5.0 & 6.0, Linux

System Hardware

CPU	AMD GX2 400 MHz
Memory	Onboard 256 MB DDR SDRAM
Indicators	Power LED, IDE LED, one programmable diagnostic LED and buzzer
Keyboard/Mouse	1 x PS/2
Storage	SSD: 1 x internal type I/II CompactFlash® slot HDD: expansion kit for one standard 2.5" HDD (Option)
VGA	DB15 connector
Watchdog Timer	Programmable

Communications

CAN	2 x isolated CAN 2.0B interfaces CAN controller: SJA-1000 CAN transceiver: 82C250
LAN	1 x 10/100Base-T with RJ-45 port
Serial Ports	1 x standard RS-232
Serial Port Speed	RS-232: 50 ~ 115.2 kbps
USB Ports	1 x USB port, OpenHCI, Rev. 1.1 compliant

Environment

Humidity	95 % @ 40° C (non-condensing)
Ingress Protection	IP40
Operating Temperature	-10 ~ 55° (14 ~ 131° F)
Shock Protection	IEC 68 2-27 CompactFlash®: 20 G @ DIN, half sine, 11 ms, 50 G @ Wall/Panel, half sine, 11 ms
Vibration Protection	IEC 68 2-6 CompactFlash®: 2 Grms @ sine, 5 ~ 500 Hz, 1 Oct./min, 1hr/axis. HDD: 1 Grms @ sine, 12 ~ 300 Hz, 1 Oct./min, 1hr/axis.

Ordering Information

UNO-2052E-IDA0E	GX2 400 M, 256 MB RAM UNO-2052E

Accessories

UNO-HD20-AE	UNO-2000 HDD expansion kit
UNO-FPM21-AE	UNO & FPM integration kit
UNO-ADAM42-AE	UNO-2000 & ADAM integration kit

Şekil A.5 : Kullanılan endüstriyel bilgisayarın veri kâğıdı.

EK A.6

Website: http://www.allimagetool.com

SCANCON

TYPE SCH24AB



CHARACTERISTICS

ENCODER TYPE	Hollow shaft absolute singleturn encoder
TECHNOLOGY	Magnetic Resistance
SMD - TECHNOLOGY	Strong compact electronics
IP-RATING	Std. IP 64 (option: Ribbon cable + IDC-Connector = IP 50)
LOW CURRENT CONSUMPTION	To be connected directly to PLCs
SUPPLY RANGE	5V ± 10% or 3,3V ± 10%
STRONG CONSTRUCTION	Based on 2 precision ball bearings, for industrial environments

ELECTRICAL SPECIFICATIONS

	At 25°C		
Output	SSI		
Supply-voltage (V+)	5V ± 10% or 3,3V ± 10%		
Current (no load)	max. 25mA		
Max. Load pr. output	4mA @ 5V (short circuit protection)		
Operating temp.	-40°C … +100°C		
Storage temp.	-55°C … +100°C		
Humidity non-cond.	5% … 85%		
Cable data	5 leads (0,14mm²), shielded		
Resolution	10 bit	12 bit ⑤ (slow mode)	12 bit ⑤ (fast mode)
electr. permissable speed (see note ②!)			
@ 512 pulses/rev.	1172 rpm	306 rpm	1121 rpm
@ 1024 pulses/rev.	586 rpm	153 rpm	610 rpm
@ 4096 pulses/rev.	---	38 rpm	153 rpm
System propagation delay (calculation time)	65 µs	384 µs	96 µs
Transition noise	0,12° rms	0,03° rms	0,06° rms
Precision	0,5°±LSB	0,5°±LSB	0,5°±LSB

MECHANICAL SPECIFICATIONS

Weight	approx. 75 g
Materials: Housing	Electroplated steel and brass
Hollow shaft	Stainless steel
Fixing Clamp	Brass
Bearings	Lifetime lubricated ball-bearings
H-shaft dimensions	ø2mm, ø3mm, ø4mm, ø5mm ø6mm, ø1/4"
H-shaft loads	axial max. 20N radial max. 20N
Mech. max. speed	10.000 rev./min (see notes ② and ③)
IP-rating	IP 64 (IP 50, with ribbon cable)
Start torque	<0,005 Nm at 25°C
Mass moment of inertia	<1,3 g·cm²
Max. shock	10 g / 11ms
Bump	10 g - 16ms (1000 x 3 axis)
Vibration	(10 - 2000 Hz) / 10 g

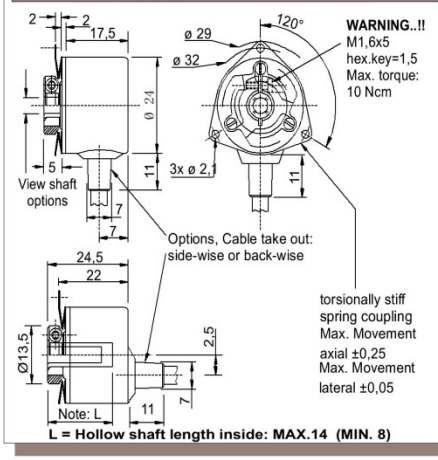
Notes:

1. The absolute angular position is always set to the highest resolution; regardless of rotational speed or the number of positions to be read out.
2. The absolute angular position is sampled at a fixed rate. This allows a maximum speed as shown in the electrical specifications without missing positions.
3. By increasing the rotational speed, the number of issued absolute angular positions will decrease.
4. Data D11:D0 resp. D9:D0 is valid when the status bits have the following configurations:

OCF	COF	LIN	MagINC	MagDEC	Parity
1	0	0	0	0	even
			0	1	checksum of
			1	0	bits 1 to 15

5. Make sure you give us information about the 12 bit speed mode in your order:
   "slow mode" - slow speed applications;
   "fast mode" - for applications with higher speeds.

MECHANICAL DIMENSIONS

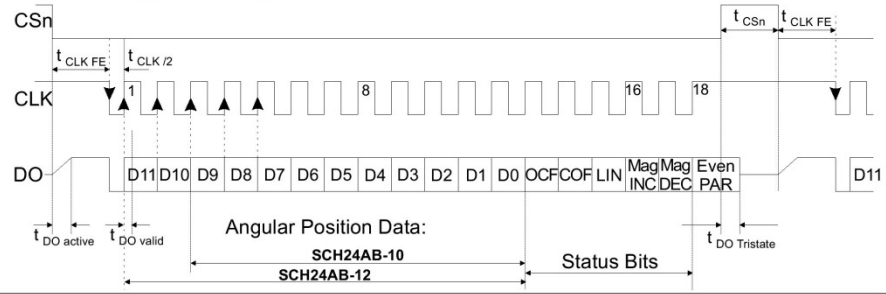


Section 14 - page 1

Şekil A.6 : Kullanılan mutlak kodlayıcının veri kâğıdı.

Website: http://www.allimagetool.com

SERIAL SSI-INTERFACE



SSI SYMBOL DECLARATION

D11 ... D0: absolute angular position data (D11: MSB @ 12bit res.; D9: MSB @10 bit res.; D0: LSB).
OCF (**O**ffset **C**ompensation **F**inished): logic high indicates the startup to be completed and data to be valid.
COF (**C**ordic **O**verflow): logic high indicates invalid data on bits D11:D0 resp. D9:D0. The absolute output maintains the last valid angular value.
LIN (**Lin**earity Alarm): logic high indicates critical output linearity on the input field. When bit is set, data bits may still be used but can contain invalid data.
MagINC (**Mag**nitude **Inc**rease): becomes high when magnetic field is increasing (pushing the magnet towards IC).
MagDEC (**Mag**nitude **Dec**rease): becomes high when magnetic field is decreasing (pulling the magnet away from IC).
Even PAR (**Even Par**ity): for transmission error detection of bits 1 to 15.

SSI-INTERFACE TIMETABLE

$t_{CLK\ FE}$ =	min. 500 ns	First data shifted to output register
$t_{CLK\ /2}$ =	min. 500 ns	Start of data output
$t_{DO\ active}$ =	max. 100 ns	Data output activated, logic high
$t_{DO\ valid}$ =	max. 375 ns	Data output valid
t_{CSn} =	min. 500 ns	Pulse width of CSn
$t_{DO\ Tristate}$ =	max. 100 ns	Data output tristate

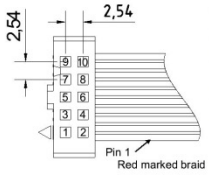
STANDARD CABLE CONNECTIONS

Color-code	Output
white	GND, 0 Volt
brown	V+
green	CSn
yellow	CLK
gray	DO

ORDERING CODES

Resolution / elec. specification:	10 bit - 3,3V	10A
	10 bit - 5,0V	10B
	12 bit - 3,3V - Fast mode	12E
	12 bit - 3,3V - Slow mode	12F
	12 bit - 5,0V - Fast mode	12G
	12 bit - 5,0V - Slow mode	12H
H.-shaft dimensions:	ø2mm	02
	ø3mm	03
	ø4mm	04
	ø5mm	05
	ø6mm	06
	ø1/4"	1/4
IP-rating:	IP 64	64
Length of cable:	Standard 1 meter	01
	No. of meters	XX
Cable take out:	Side	S
	Back	B
OPTION:		
Cable type:	Ribbon cable + IDC (HP-compatible)	IDC
Ribbon cable length:	0,5meter	0,5
	1,0meter	1,0
	2,0meter	2,0

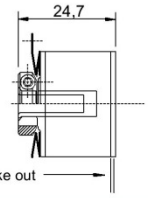
Option: IDC Connector with ribbon cable



Pin no.	Diff.
1 (Red)	Do not connect!
2	CSn
3	Do
4	CLK
5	V+
6	GND, 0 Volt
7	Do not connect!
8	Do not connect!
9	Do not connect!
10	Do not connect!

See Section 20 - Page 10 for ribbon cable outline dimensions.

Note:
Mechanical outline for Encoder with IDC connector and ribbon cable



To order the standard version: **SCH24AB** [ ][ ][ ] [ ][ ] 64 [ ][ ] [ ]
Resolution / electr. specification | H.-shaft | IP-rating | Cable length | Cable take out

- the version with ribbon cable: **SCH24AB** [ ][ ][ ] [ ][ ] 50 [ ][ ] IDC
Resolution / electr. specification | H.-shaft | IP-rating | Cable length | Cable type

TLF. +45 48 17 27 02 · FAX +45 48 17 22 84
info@scancon.dk www.scancon.dk

Eng. Rev 001
Feb - 2007

SCANCON

Şekil A.6 : (Devam) Kullanılan mutlak kodlayıcının veri kâğıdı.

EK A.7

MINI85

Product Advantages

Extremely High Strength:

- EDM wire-cut from high yield-strength stainless steel.
- Maximum allowable overload values are 10 to 13 times rated capacities.
- Larger 20mm through-hole available.

High Signal-to-Noise Ratio: Silicon strain gages provide a signal 75 times stronger than conventional foil gages. This signal is amplified, resulting in near-zero noise distortion.

Typical Applications

- Telerobotics
- Humanoid robots
- Prosthetics research
- Rehabilitation research



The Mini85 F/T transducer
The transducer is made of hardened stainless steel.

English Calibrations

SENSING RANGES / Axes	Calibrations US-55-110		US-110-220		US-220-440	
Fx, Fy (±lbf)	55		110		220	
Fz (±lbf)	110		220		440	
Tx, Ty (±lbf-in)	110		220		440	
Tz (±lbf-in)	110		220		440	
RESOLUTION / Axes	System Type* CTL	Net/DAQ	CTL	Net/DAQ	CTL	Net/DAQ
Fx, Fy (lbf)	7/320	7/640	7/160	7/320	7/80	7/160
Fz (lbf)	1/32	1/64	1/16	1/32	1/8	1/16
Tx, Ty (lbf-in)	3/80	3/160	3/40	3/80	3/20	3/40
Tz (lbf-in)	1/32	1/64	1/16	1/32	1/8	1/16

Metric Calibrations

SENSING RANGES / Axes	Calibrations SI-250-12.5		SI-500-25		SI-1000-50	
Fx, Fy (±N)	250		500		1000	
Fz (±N)	500		1000		2000	
Tx, Ty (±Nm)	12.5		25		50	
Tz (±Nm)	12.5		25		50	
RESOLUTION / Axes	System Type* CTL	Net/DAQ	CTL	Net/DAQ	CTL	Net/DAQ
Fx, Fy (N)	7/72	7/144	7/36	7/72	7/18	7/36
Fz (N)	5/36	5/72	5/18	5/36	5/9	5/18
Tx, Ty (Nm)	7/1800	7/3600	7/900	7/1800	7/450	7/900
Tz (Nm)	1/300	1/600	1/150	1/300	1/75	1/150

Refer to the transducer manual for complex loading information. System resolutions quoted are the effective resolution after dropping four counts of noise (Net/DAQ) or eight counts of noise (CTL). The effective resolution can be improved with filtering. Resolutions are typical. All sensors calibrated by ATI.
*CTL: Controller F/T System, Net: Net F/T System, DAQ: 16-bit DAQ F/T System

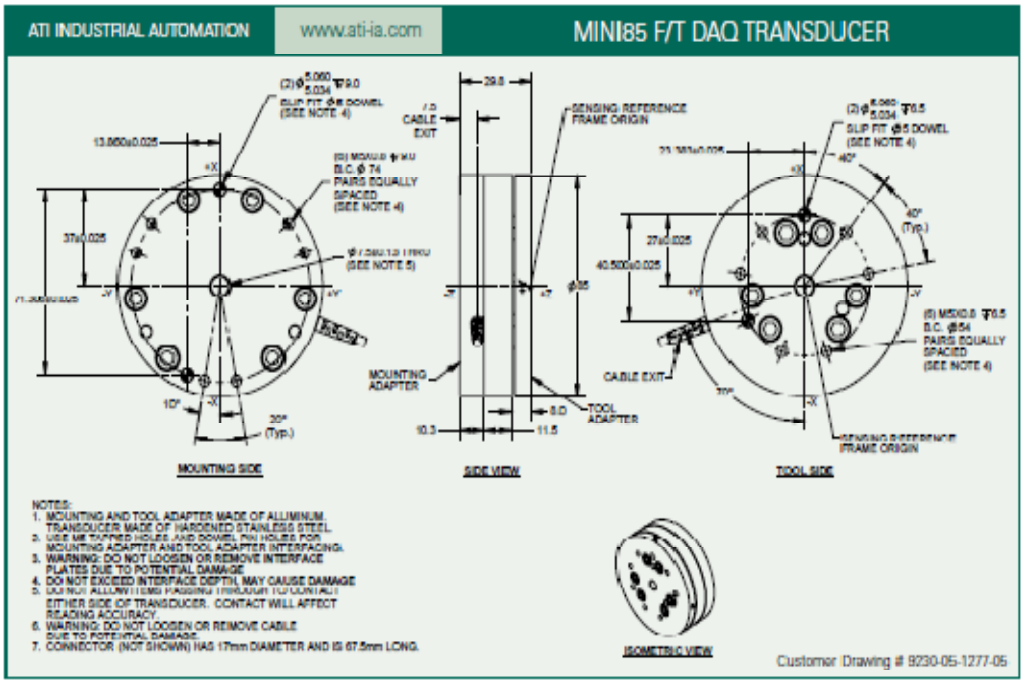
Şekil A.7 : K/M algılayıcısının veri kâğıdı.

Single-Axis Overload	English	Metric
Fxy	±2800 lbf	±13000 N
Fz	±6100 lbf	±27000 N
Txy	±4400 lbf-in	±500 Nm
Tz	±5400 lbf-in	±610 Nm
Stiffness (Calculated)	**English**	**Metric**
X-axis & Y-axis force (Kx, Ky)	4.4x10⁶ lb/in	7.7x10⁷ N/m
Z-axis force (Kz)	6.8x10⁶ lb/in	1.2x10⁹ N/m
X-axis & Y-axis torque (Ktx, Kty)	7.2x10⁶ lbf-in/rad	8.1x10⁴ Nm/rad
Z-axis torque (Ktz)	1.2x10⁶ lbf-in/rad	1.3x10⁴ Nm/rad
Resonant Frequency (Non-IP rated, Measured)		
Fx, Fy, Tz	2400 Hz	
Fz, Tx, Ty	3100 Hz	
Physical Specifications	**English**	**Metric**
Weight*	1.4 lb	0.635 kg
Diameter*	3.4 in	85 mm
Height*	1.2 in	30 mm

*Specifications include standard interface plates. Diameter excludes any connector block.

"ATI's products and knowledgeable staff were instrumental in allowing us to rapidly prototype our latest surgical robotic platform. Because of their off-the-shelf solutions for miniature force and torque sensing, we were able to go from concept to working prototype in months instead of years."

Patrick Jensen, Ph.D.
Assistant Professor of Ophthalmology
Johns Hopkins University



Şekil A.7 : (Devam) K/M algılayıcısının veri kâğıdı.

EK A.8

Çizelge A.8 : Sürücü kartlarının içinde bulunan nesne sözlüğü.

İndeks	Alt İndeks	Nesne İsmi	Erişim İzni
0x1000		Cihaz türü	Salt Okunur
0x1001		Hata Yazmacı	Salt Okunur
0x1003		Hata Geçmişi	Salt Okunur
0x1010		Kaydetme	Yazma/Okuma
0x1011		Varsayılan ayarlara dönüş	Yazma/Okuma
0x2000		Cihazın CANbus ID'si	Yazma/Okuma
0x2001		CanBus Veri Aktarım Hızı	Yazma/Okuma
0x2003		Versiyon	Salt Okunur
0x200A		CANbus Veri Aktarım Hızı	Salt Okunur
0x2027		Ortalamalı Güncel Akım Değeri	Salt Okunur
0x2028		Ortalamalı Güncel Hız Bilgisi	Salt Okunur
0x2030		Akım Modu Ayarları	Yazma/Okuma
0x2031		İstenen Akım Değeri	Salt Okunur
0x2062		Pozisyon Modu Ayarları	Yazma/Okuma
0x206B		Hız Modu Ayarları	Yazma/Okuma
0x2070		Dijital Girişlerin Konfigürasyonu	Yazma/Okuma
0x2074	0x01..0x04	Pozisyon İşaretleyici	Salt Okunur
0x2079	0x01, 0x02	Dijital Çıkışların Konfigürasyonu	Yazma/Okuma
0x207B	0x01, 0x02	Analog Girişlerin Konfigürasyonu	Yazma/Okuma
0x207C	0x01, 0x02	Analog Girişler	Salt Okunur
0x207E		Analog Çıkış 1	Yazma/Okuma
0x2081		Başlangıç Pozisyonu	Yazma/Okuma
0x2210	0x01..0x04	Algılayıcı Konfigürasyonu	Yazma/Okuma
0x2211	0x01..0x04	SSI Kodlayıcı Konfigürasyonu	Yazma/Okuma
0x2220		Kontrolcü yapısı	Yazma/Okuma
0x2230	0x01..0x03	Dişli Konfigürasyonu	Yazma/Okuma
0x6040		Kontrol Kelimesi	Yazma/Okuma
0x6041		Durum Kelimesi	Salt Okunur
0x6060		İşlem Modu	Yazma/Okuma
0x6061		İşlem Modu Görüntüleme	Salt Okunur
0x6062		İstenen Konum Değeri	Salt Okunur
0x6064		Güncel Konum Değeri	Salt Okunur
0x6065		Maksimum Takip Hatası	Yazma/Okuma
0x606B		İstenen Hız Değeri	Salt Okunur
0x606C		Güncel Hız Değeri	Salt Okunur
0x6078		Güncel Akım Değeri	Salt Okunur
0x607A		Hedeflenen Konum Değeri	Yazma/Okuma
0x6081		Hız Profili	Yazma/Okuma
0x6086		Hareket Profili Türü	Yazma/Okuma
0x60C5		Maksimum İvme Değeri	Yazma/Okuma
0x60F6	0x01, 0x02	Akım Kont. için Kontrol Parametreleri	Yazma/Okuma
0x60F9	0x01..0x05	Hız Kontrolü için Kontrol Parametreleri	Yazma/Okuma
0x60FB	0x01..0x05	Konum Kont. için Kontrol Parametreleri	Yazma/Okuma
0x60FF		Hedeflenen Hız Değeri	Yazma/Okuma
0x6402		Motor Tipi	Yazma/Okuma
0x6410	0x01..0x05	Motor Verileri	Yazma/Okuma

EK B.1

```
1 :  /*****************************************************
2 :                          -j2mtrans.c-
3 :  Bu program, İTÜ Biped Robot'un eklem referanslarını motor
4 :  referanslarına çevirir. Eklem referanslarını "jointAngles.bin"
5 :  dosyasından okur. Oluşturduğu motor referanslarını
6 :  "motorAngles.bin" dosyasına yazar.
7 :  3 Mart 2011 tarihinde Burak Tasasız ve Mesut Acar
8 :  tarafından oluşturulmuştur.
9 :  *****************************************************/
10 : #include <stdio.h>
11 : #include <stdlib.h>
12 : #include <math.h>
13 : #include <string.h>
14 : #include <sys/types.h>
15 : #include <sys/stat.h>
16 : #include <fcntl.h>
17 : #include <unistd.h>
18 : #include <time.h>
19 : #include "can4linux.h"
20 : #define STDDEV "can0"
21 : #define RXBUFFERSIZE 100
22 : int fd;
23 : canmsg_t tx;
24 : canmsg_t rx[RXBUFFERSIZE];
25 : long incDegrees[12];
26 : long shiftValues[12];
27 : double th_hzl, th_hxl, th_hyl, th_kl, th_ayl, th_axl, th_hzr, th_hxr, th_hyr, th_kr, th_ayr, th_axr;
28 : signed long int th_hzml, th_hxml, th_hyml, th_kml, th_amll, th_amrl, th_hzmr, th_hxmr, th_hymr,
       th_kmr, th_amlr, th_amrr;
29 : void initCan(void);
30 : void sendCan(int ID, char command, int index, int subindex,unsigned long object);
31 : void incEncodersRead(int ID);
32 : int main(int argc, char *argv[])
33 : {
34 : const double pi = 3.14159;
35 : const double r2c = 1000/pi;
36 : //Hip Parameters
37 : const int n_hzl = 120;
38 : const int n_hxl = 160;
39 : const int n_hyl = 160;
40 : const int n_hzr = 120;
41 : const int n_hxr = 160;
42 : const int n_hyr = 160;
43 : //Knee Parameters
44 : const double Sv_k = 0.002; //knee screw pitch
45 : const double a_k = 0.033;
46 : const double b_k = 0.035;
47 : const double l_k = 0.337;
48 : //Ankle Parameters
49 : const double l_a = 0.17226;
50 : const double Sv_a = 0.001;
51 : const double a_all = -0.036666;
52 : const double b_all = 0.031;
53 : const double c_all = -0.006067;
54 : const double a_arl = -0.036666;
55 : const double b_arl = -0.031;
56 : const double c_arl = -0.006067;
57 : const double x1_all = -0.039;
58 : const double y1_all = 0.0275;
59 : const double x1_arl = -0.039;
60 : const double y1_arl = -0.0275;
61 : const double a_alr = -0.036666;
62 : const double b_alr = 0.031;
63 : const double c_alr = -0.006067;
64 : const double a_arr = -0.036666;
65 : const double b_arr = -0.031;
```

```
66 :  const double c_arr = -0.006067;
67 :  const double x1_alr = -0.039;
68 :  const double y1_alr = 0.0275;
69 :  const double x1_arr = -0.039;
70 :  const double y1_arr = -0.0275;
71 :  double x0_all;
72 :  double y0_all;
73 :  double z0_all;
74 :  double z1_all;
75 :  double delta_all;
76 :  double z1i_all;
77 :  double x0_arl;
78 :  double y0_arl;
79 :  double z0_arl;
80 :  double z1_arl;
81 :  double delta_arl;
82 :  double z1i_arl;
83 :  double x0_alr;
84 :  double y0_alr;
85 :  double z0_alr;
86 :  double z1_alr;
87 :  double delta_alr;
88 :  double z1i_alr;
89 :  double x0_arr;
90 :  double y0_arr;
91 :  double z0_arr;
92 :  double z1_arr;
93 :  double delta_arr;
94 :  double z1i_arr;
95 :  FILE *fj;
96 :  FILE *fm;
97 :  int count = 0;
98 :  initCan();
99 :  incEncodersRead(1);
100 : incEncodersRead(2);
101 : incEncodersRead(3);
102 : incEncodersRead(4);
103 : incEncodersRead(5);
104 : incEncodersRead(6);
105 : incEncodersRead(7);
106 : incEncodersRead(8);
107 : incEncodersRead(9);
108 : incEncodersRead(10);
109 : incEncodersRead(11);
110 : incEncodersRead(12);
111 : printf("LHZ=%ld\tLHX=%ld\tLHY=%ld\tLK=%ld\tLAL=%ld\tLAR=%ld\n",incDegrees[0],incDegre
      es[1],incDegrees[2],incDegrees[3],incDegrees[4],incDegrees[5]);
112 : printf("RHZ=%ld\tRHX=%ld\tRHY=%ld\tRK=%ld\tRAL=%ld\tRAR=%ld\n",incDegrees[6],incDegr
      ees[7],incDegrees[8],incDegrees[9],incDegrees[10],incDegrees[11]);
113 : if ((fj = fopen("jointTrajectory.bin", "r")) == NULL) {
114 : printf("Cannot open jointAngles.bin file!\n");
115 : exit(EXIT_FAILURE);
116 : }
117 : if ((fm = fopen("motorTrajectory.bin", "w")) == NULL) {
118 : printf("Cannot open motorAngles.bin file!\n");
119 : exit(EXIT_FAILURE);
120 : }
121 : while (fscanf(fj, "%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf", &th_hzl, &th_hxl, &th_hyl, &th_kl,
      &th_ayl, &th_axl, &th_hzr, &th_hxr, &th_hyr, &th_kr, &th_ayr, &th_axr) != EOF) {
122 : //Hip Motors Analysis//
123 : th_hzml = (r2c) * th_hzl * n_hzl;
124 : th_hxml = (-r2c) * th_hxl * n_hxl;
125 : th_hyml = (-r2c) * th_hyl * n_hyl;
126 : th_hzmr = (r2c) * th_hzr * n_hzr;
127 : th_hxmr = (-r2c) * th_hxr * n_hxr;
128 : th_hymr = (r2c) * th_hyr * n_hyr;
129 : //Knee Motors Analysis//
```

```
130 : th_kml = (r2c) * 2 * pi / Sv_k * (sqrt(powf(b_k,2) + powf((a_k + l_k),2)) - sqrt(powf(a_k,2) +
      powf(b_k,2) + powf(l_k,2) - 2*sqrt(powf(a_k,2) + powf(b_k,2)) * l_k * cos(pi-atan(b_k/a_k) - th_kl)));
131 : th_kmr = (r2c) * 2 * pi / Sv_k * (sqrt(powf(b_k,2) + powf((a_k + l_k),2)) - sqrt(powf(a_k,2) +
      powf(b_k,2) + powf(l_k,2) - 2*sqrt(powf(a_k,2) + powf(b_k,2)) * l_k * cos(pi-atan(b_k/a_k) - th_kr)));
132 : //Ankle Motors Analysis//
133 : x0_all = a_all*cos(th_ayl) + b_all*sin(th_ayl)*sin(th_axl) + c_all*sin(th_ayl)*cos(th_axl);
134 : y0_all = b_all*cos(th_axl) - c_all*sin(th_axl);
135 : z0_all = -a_all*sin(th_ayl) + b_all*cos(th_ayl)*sin(th_axl) + c_all*cos(th_ayl)*cos(th_axl);
136 : z1_all = sqrt(powf(l_a,2) - powf(x1_all-x0_all,2) - powf(y1_all-y0_all,2)) + z0_all;
137 : if (count == 0)
138 : z1i_all = z1_all;
139 : delta_all = z1_all - z1i_all;
140 : th_amll = (r2c) * 2*pi/Sv_a*delta_all;
141 : x0_arl = a_arl*cos(th_ayl) + b_arl*sin(th_ayl)*sin(th_axl) + c_arl*sin(th_ayl)*cos(th_axl);
142 : y0_arl = b_arl*cos(th_axl) - c_arl*sin(th_axl);
143 : z0_arl = -a_arl*sin(th_ayl) + b_arl*cos(th_ayl)*sin(th_axl) + c_arl*cos(th_ayl)*cos(th_axl);
144 : z1_arl = sqrt(powf(l_a,2) - powf(x1_arl-x0_arl,2) - powf(y1_arl-y0_arl,2)) + z0_arl;
145 : if (count == 0)
146 : z1i_arl = z1_arl;
147 : delta_arl = z1_arl - z1i_arl;
148 : th_amrl = (r2c) * 2*pi/Sv_a*delta_arl;
149 : x0_alr = a_alr*cos(th_ayr) + b_alr*sin(th_ayr)*sin(th_axr) + c_alr*sin(th_ayr)*cos(th_axr);
150 : y0_alr = b_alr*cos(th_axr) - c_alr*sin(th_axr);
151 : z0_alr = -a_alr*sin(th_ayr) + b_alr*cos(th_ayr)*sin(th_axr) + c_alr*cos(th_ayr)*cos(th_axr);
152 : z1_alr = sqrt(powf(l_a,2) - powf(x1_alr-x0_alr,2) - powf(y1_alr-y0_alr,2)) + z0_alr;
153 : if (count == 0)
154 : z1i_alr = z1_alr;
155 : delta_alr = z1_alr - z1i_alr;
156 : th_amlr = (r2c) * 2*pi/Sv_a*delta_alr;
157 : x0_arr = a_arr*cos(th_ayr) + b_arr*sin(th_ayr)*sin(th_axr) + c_arr*sin(th_ayr)*cos(th_axr);
158 : y0_arr = b_arr*cos(th_axr) - c_arr*sin(th_axr);
159 : z0_arr = -a_arr*sin(th_ayr) + b_arr*cos(th_ayr)*sin(th_axr) + c_arr*cos(th_ayr)*cos(th_axr);
160 : z1_arr = sqrt(powf(l_a,2) - powf(x1_arr-x0_arr,2) - powf(y1_arr-y0_arr,2)) + z0_arr;
161 : if (count == 0)
162 : z1i_arr = z1_arr;
163 : delta_arr = z1_arr - z1i_arr;
164 : th_amrr = (r2c) * 2*pi/Sv_a*delta_arr;
165 : if(count==0)
166 : {
167 : shiftValues[0]= th_hzml;
168 : shiftValues[1]= th_hxml;
169 : shiftValues[2]= th_hyml;
170 : shiftValues[3]= th_kml;
171 : shiftValues[4]= th_amll;
172 : shiftValues[5]= th_amrl;
173 : shiftValues[6]= th_hzmr;
174 : shiftValues[7]= th_hxmr;
175 : shiftValues[8]= th_hymr;
176 : shiftValues[9]= th_kmr;
177 : shiftValues[10]= th_amlr;
178 : shiftValues[11]= th_amrr;
179 : }
180 : th_hzml= th_hzml + incDegrees[0] - shiftValues[0];
181 : th_hxml= th_hxml + incDegrees[1] - shiftValues[1];
182 : th_hyml= th_hyml + incDegrees[2] - shiftValues[2];
183 : th_kml=   th_kml + incDegrees[3] - shiftValues[3];
184 : th_amll= th_amll + incDegrees[4] - shiftValues[4];
185 : th_amrl= th_amrl + incDegrees[5] - shiftValues[5];
186 : th_hzmr= th_hzmr + incDegrees[6] - shiftValues[6];
187 : th_hxmr= th_hxmr + incDegrees[7] - shiftValues[7];
188 : th_hymr= th_hymr + incDegrees[8] - shiftValues[8];
189 : th_kmr=   th_kmr + incDegrees[9] - shiftValues[9];
190 : th_amlr= th_amlr + incDegrees[10] - shiftValues[10];
191 : th_amrr= th_amrr + incDegrees[11] - shiftValues[11];
192 : fprintf(fm, "%ld\t%ld\t%ld\t%ld\t%ld\t%ld\t%ld\t%ld\t%ld\t%ld\t%ld\t%ld\t%ld\n", th_hzml, th_hxml,
      th_hyml, th_kml, th_amll, th_amrl, th_hzmr, th_hxmr, th_hymr, th_kmr, th_amlr, th_amrr);
193 : count++;
```

```
//if (count == 10)
//break;
}
printf("Row Number = %d\n",count);
fclose(fj);
//    printf("jointAngles.bin closed!\n");
fclose(fm);
//    printf("motorAngles.bin closed!\n");
close(fd);
return 0;
}
/* incEncodersRead() FONKSIYONUNUN BAŞLANGICI */
/* Bu fonksiyon,  istenen artırımlı kodlayıcının değerinin bağlı olduğu sürücü kartının
ilgili nesnesinden okunabilmesi için yazılmıştır. */
void incEncodersRead(int ID)
{
int i;
double motorDegree;
unsigned long usigndata;
long motorCount;
//      printf("Reading Inc. Encoder...\n");
sendCan(ID, 'r', 0x6064, 0x00, 0x00);   // Read Position
usigndata= 65536*rx[0].data[6]+256*rx[0].data[5]+rx[0].data[4];
if(rx[0].data[6]>128)
motorCount =usigndata-16777216;
else
motorCount =usigndata;
incDegrees[ID-1]=motorCount;
}
/* sendCan() FONKSIYONUNUN BAŞLANGICI */
/* Bu fonksiyon Can mesajlarını Sürücü Kartlarının istediği formatta oluşturmak
için yazılmıştır. */
void sendCan(int ID, char command, int index, int subindex,unsigned long object)
{
int index_msb,index_lsb;
int value[3];
int sent,i,got;
tx.id=0x600+ID;
if(command == 'w')
tx.data[0]=0x22;
if(command == 'r')
tx.data[0]=0x40;
index_msb=index/256;
index_lsb=index-index_msb*256;
tx.data[1]=index_lsb;
tx.data[2]=index_msb;
tx.data[3]=subindex;
value[3]=object/16777216;
value[2]=(object-value[3]*16777216)/65536;
value[1]=(object-value[3]*16777216-value[2]*65536)/256;
value[0]=object-value[3]*16777216-value[2]*65536-value[1]*256;
tx.data[4]=value[0];
tx.data[5]=value[1];
tx.data[6]=value[2];
tx.data[7]=value[3];
tx.flags=0;
tx.length=8;
do{
sent = write(fd,&tx,1); //mesajin gonderilmesi
}while(sent<1);
got=0;
while(got<1)
{
got= read(fd,&rx,1);
}
}
/* initCan() FONKSIYONUNUN BAŞLANGICI */
```

```
261 : /* Bu fonksiyon  bilgisayarın Can modülünün çalıştırılabilmesi için gereken
262 : ayarlamalar için yazılmıştır. */
263 : void initCan(void)
264 : {
265 : char device[40];
266 : //        printf("Initializing Can...\n");
267 : sprintf(device, "/dev/%s", STDDEV);
268 : if(( fd = open(device, O_RDWR )) < 0 )
269 : {
270 : fprintf(stderr, "Error opening CAN device %s\n", device);
271 : exit(1);
272 : }
273 : }
```

EK B.2

```
1 :  /*****************************************************
2 :                              - can4linux.h -
3 :  Bu kütüphane, Linux işletim sistemi için hazırlanmış CANbus
4 :  sürücüsüdür. Bu kütüphanenin fonksiyonları kullanılarak CANbus
5 :  portunda bilgi gönderilir ve alınır.
6 :  Advantech firması tarafından yazılmıştır.
7 :  *****************************************************/
8 :  #ifndef __CAN_H
9 :  #define __CAN_H
10 : # define CAN4LINUXVERSION 0x0304 /*(Version 3.3)*/
11 : #ifndef __KERNEL__
12 : #include <sys/time.h>
13 : #endif
14 : /*---------- the can message structure */
15 : #define CAN_MSG_LENGTH 8      /**< maximum length of a CAN frame */
16 : #define MSG_RTR      (1<<0)     /**< RTR Message */
17 : #define MSG_OVR      (1<<1)     /**< CAN controller Msg overflow error */
18 : #define MSG_EXT      (1<<2)     /**< extended message format */
19 : #define MSG_SELF     (1<<3)     /**< message received from own tx */
20 : #define MSG_PASSIVE  (1<<4)     /**< controller in error passive */
21 : #define MSG_BUSOFF   (1<<5)     /**< controller Bus Off  */
22 : #define MSG_         (1<<6)     /**<  */
23 : #define MSG_BOVR     (1<<7)     /**< receive/transmit buffer overflow */
24 : /**
25 : mask used for detecting CAN errors in the canmsg_t flags field
26 : */
27 : #define MSG_ERR_MASK   (MSG_OVR + MSG_PASSIVE + MSG_BUSOFF + MSG_BOVR)
28 : /**
29 : The CAN message structure.
30 : Used for all data transfers between the application and the driver
31 : using read() or write().
32 : */
33 : typedef struct
34 : {
35 : /** flags, indicating or controlling special message properties */
36 : int           flags;
37 : int           cob;              /**< CAN object number, used in Full CAN  */
38 : unsigned   long id;                 /**< CAN message ID, 4 bytes  */
39 : struct timeval  timestamp;          /**< time stamp for received messages */
40 : short      int  length;           /**< number of bytes in the CAN message */
41 : unsigned   char data[CAN_MSG_LENGTH]; /**< data, 0...8 bytes */
42 : } canmsg_t;
43 : /**
44 : ---------- IOCTL requests */
45 : /* Use 'c' as magic number, follow chapter 6 of LDD3 */
46 : #define CAN4L_IOC_MAGIC 'c'
47 : #define CAN_IOCTL_COMMAND     0   /**< IOCTL command request */
48 : #define CAN_IOCTL_CONFIG      1   /**< IOCTL configuration request */
49 : #define CAN_IOCTL_SEND        2   /**< IOCTL request */
50 : #define CAN_IOCTL_RECEIVE     3   /**< IOCTL request */
51 : #define CAN_IOCTL_CONFIGURERTR     4   /**< IOCTL request */
52 : #define CAN_IOCTL_STATUS         5      /**< IOCTL status request */
53 : /*---------- CAN ioctl parameter types */
54 : /**
55 : IOCTL Command request parameter structure */
56 : struct Command_par {
57 : int cmd;              /**< special driver command */
58 : int target;           /**< special configuration target */
59 : unsigned long val1;    /**< 1. parameter for the target */
60 : unsigned long val2;      /**< 2. parameter for the target */
61 : int error;           /**< return value */
62 : unsigned long retval;  /**< return value */
63 : };
64 : /* IOCTL Command request parameter structure */
65 : typedef struct Command_par Command_par_t ; /**< Command parameter struct */
66 : /* IOCTL CConfiguration request parameter structure */
```

```
67 :  typedef struct Command_par  Config_par_t ; /**< Configuration parameter struct */
68 :  /* IOCTL generic CAN controller status request parameter structure */
69 :  typedef struct CanStatusPar {
70 :  unsigned int   baud;                /**< actual bit rate */
71 :  unsigned int   status;              /**< CAN controller status register */
72 :  unsigned int   error_warning_limit;   /**< the error warning limit */
73 :  unsigned int   rx_errors;           /**< content of RX error counter */
74 :  unsigned int   tx_errors;           /**< content of TX error counter */
75 :  unsigned int   error_code;          /**< content of error code register */
76 :  unsigned int   rx_buffer_size;      /**< size of rx buffer  */
77 :  unsigned int   rx_buffer_used;      /**< number of messages */
78 :  unsigned int   tx_buffer_size;      /**< size of tx buffer  */
79 :  unsigned int   tx_buffer_used;      /**< number of messages */
80 :  unsigned long  retval;              /**< return value */
81 :  unsigned int   type;               /**< CAN controller / driver type */
82 :  } CanStatusPar_t;
83 :  /* IOCTL  CanStatusPar.type CAN controller hardware chips */
84 :  #define CAN_TYPE_UNSPEC      0
85 :  #define CAN_TYPE_SJA1000     1
86 :  #define CAN_TYPE_FlexCAN     2
87 :  #define CAN_TYPE_TouCAN      3
88 :  #define CAN_TYPE_82527       4
89 :  #define CAN_TYPE_TwinCAN     5
90 :  #define CAN_TYPE_BlackFinCAN  6
91 :  /**
92 :  IOCTL Send request parameter structure */
93 :  typedef struct Send_par {
94 :  canmsg_t *Tx;          /**< CAN message struct  */
95 :  int error;           /**< return value for errno */
96 :  unsigned long retval;   /**< return value */
97 :  } Send_par_t ;
98 :  /**
99 :  IOCTL Receive request parameter structure */
100 : typedef struct Receive_par {
101 : canmsg_t *Rx;         /**< CAN message struct  */
102 : int error;           /**< return value for errno */
103 : unsigned long retval;  /**< return value */
104 : } Receive_par_t ;
105 : /**
106 : IOCTL ConfigureRTR request parameter structure */
107 : typedef struct ConfigureRTR_par {
108 : unsigned message;      /**< CAN message ID */
109 : canmsg_t *Tx;         /**< CAN message struct  */
110 : int error;           /**< return value for errno */
111 : unsigned long retval;  /**< return value */
112 : } ConfigureRTR_par_t ;
113 : /*---------- IOCTL Command subcommands and there targets */
114 : # define CMD_START        1
115 : # define CMD_STOP         2
116 : # define CMD_RESET        3
117 : # define CMD_CLEARBUFFERS  4
118 : /**
119 : ---------- IOCTL Configure targets */
120 : # define CONF_ACC              0  /* mask and code */
121 : # define CONF_ACCM             1  /* mask only */
122 : # define CONF_ACCC             2  /* code only */
123 : # define CONF_TIMING           3  /* bit timing */
124 : # define CONF_OMODE            4  /* output control register */
125 : # define CONF_FILTER           5
126 : # define CONF_FENABLE          6
127 : # define CONF_FDISABLE         7
128 : # define CONF_LISTEN_ONLY_MODE   8  /* for SJA1000 PeliCAN */
129 : # define CONF_SELF_RECEPTION     9
130 : # define CONF_BTR              10 /* set direct bit timing registers(SJA1000) */
131 : # define CONF_TIMESTAMP         11 /* use TS in received messages */
132 : # define CONF_WAKEUP           12 /* wake up processes */
133 : #endif    /* __CAN_H */
```

EK B.3

```
1 :  /*****************************************************
2 :                          -robotMove.c-
3 :  Bu program motor referanslarını robotun sürücü kartlarına
4 :  göndererek robotun hareket etmesini sağlar. Motor referanslarını
5 :  "motorAngles.bin" dosyasından okur.
6 :   Burak Tasasız tarafından 5 Şubat 2011 tarihinde yazılmıştır.
7 :  *****************************************************/
8 :  #include <stdio.h>
9 :  #include <stdlib.h>
10 : #include <string.h>
11 : #include <sys/types.h>
12 : #include <sys/stat.h>
13 : #include <fcntl.h>
14 : #include <unistd.h>
15 : #include <time.h>
16 : #include <math.h>
17 : #include "can4linux.h"
18 : #define STDDEV "can0"
19 : #define RXBUFFERSIZE 100
20 : #define MAX_SIZE 10000
21 : int fd,T_F;
22 : canmsg_t tx;
23 : canmsg_t rx[RXBUFFERSIZE];
24 : clock_t start, end;
25 : unsigned int elaptime;
26 : int currents[MAX_SIZE][12];
27 : long maxVel[12];
28 : long motorTra[MAX_SIZE][12];
29 : void initCan(void);
30 : void readCurrent(int,int);
31 : void absEncodersRead(void);
32 : void sendCan(int ID, char command, int index, int subindex,unsigned long object);
33 : void motorDon(int ID, long count);
34 : void motorInit(int ID);
35 : void readTrajectory(void);
36 : void myDelayUs(unsigned int start_value,unsigned int delay_value);
37 : void writeCurrent(void);
38 : int main(void)
39 : {
40 : int j,m,k;
41 : char key;
42 : readTrajectory();
43 : printf("op. time=%d s\n",T_F/100);
44 : printf("max. motor speeds (rpm)\n");
45 : printf("%8s%8s%8s%8s%8s%8s\n","HipZ","HipY","HipX","Knee","AnkleL","AnkleR");
46 : for(j=0;j<6;++j)
47 : {
48 : printf("%8ld", maxVel[j]);
49 : }
50 : printf("\n");
51 : for(j=6;j<12;++j)
52 : {
53 : printf("%8ld", maxVel[j]);
54 : }
55 : printf("\n");
56 : printf("Press 's' and enter to start motion...\n");
57 : do{
58 : key=getchar();
59 : }while(key!='s' && key!='S');
60 : printf("%8s%8s%8s%8s%8s%8s%8s%8s","  ","LHZ","LHY","LHX","LK","LAX","LAY");
61 : printf("%8s%8s%8s%8s%8s%8s\n","RHZ","RHY","RHX","RK","RAX","RAY");
62 : initCan();
63 : for(j=1;j<13;++j)
64 : motorInit(j);
```

```
65 :  close(fd);
66 :  initCan();
67 :  for(j=0;j<T_F;++j)
68 :  {
69 :  start=clock();
70 :  for(m=1;m<13;++m)
71 :  motorDon(m,motorTra[j][m-1]);
72 :  for(k=1;k<13;++k)
73 :  readCurrent(j,k);
74 :  if(j%50==0)
75 :  {
76 :  printf("%4.1f s =",j/100.);
77 :  for(k=0;k<12;++k)
78 :  {
79 :  printf("%8d",currents[j][k]);
80 :  }
81 :  printf("\n");
82 :  }
83 :  myDelayUs(start,10000);
84 :  }
85 :  writeCurrent();
86 :  close(fd);
87 :  return 0;
88 :  }
89 :  /* motorDon() FONKSIYONUNUN BAŞLANGICI */
90 :   /*Bu fonksiyon  istenilen motorun referansını artım cinsinden motorun bağlı olduğu
91 :  sürücü kartınının ilgili nesnesine yazmak için kullanılmıştır. */
92 :  void motorDon(int ID, long count)
93 :  {
94 :  sendCan(ID, 'w', 0x2062, 0x00, count);
95 :  }
96 :  /* motorInit() FONKSIYONUNUN BAŞLANGICI */
97 :  /*Bu fonksiyon  istenilen motorun harekete başlayabilmesi için bağlı olduğu
98 :   sürücü kartının ilgili nesnelerinde düzenlemeler yapmak için yazılmıştır. */
99 :  void motorInit(int ID)
100 : {
101 : sendCan(ID, 'w', 0x6060, 0x00, 0xFF);   // Modes of Operation: Position Mode
102 : myDelayUs(clock(),10000);
103 : sendCan(ID, 'w', 0x6040, 0x00, 0x0006);   // Control Word: Shutdown
104 : myDelayUs(clock(),10000);
105 : sendCan(ID, 'w', 0x6040, 0x00, 0x000F);   // Control Word: Switch-on
106 : }
107 :  /* initCan() FONKSIYONUNUN BAŞLANGICI */
108 :  /* Bu fonksiyon  bilgisayarın Can modülünün çalıştırılabilmesi için gereken
109 : ayarlamalar için yazılmıştır. */
110 : void initCan(void)
111 : {
112 : char device[40];
113 : sprintf(device, "/dev/%s", STDDEV);
114 : if(( fd = open(device, O_RDWR )) < 0 )
115 : {
116 : fprintf(stderr, "Error opening CAN device %s\n", device);
117 : exit(1);
118 : }
119 : }
120 :  /* sendCan() FONKSIYONUNUN BAŞLANGICI */
121 : /* Bu fonksiyon Can mesajlarını Sürücü Kartlarının istediği formatta oluşturmak
122 :  için yazılmıştır. */
123 : void sendCan(int ID, char command, int index, int subindex,unsigned long object)
124 : {
125 : int index_msb,index_lsb;
126 : int value[3];
127 : int got,sent,i;
128 : tx.id=0x600+ID;
129 : if(command == 'w')
130 : tx.data[0]=0x22;
131 : if(command == 'r')
```

```
132 : tx.data[0]=0x40;
133 : index_msb=index/256;
134 : index_lsb=index-index_msb*256;
135 : tx.data[1]=index_lsb;
136 : tx.data[2]=index_msb;
137 : tx.data[3]=subindex;
138 : value[3]=object/16777216;
139 : value[2]=(object-value[3]*16777216)/65536;
140 : value[1]=(object-value[3]*16777216-value[2]*65536)/256;
141 : value[0]=object-value[3]*16777216-value[2]*65536-value[1]*256;
142 : tx.data[4]=value[0];
143 : tx.data[5]=value[1];
144 : tx.data[6]=value[2];
145 : tx.data[7]=value[3];
146 : tx.flags=0;
147 : tx.length=8;
148 : do{
149 : sent = write(fd,&tx,1); //mesajin gonderilmesi
150 : }while(sent<1);
151 : do{
152 : got= read(fd,&rx,1);
153 : }while(got<1);
154 : }
155 : /* readTrajectory() FONKSIYONUNUN BAŞLANGICI */
156 : /* Bu fonksiyon, harekete başlamadan önce motor referanslarını motorTrajectory.bin
157 : dosyasından alır ve referansları motorTra isimli iki boyutlu dizinin içine atar. Böylelikle
158 :  hareket sırasında dosya okuma işlemi ile zaman kaybedilmez. */
159 : void readTrajectory()
160 : {
161 : FILE *fl;
162 : int count=0,i,Row[12];
163 : long temp[12],exPos[12];
164 : long velocity;
165 : for(i=0;i<12;i++)
166 : {
167 : exPos[i]=0;
168 : Row[i]=0;
169 : exPos[i]=0;
170 : }
171 : if( (fl=fopen("motorTrajectory.bin","rb")) == NULL)
172 : {
173 : printf("Can not open file\n");
174 : exit(EXIT_FAILURE);
175 : }
176 : while (fscanf(fl, "%ld%ld%ld%ld%ld%ld%ld%ld%ld%ld%ld%ld",&temp[0], &temp[1], &temp[2],
       &temp[3], &temp[4], &temp[5], &temp[6], &temp[7], &temp[8], &temp[9], &temp[10], &temp[11])
       != EOF)
177 : {
178 : for(i=0;i<12;++i)
179 : motorTra[count][i]=temp[i];
180 : if(count==0)
181 : for(i=0;i<12;++i)
182 : exPos[i]=motorTra[count][i];
183 : for(i=0;i<12;++i)
184 : {
185 : velocity= (3*(motorTra[count][i]-exPos[i])); //rpm cinsinden 2000 e ve 0.01 e bÃ¶l ve 60la carp
186 : if(abs(maxVel[i])<abs(velocity))
187 : {
188 : maxVel[i]=velocity;
189 : Row[i]=count;
190 : }
191 : exPos[i]=motorTra[count][i];
192 : }
193 : count++;
194 : if(count>MAX_SIZE)
195 : {
196 : printf("Size of trajectory arrays are too small !!!\n");
```

```
197 : exit(-1);
198 : }
199 : }
200 : T_F=count;
201 : for(i=0;i<12;++i)
202 : printf("Row=%d\t",Row[i]);
203 : printf("\n");
204 : fclose(fl);
205 : }
206 : /* myDelayUs() FONKSIYONUNUN BAŞLANGICI */
207 : /* Bu fonksiyon Motor referanslarının sürücü kartlarına belirlenen örnekleme zamanında
208 :  gönderilmesini sağlar. Bunu bilgisayarın gerçek zamanlı saatinden aldığı hassas zaman
209 : değeri ile yapar. Eğer sürücü kartlarına referans gönderme işi örnekleme
210 :  zamanını geçerse hata verir ve programın çalışmasını durdurur. */
211 : void myDelayUs(unsigned int start_value,unsigned int time_value)
212 : {
213 : //unsigned int elaptime;
214 : do{
215 : end=clock();
216 : elaptime=end-start_value;
217 : if(elaptime>10000)
218 : {
219 : printf("!!! elapsed time: %d us !!!\n", elaptime);
220 : exit(-1);
221 : }
222 : }while(elaptime<time_value);
223 : }
224 : /* readCurrent() FONKSIYONUNUN BAŞLANGICI */
225 : /* Bu fonksiyon istenilen motorun çektiği akım değerini bağlı olduğu sürücü kartının ilgili
226 :  nesnesinden okumak için yazılmıştır. */
227 : void readCurrent(int row,int ID)
228 : {
229 : unsigned int udata;
230 : //   sendCan(ID, 'r', 0x6078, 0x00, 0x00);   // Read Current
231 : sendCan(ID, 'r', 0x2027, 0x00, 0x00);   // Read Avarage Current
232 : udata= 256*rx[0].data[5]+rx[0].data[4];
233 : if(rx[0].data[5]>128)
234 : currents[row][ID-1] =udata-65536;
235 : else
236 : currents[row][ID-1] =udata;
237 : }
238 : /* writeCurrent() FONKSIYONUNUN BAŞLANGICI */
239 : /* Bu fonksiyon motorların okunan akım değerlerinin oluştural bir dosyada saklanması
240 :  için yazılmıştır. */
241 : void writeCurrent(void)
242 : {
243 : FILE *fcurrents,*fmotorTra;
244 : int i,k;
245 : time_t timer;
246 : struct tm *tptr;
247 : char fileNameCur[50]="currents_";
248 : char fileNameTra[50]="motTra_";
249 : char date[30];
250 : time(&timer);
251 : tptr = localtime(&timer);
252 : sprintf(date,"%02d%02d%02d_%02d:%02d:%02d\0",tptr->tm_mday,(tptr->tm_mon+1),(tptr-
       >tm_year + 1900)%100,tptr->tm_hour,tptr->tm_min,tptr->tm_sec);
253 : strcat(fileNameCur,date);
254 : strcat(fileNameTra,date);
255 : strcat(fileNameCur, ".bin");
256 : strcat(fileNameTra, ".bin");
257 : if( (fcurrents=fopen(fileNameCur,"wb")) == NULL)
258 : {
259 : printf("Can not open file\n");
260 : exit(EXIT_FAILURE);
261 : }
262 : if( (fmotorTra=fopen(fileNameTra,"wb")) == NULL)
```

```
263 : {
264 : printf("Can not open file\n");
265 : exit(EXIT_FAILURE);
266 : }
267 : for(k=0;k<T_F;++k)
268 : {
269 : for(i=0;i<12;++i)
270 : {
271 : fprintf(fcurrents,"%d\t",currents[k][i]);
272 : }
273 : fprintf(fcurrents,"\n");
274 : }
275 : for(k=0;k<T_F;++k)
276 : {
277 : for(i=0;i<12;++i)
278 : {
279 : fprintf(fmotorTra,"%ld\t",motorTra[k][i]);
280 : }
281 : fprintf(fmotorTra,"\n");
282 : }
283 : fclose(fmotorTra);
284 : fclose(fcurrents);
285 : }
```

EK B.4

```
1 : /*****************************************************
2 :                          -zeroPos.c-
3 : Bu program, robotun eklemlerini herhangi açı değerlerinden
4 : istenen açı değerlerine götürülmesi için her eklem için eklem
5 : referansları üretir. Eklemlerin açı değerleri mutlak kodlayıcılar
6 : tarafından ölçülür.
7 : 15 Mart 2011 tarihinde Burak Tasasız tarafından yazılmıştır.
8 : *****************************************************/
9 : #include <stdio.h>
10 : #include <stdlib.h>
11 : #include <string.h>
12 : #include <sys/types.h>
13 : #include <sys/stat.h>
14 : #include <fcntl.h>
15 : #include <unistd.h>
16 : #include <time.h>
17 : #include "can4linux.h"
18 : #define STDDEV "can0"
19 : #define PI 3.14159
20 : #define D2R PI/180.
21 : /* LIMITLERIN TANIMLANMASI */
22 : #define LHZ_max 20 // problem yok!
23 : #define LHZ_set 211.8
24 : #define LHZ_min -20
25 : #define LHZ_dir -1
26 : #define LHX_max 60 //problem yok!
27 : #define LHX_set 71.8
28 : #define LHX_min -60
29 : #define LHX_dir 1
30 : #define LHY_max 90 //kritik! sorunlu!
31 : #define LHY_set 306.4
32 : #define LHY_min -90
33 : #define LHY_dir 1
34 : #define LK_max 82 // problem yok!
35 : #define LK_set 137.2+2.6
36 : #define LK_min 5
37 : #define LK_dir 1
38 : #define LAY_max -1 // problem yok!
39 : #define LAY_set 194.9+1
40 : #define LAY_min -53
41 : #define LAY_dir -1
42 : #define LAX_max 30 // sorunlu!
43 : #define LAX_set 345.6 //344.8 zeki hocayla //359.8 labda degistirildi
44 : #define LAX_min -30
45 : #define LAX_dir 1
46 : #define RHZ_max 20 // problem yok!
47 : #define RHZ_set 112.6
48 : #define RHZ_min -20
49 : #define RHZ_dir -1
50 : #define RHX_max 60 //problem yok!
51 : #define RHX_set 83.6
52 : #define RHX_min -60
53 : #define RHX_dir 1
54 : #define RHY_max 90 //problem yok!
55 : #define RHY_set 216.5
56 : #define RHY_min -90
57 : #define RHY_dir -1
58 : #define RK_max 82 //sorunlu !
59 : #define RK_set 49.9-2
60 : #define RK_min 2
61 : #define RK_dir -1
62 : #define RAY_max -3 // problem yok!
63 : #define RAY_set 268.2-2//268.2
64 : #define RAY_min -52
```

```
65 :  #define RAY_dir 1
66 :  #define RAX_max 30 //sorunlu!
67 :  #define RAX_set 15.4 //17.2 Zeki hocayla // 28 labda degistirildi
68 :  #define RAX_min -30
69 :  #define RAX_dir 1
70 :  #define T_FINAL 5
71 :  #define RXBUFFERSIZE 100
72 :  int fd;
73 :  canmsg_t tx;
74 :  canmsg_t rx[RXBUFFERSIZE];
75 :  double absDegrees[12];
76 :  double i_jointDegrees[12];
77 :  ////////////////////////////////////////////
78 :  /* FINAL VALUES OF JOINT DEGREES ARE HERE */
79 :  double d_jointDegrees[12]=    { 0, 0, -21.4147, 44.0706, -22.6559, 0,
80 :  0, 0, -21.4147, 44.0706, -22.6559, 0 } ;
81 :  ////////////////////////////////////////////
82 :  unsigned int OCF[12], COF[12];
83 :  clock_t start, end;
84 :  unsigned int elaptime;
85 :  FILE *f_tra;
86 :  void initCan(void);
87 :  void sendCan(int ID, char command, int index, int subindex,unsigned long object);
88 :  void recieveCan(void);
89 :  void delayUs(unsigned int start_value,unsigned int delay_value);
90 :  void absEncodersRead(void);
91 :  int main(void)
92 :  {
93 :  int i;
94 :  double t;
95 :  double a1[12], a2[12], b1[12], b2[12], c1[12], c2[12];
96 :  //   printf("Starting...\n");
97 :  if( (f_tra=fopen("jointTrajectory.bin","wb")) ==NULL)
98 :  {
99 :  printf("Cannot open file... \n");
100 : exit(EXIT_FAILURE);
101 : }
102 : initCan();
103 : absEncodersRead();
104 : printf("LHZ=%.1f\tLHX=%.1f\tLHY=%.1f\tLK=%.1f\tLAY=%.1f\tLAX=%.1f\n",i_jointDegrees[0],i
      _jointDegrees[1],i_jointDegrees[2],i_jointDegrees[3],i_jointDegrees[4],i_jointDegrees[5]);
105 : printf("RHZ=%.1f\tRHX=%.1f\tRHY=%.1f\tRK=%.1f\tRAY=%.1f\tRAX=%.1f\n",i_jointDegrees[6],
      i_jointDegrees[7],i_jointDegrees[8],i_jointDegrees[9],i_jointDegrees[10],i_jointDegrees[11]);
106 : printf("OCF = %d %d %d %d %d %d %d %d %d %d %d %d
      \n",OCF[0],OCF[1],OCF[2],OCF[3],OCF[4],OCF[5],OCF[6],OCF[7],OCF[8],OCF[9],OCF[10],OCF[
      11]);
107 : printf("COF = %d %d %d %d %d %d %d %d %d %d %d %d
      \n",COF[0],COF[1],COF[2],COF[3],COF[4],COF[5],COF[6],COF[7],COF[8],COF[9],COF[10],COF[
      11]);
108 : for(i=0;i<12;++i)
109 : {
110 : a1[i]= 2*((d_jointDegrees[i]-i_jointDegrees[i])/(T_FINAL*T_FINAL));
111 : a2[i]= -1*a1[i];
112 : b1[i]= 0;
113 : b2[i]= -2*(a2[i]*T_FINAL);
114 : c1[i]= i_jointDegrees[i];
115 : c2[i]= d_jointDegrees[i]+(a2[i]*(T_FINAL*T_FINAL));
116 : }
117 : for(t=0;t<T_FINAL/2;t=t+0.01)
118 : {
119 : for(i=0;i<12;++i)
120 : fprintf(f_tra,"%.4f\t",D2R*(((a1[i]*t)*t)+c1[i]));
121 : fprintf(f_tra,"\n");
122 : }
123 : for(t=T_FINAL/2;t<T_FINAL;t=t+0.01)
124 : {
125 : for(i=0;i<12;++i)
```

```
126 : fprintf(f_tra,"%.4f\t",D2R*(a2[i]*t*t+b2[i]*t+c2[i]));
127 : fprintf(f_tra,"\n");
128 : }
129 : close(fd);
130 : fclose(f_tra);
131 : return 0;
132 : }
133 : /* absEncodersRead() FONKSIYONUNUN BAŞLANGICI */
134 : /* Bu fonksiyon Sürücü kartlarının dijital giriş ve çıkışları kullanılarak mutlak kodlayıcılardan
135 : eklem açılarının alınabilmesi için yazılmıştır.*/
136 : void absEncodersRead(void)
137 : {
138 : unsigned long data[12],dataTemp;
139 : int dataBIT,dataBin;
140 : do{                //status kontrolÃ¼ iÃ§in ! ! !
141 : printf("Reading Abs.Encoder...\n");
142 : sendCan(1, 'w', 0x2078, 0x02, 0x8000);  // Digital Output Func. Mask
143 : sendCan(2, 'w', 0x2078, 0x02, 0x8000);
144 : sendCan(3, 'w', 0x2078, 0x02, 0x8000);
145 : sendCan(4, 'w', 0x2078, 0x02, 0x8000);
146 : sendCan(5, 'w', 0x2078, 0x02, 0x8000);
147 : sendCan(6, 'w', 0x2078, 0x02, 0x8000);
148 : sendCan(7, 'w', 0x2078, 0x02, 0x8000);
149 : sendCan(8, 'w', 0x2078, 0x02, 0x8000);
150 : sendCan(9, 'w', 0x2078, 0x02, 0x8000);
151 : sendCan(10, 'w', 0x2078, 0x02, 0x8000);
152 : sendCan(11, 'w', 0x2078, 0x02, 0x8000);
153 : sendCan(12, 'w', 0x2078, 0x02, 0x8000);
154 : sendCan(1, 'w', 0x2071, 0x02, 0x8000); // Digital Input Func. Mask
155 : sendCan(2, 'w', 0x2071, 0x02, 0x8000);
156 : sendCan(3, 'w', 0x2071, 0x02, 0x8000);
157 : sendCan(4, 'w', 0x2071, 0x02, 0x8000);
158 : sendCan(5, 'w', 0x2071, 0x02, 0x8000);
159 : sendCan(6, 'w', 0x2071, 0x02, 0x8000);
160 : sendCan(7, 'w', 0x2071, 0x02, 0x8000);
161 : sendCan(8, 'w', 0x2071, 0x02, 0x8000);
162 : sendCan(9, 'w', 0x2071, 0x02, 0x8000);
163 : sendCan(10, 'w', 0x2071, 0x02, 0x8000);
164 : sendCan(11, 'w', 0x2071, 0x02, 0x8000);
165 : sendCan(12, 'w', 0x2071, 0x02, 0x8000);
166 : sendCan(1, 'w', 0x2078, 0x01, 0x8000); // CSn: HIGH
167 : sendCan(2, 'w', 0x2078, 0x01, 0x8000);
168 : sendCan(6, 'w', 0x2078, 0x01, 0x8000);
169 : sendCan(7, 'w', 0x2078, 0x01, 0x8000);
170 : sendCan(8, 'w', 0x2078, 0x01, 0x8000);
171 : sendCan(12, 'w', 0x2078, 0x01, 0x8000);
172 : sendCan(3, 'w', 0x2078, 0x01, 0x8000); // CLK: HIGH
173 : sendCan(4, 'w', 0x2078, 0x01, 0x8000);
174 : sendCan(5, 'w', 0x2078, 0x01, 0x8000);
175 : sendCan(9, 'w', 0x2078, 0x01, 0x8000);
176 : sendCan(10, 'w', 0x2078, 0x01, 0x8000);
177 : sendCan(11, 'w', 0x2078, 0x01, 0x8000);
178 : start=clock();
179 : delayUs(start,1000);
180 : for(dataBIT=0;dataBIT<12;++dataBIT)
181 : {
182 : data[dataBIT]=0;
183 : OCF[dataBIT]=0;
184 : COF[dataBIT]=0;
185 : }
186 : sendCan(1, 'w', 0x2078, 0x01, 0x0000); // CSn: LOW
187 : sendCan(2, 'w', 0x2078, 0x01, 0x0000);
188 : sendCan(6, 'w', 0x2078, 0x01, 0x0000);
189 : sendCan(7, 'w', 0x2078, 0x01, 0x0000);
190 : sendCan(8, 'w', 0x2078, 0x01, 0x0000);
191 : sendCan(12, 'w', 0x2078, 0x01, 0x0000);
192 : start=clock();
```

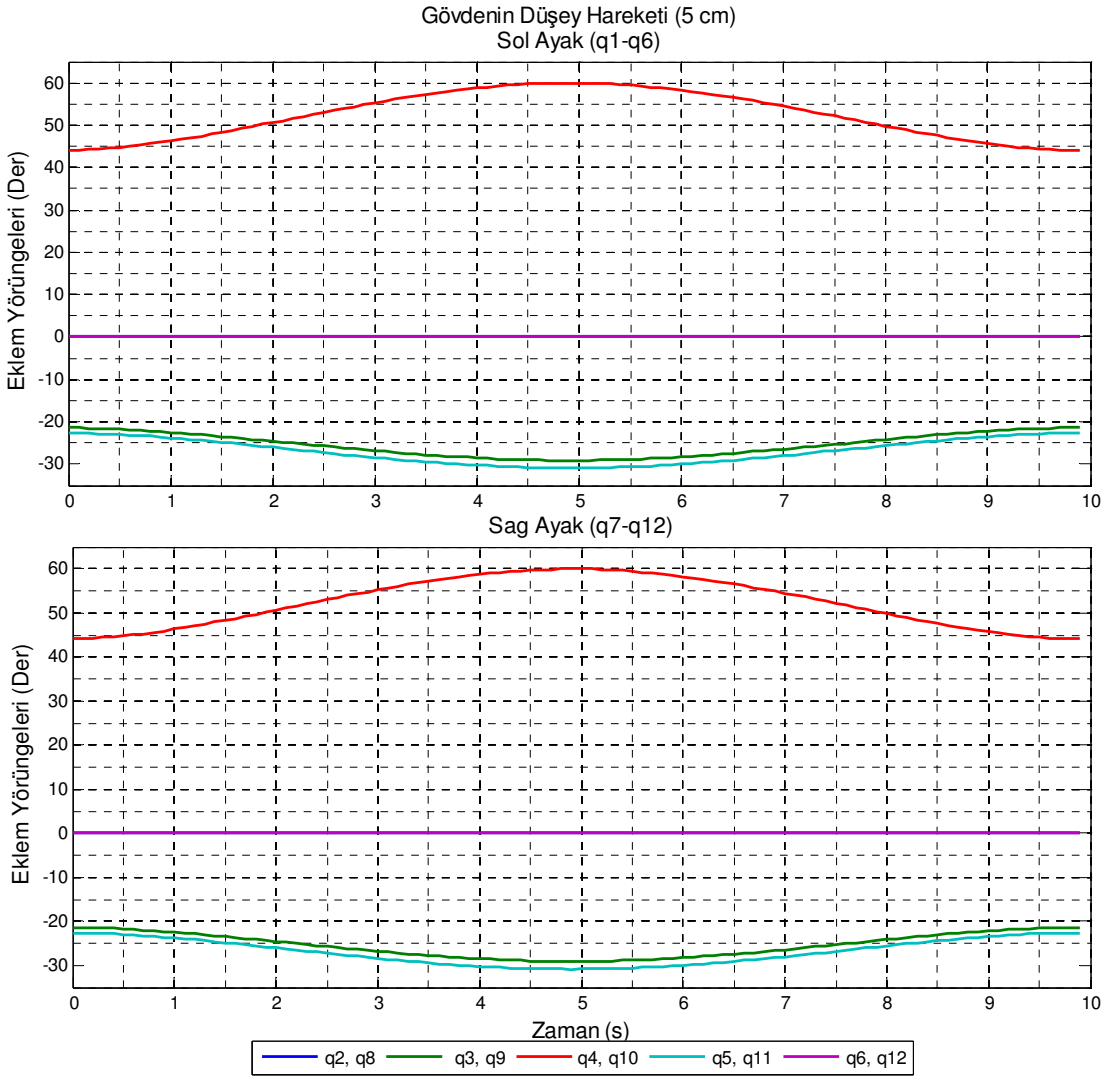
```
193 : delayUs(start,1000);
194 : for(dataBIT=18;dataBIT>0;dataBIT--)
195 : {
196 : sendCan(3, 'w', 0x2078, 0x01, 0x0000); // CLK: LOW
197 : sendCan(4, 'w', 0x2078, 0x01, 0x0000);
198 : sendCan(5, 'w', 0x2078, 0x01, 0x0000);
199 : sendCan(9, 'w', 0x2078, 0x01, 0x0000);
200 : sendCan(10, 'w', 0x2078, 0x01, 0x0000);
201 : sendCan(11, 'w', 0x2078, 0x01, 0x0000);
202 : start=clock();
203 : delayUs(start,1000);
204 : sendCan(3, 'w', 0x2078, 0x01, 0x8000); // CLK: HIGH
205 : sendCan(4, 'w', 0x2078, 0x01, 0x8000);
206 : sendCan(5, 'w', 0x2078, 0x01, 0x8000);
207 : sendCan(9, 'w', 0x2078, 0x01, 0x8000);
208 : sendCan(10, 'w', 0x2078, 0x01, 0x8000);
209 : sendCan(11, 'w', 0x2078, 0x01, 0x8000);
210 : start=clock();
211 : delayUs(start,1000);
212 : sendCan(1, 'r', 0x2071, 0x01, 0x00); // READ DATA: 1
213 : if(rx[0].data[5]==0x80)
214 : dataBin=1;
215 : else
216 : dataBin=0;
217 : dataTemp=dataBin<<(dataBIT-1);
218 : data[0]=dataTemp + data[0];
219 : sendCan(2, 'r', 0x2071, 0x01, 0x00); // READ DATA: 2
220 : if(rx[0].data[5]==0x80)
221 : dataBin=1;
222 : else
223 : dataBin=0;
224 : dataTemp=dataBin<<(dataBIT-1);
225 : data[1]=dataTemp + data[1];
226 : sendCan(3, 'r', 0x2071, 0x01, 0x00); // READ DATA: 3
227 : if(rx[0].data[5]==0x80)
228 : dataBin=1;
229 : else
230 : dataBin=0;
231 : dataTemp=dataBin<<(dataBIT-1);
232 : data[2]=dataTemp + data[2];
233 : sendCan(4, 'r', 0x2071, 0x01, 0x00); // READ DATA: 4
234 : if(rx[0].data[5]==0x80)
235 : dataBin=1;
236 : else
237 : dataBin=0;
238 : dataTemp=dataBin<<(dataBIT-1);
239 : data[3]=dataTemp + data[3];
240 : sendCan(5, 'r', 0x2071, 0x01, 0x00); // READ DATA: 5
241 : if(rx[0].data[5]==0x80)
242 : dataBin=1;
243 : else
244 : dataBin=0;
245 : dataTemp=dataBin<<(dataBIT-1);
246 : data[4]=dataTemp + data[4];
247 : sendCan(6, 'r', 0x2071, 0x01, 0x00); // READ DATA: 6
248 : if(rx[0].data[5]==0x80)
249 : dataBin=1;
250 : else
251 : dataBin=0;
252 : dataTemp=dataBin<<(dataBIT-1);
253 : data[5]=dataTemp + data[5];
254 : sendCan(7, 'r', 0x2071, 0x01, 0x00); // READ DATA: 7
255 : if(rx[0].data[5]==0x80)
256 : dataBin=1;
257 : else
258 : dataBin=0;
259 : dataTemp=dataBin<<(dataBIT-1);
```

```
260 : data[6]=dataTemp + data[6];
261 : sendCan(8, 'r', 0x2071, 0x01, 0x00); // READ DATA: 8
262 : if(rx[0].data[5]==0x80)
263 : dataBin=1;
264 : else
265 : dataBin=0;
266 : dataTemp=dataBin<<(dataBIT-1);
267 : data[7]=dataTemp + data[7];
268 : sendCan(9, 'r', 0x2071, 0x01, 0x00); // READ DATA: 9
269 : if(rx[0].data[5]==0x80)
270 : dataBin=1;
271 : else
272 : dataBin=0;
273 : dataTemp=dataBin<<(dataBIT-1);
274 : data[8]=dataTemp + data[8];
275 : sendCan(10, 'r', 0x2071, 0x01, 0x00); // READ DATA: 10
276 : if(rx[0].data[5]==0x80)
277 : dataBin=1;
278 : else
279 : dataBin=0;
280 : dataTemp=dataBin<<(dataBIT-1);
281 : data[9]=dataTemp + data[9];
282 : sendCan(11, 'r', 0x2071, 0x01, 0x00); // READ DATA: 11
283 : if(rx[0].data[5]==0x80)
284 : dataBin=1;
285 : else
286 : dataBin=0;
287 : dataTemp=dataBin<<(dataBIT-1);
288 : data[10]=dataTemp + data[10];
289 : sendCan(12, 'r', 0x2071, 0x01, 0x00); // READ DATA: 12
290 : if(rx[0].data[5]==0x80)
291 : dataBin=1;
292 : else
293 : dataBin=0;
294 : dataTemp=dataBin<<(dataBIT-1);
295 : data[11]=dataTemp + data[11];
296 : }
297 : for(dataBIT=0;dataBIT<12;++dataBIT)
298 : {
299 : absDegrees[dataBIT]=(360./4096.)*(data[dataBIT]>>6);
300 : OCF[dataBIT]=(data[dataBIT]>>5)%2;
301 : COF[dataBIT]=(data[dataBIT]>>4)%2;
302 : }
303 : }while( OCF[0]==0 || OCF[1]==0 || OCF[2]==0 || OCF[3]==0 || OCF[4]==0 || OCF[5]==0 ||
304 : OCF[6]==0 || OCF[7]==0 || OCF[8]==0 || OCF[9]==0 || OCF[10]==0 || OCF[11]==0 ||
305 : COF[0]==1 || COF[1]==1 || COF[2]==1 || COF[3]==1 || COF[4]==1 || COF[5]==1 ||
306 : COF[6]==1 || COF[7]==1 || COF[8]==1 || COF[9]==1 || COF[10]==1 || COF[11]==1 );
307 : /* LHZ */
308 : i_jointDegrees[0] = LHZ_dir*(absDegrees[0]-LHZ_set)*0.6;
309 : /* LHX */
310 : i_jointDegrees[1] = LHX_dir*(absDegrees[1]-LHX_set);
311 : /* LHY */
312 : if(absDegrees[2]<100)
313 : absDegrees[2]= 360+absDegrees[2];
314 : i_jointDegrees[2] = LHY_dir*(absDegrees[2]-LHY_set);
315 : /* LK */
316 : /* diz en asagi konumdayken sifir kabul ediliyor */
317 : i_jointDegrees[3] = LK_dir*(absDegrees[3]-LK_set);
318 : /* LAY */
319 : /* ayak burnu en asagidayken sifir kabul ediliyor */
320 : i_jointDegrees[4] = LAY_dir*(absDegrees[4]-LAY_set);
321 : /* LAX */
322 : if(absDegrees[5]<100)
323 : absDegrees[5]= 360+absDegrees[5];
324 : i_jointDegrees[5] = LAX_dir*(absDegrees[5]-LAX_set);
325 : /* RHZ */
326 : i_jointDegrees[6] = RHZ_dir*(absDegrees[6]-RHZ_set)*0.6;
```

```
/* RHX */
i_jointDegrees[7] = RHX_dir*(absDegrees[7]-RHX_set);
/* RHY */
i_jointDegrees[8] = RHY_dir*(absDegrees[8]-RHY_set);
/* RK */
/* diz en asagi konumdayken sifir kabul ediliyor */
if(absDegrees[9]>200)
absDegrees[9]= absDegrees[9]-360;
i_jointDegrees[9] = RK_dir*(absDegrees[9]-RK_set);
/* RAY */
/* ayak burnu en asagidayken sifir kabul ediliyor */
i_jointDegrees[10] = RAY_dir*(absDegrees[10]-RAY_set);
/* RAX */
if(absDegrees[11]>200)
absDegrees[11]= absDegrees[11]-360;
i_jointDegrees[11] = RAX_dir*(absDegrees[11]-RAX_set);
}
/* initCan() FONKSIYONUNUN BAŞLANGICI */
/* Bu fonksiyon  bilgisayarın Can modülünün çalıştırılabilmesi için gereken
 ayarlamalar için yazılmıştır. */
void initCan(void)
{
char device[40];
//        printf("Initializing Can...\n");
sprintf(device, "/dev/%s", STDDEV);
if(( fd = open(device, O_RDWR )) < 0 )
{
fprintf(stderr, "Error opening CAN device %s\n", device);
exit(1);
}
}
/* sendCan() FONKSIYONUNUN BAŞLANGICI */
/* Bu fonksiyon Can mesajlarını Sürücü Kartlarının istediği formatta oluşturmak
için yazılmıştır. */
void sendCan(int ID, char command, int index, int subindex,unsigned long object)
{
int index_msb,index_lsb;
int value[3];
int got,sent,i;
tx.id=0x600+ID;
if(command == 'w')
tx.data[0]=0x22;
if(command == 'r')
tx.data[0]=0x40;
index_msb=index/256;
index_lsb=index-index_msb*256;
tx.data[1]=index_lsb;
tx.data[2]=index_msb;
tx.data[3]=subindex;
value[3]=object/16777216;
value[2]=(object-value[3]*16777216)/65536;
value[1]=(object-value[3]*16777216-value[2]*65536)/256;
value[0]=object-value[3]*16777216-value[2]*65536-value[1]*256;
tx.data[4]=value[0];
tx.data[5]=value[1];
tx.data[6]=value[2];
tx.data[7]=value[3];
tx.flags=0;
tx.length=8;
do{
sent = write(fd,&tx,1); //mesajin gonderilmesi
}while(sent<1);
do{
got= read(fd,&rx,1);
}while(got<1);
}
```

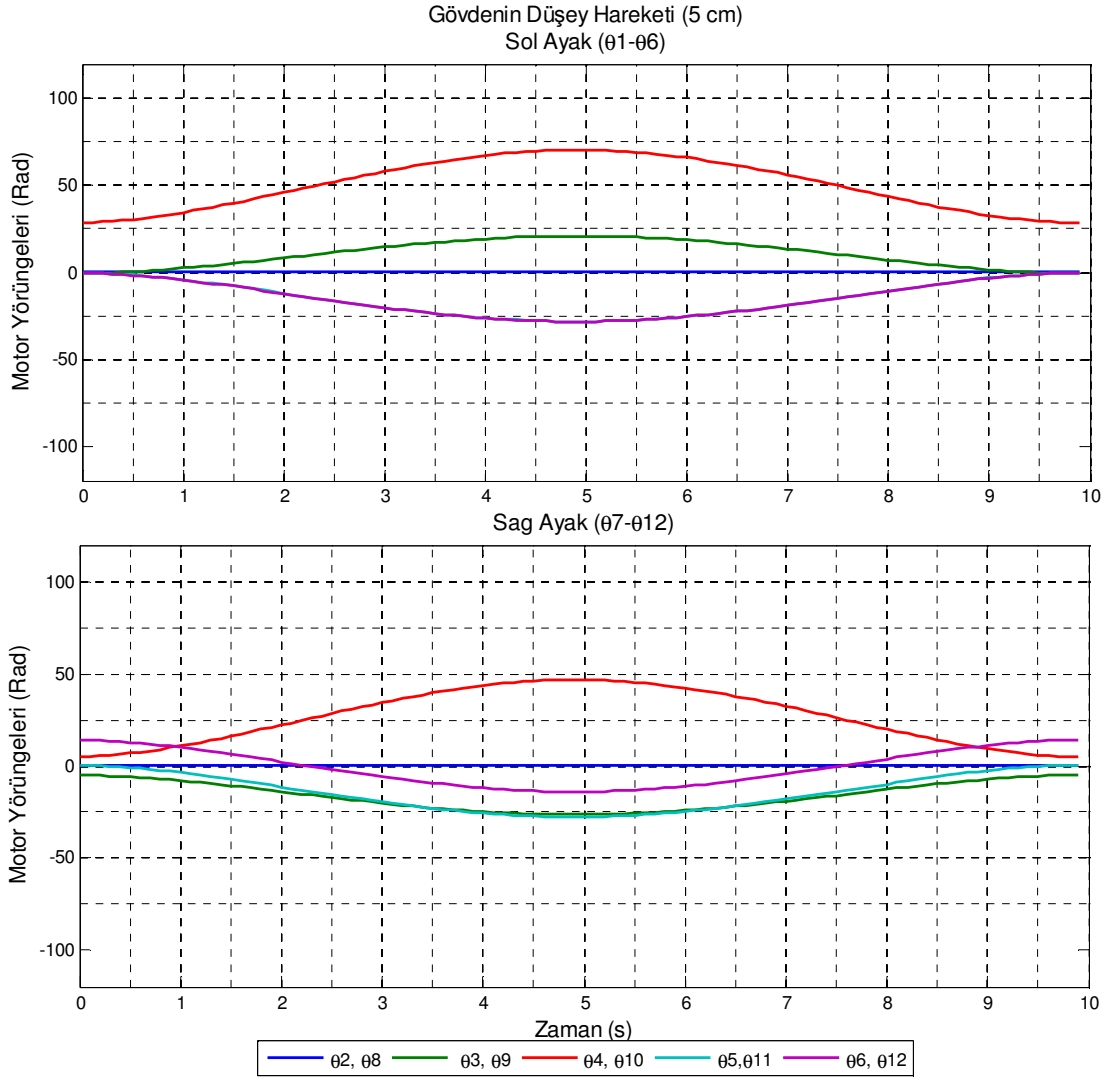
```
394 : /* delayUs() FONKSIYONUNUN BAŞLANGICI */
395 : /* Bu fonksiyon bilgisayarın gerçek zamanlı saatinden aldığı hassas zaman değeri ile
396 : mikro saniye cinsinden gecikmeler oluşturabilmek için yazılmıştır. */
397 : void delayUs(unsigned int start_value,unsigned int time_value)
398 : {
399 : do{
400 : end=clock();
401 : elaptime=end-start_value;
402 : }while(elaptime<time_value);
403 : }
```

EK C.1



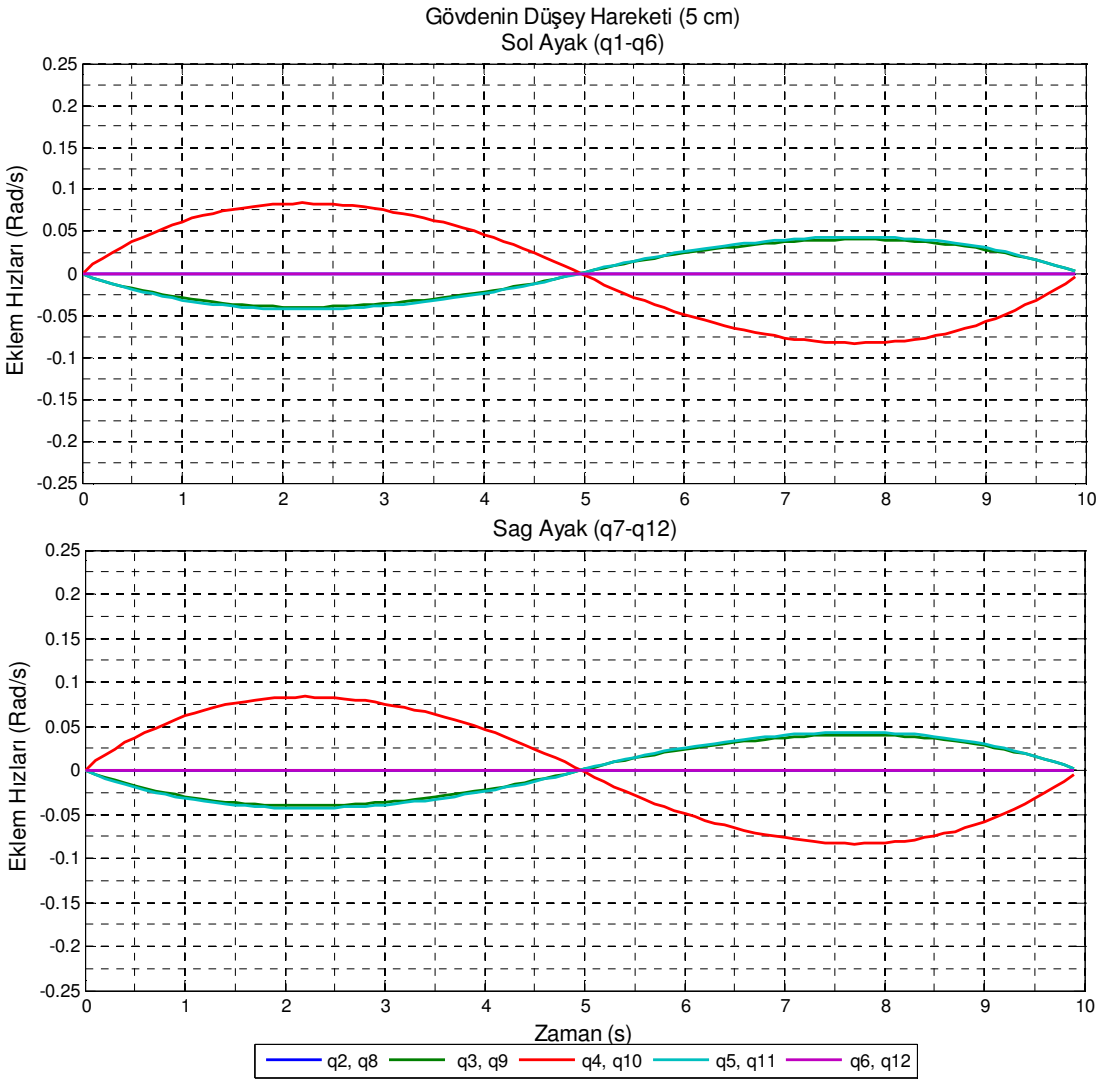
Şekil C.1 : Gövdenin 5 cm'lik düşey hareketi için referans eklem yörüngeleri.

EK C.2



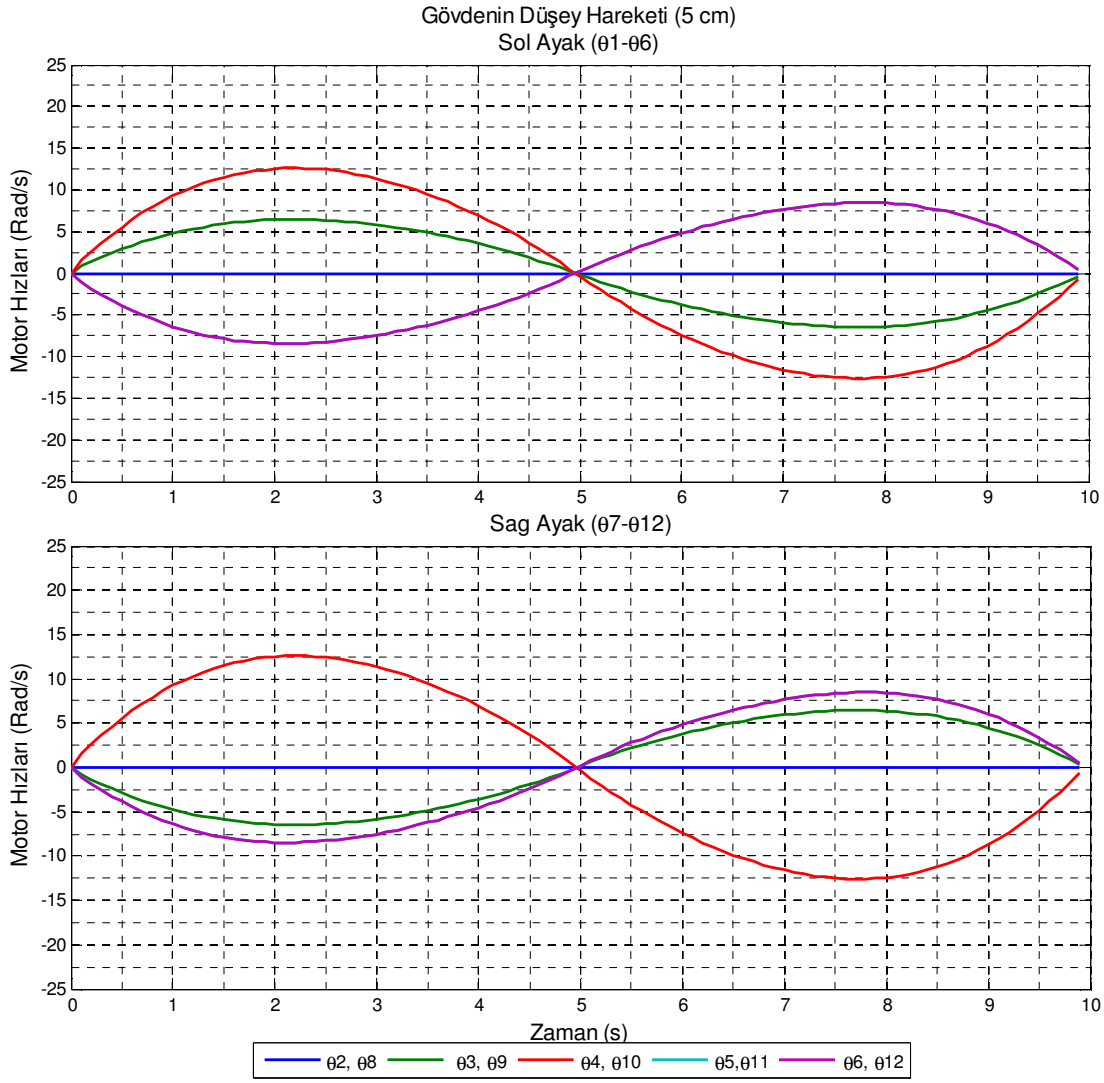
Şekil C.2 : Gövdenin 5 cm'lik düşey hareketi için referans motor yörüngeleri.

EK C.3



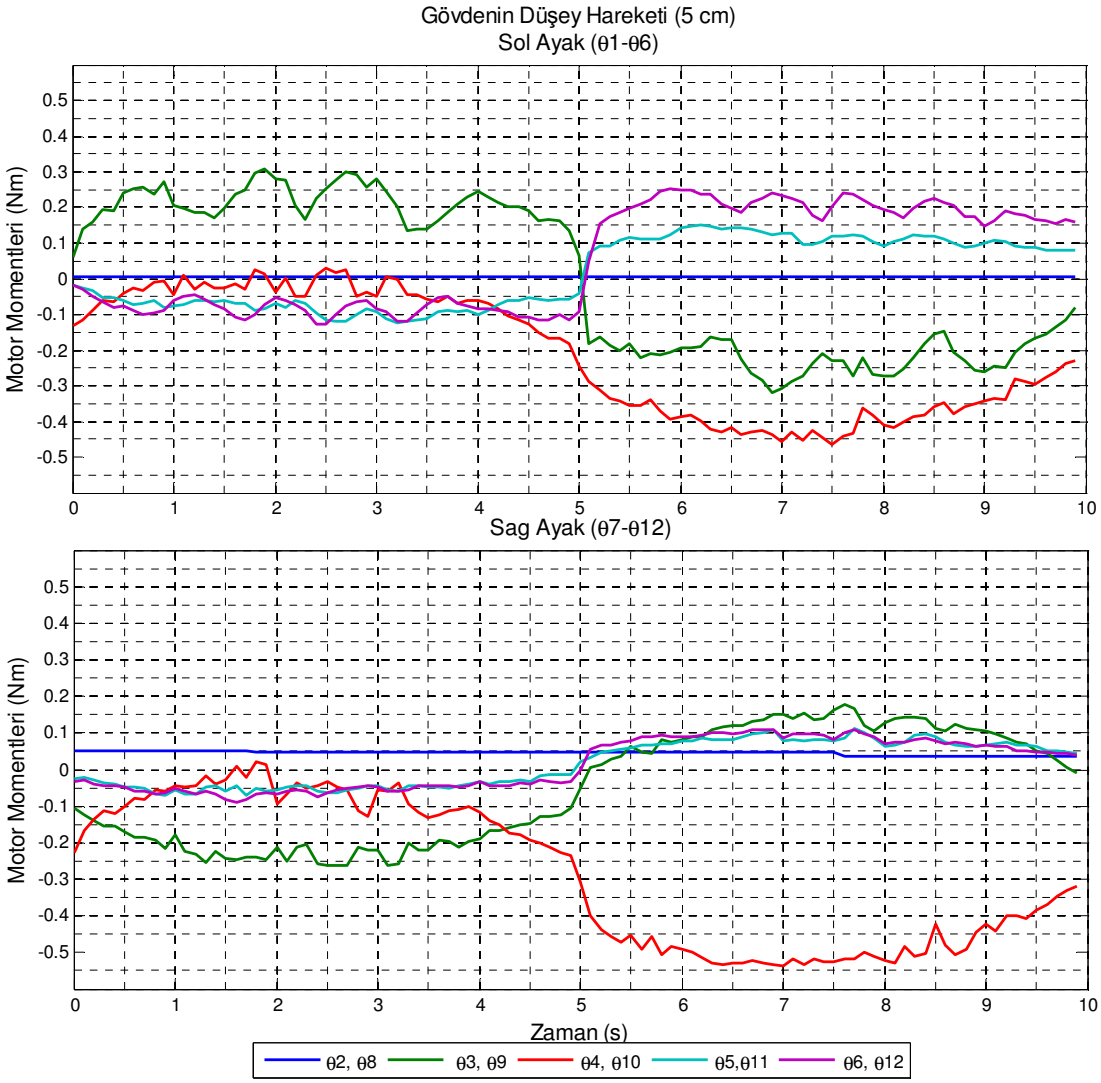
Şekil C.3 : Gövdenin 5 cm'lik düşey hareketi için referans eklem hızları.

EK C.4



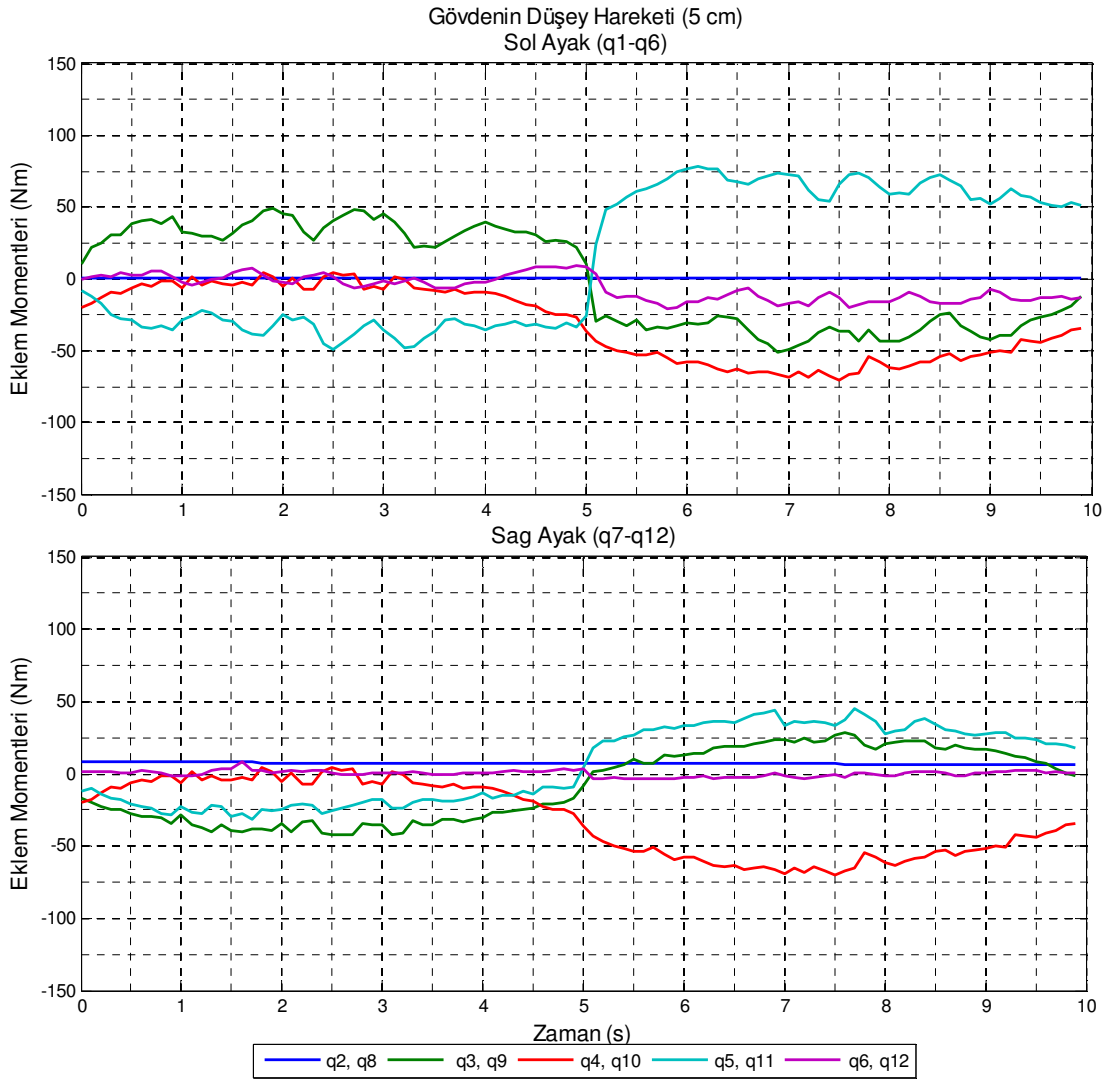
Şekil C.4 : Gövdenin 5 cm'lik düşey hareketi için referans motor hızları.

EK C.5



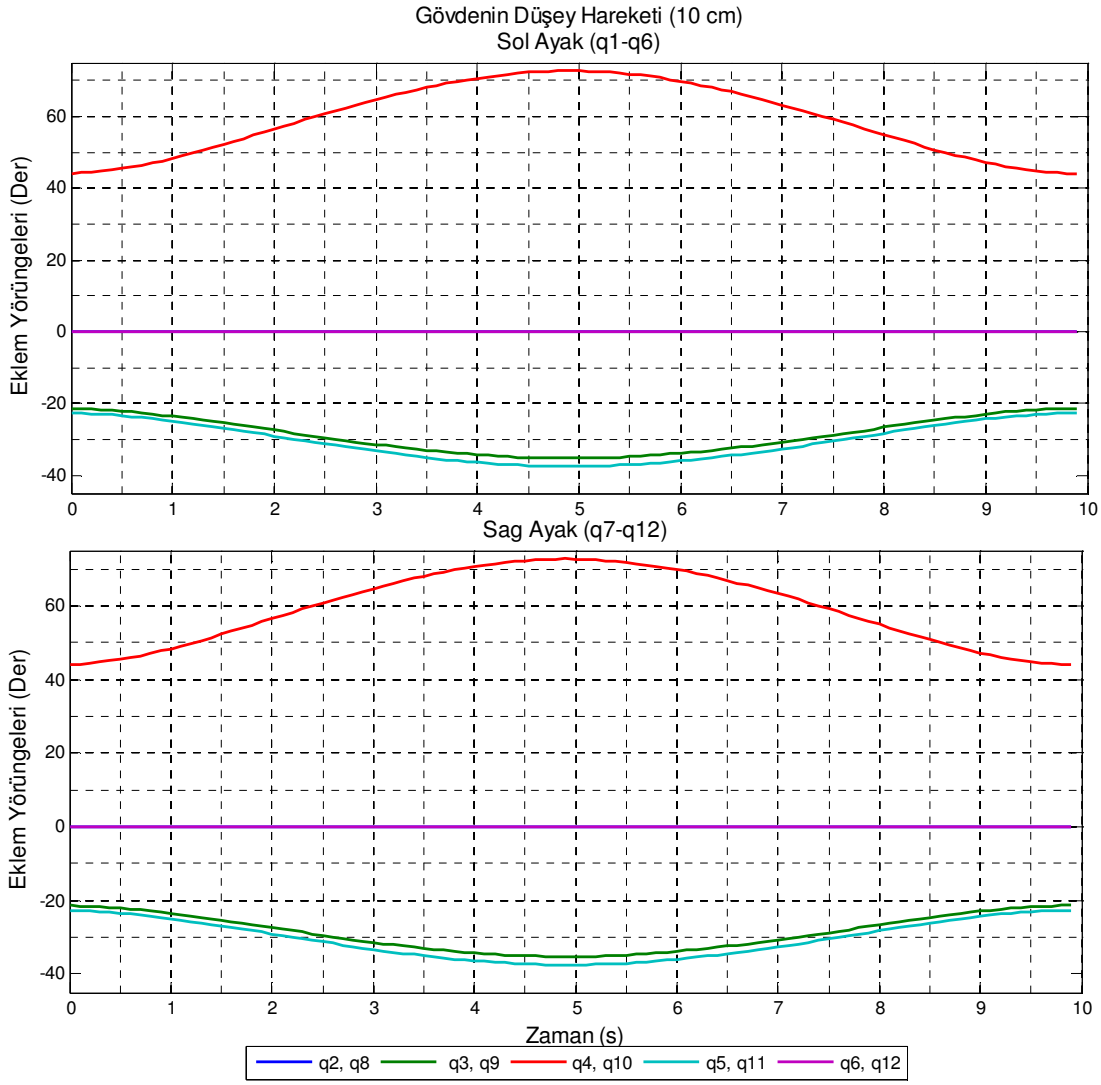
Şekil C.5 : Gövdenin 5 cm'lik düşey hareketinde ölçülen motor momentleri.

EK C.6



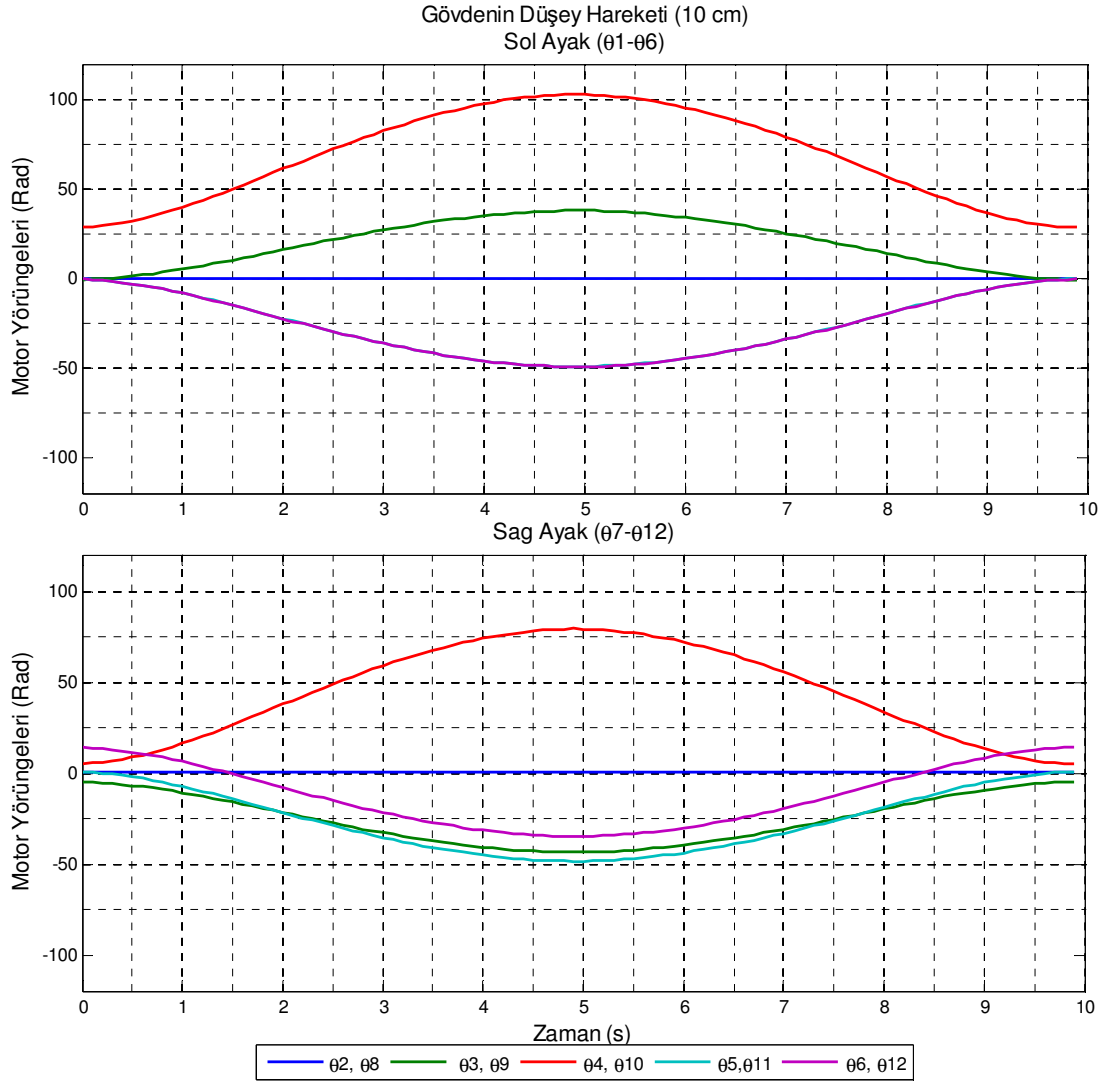
Şekil C.6 : Gövdenin 5 cm'lik düşey hareketindeki eklem momentleri.

EK C.7



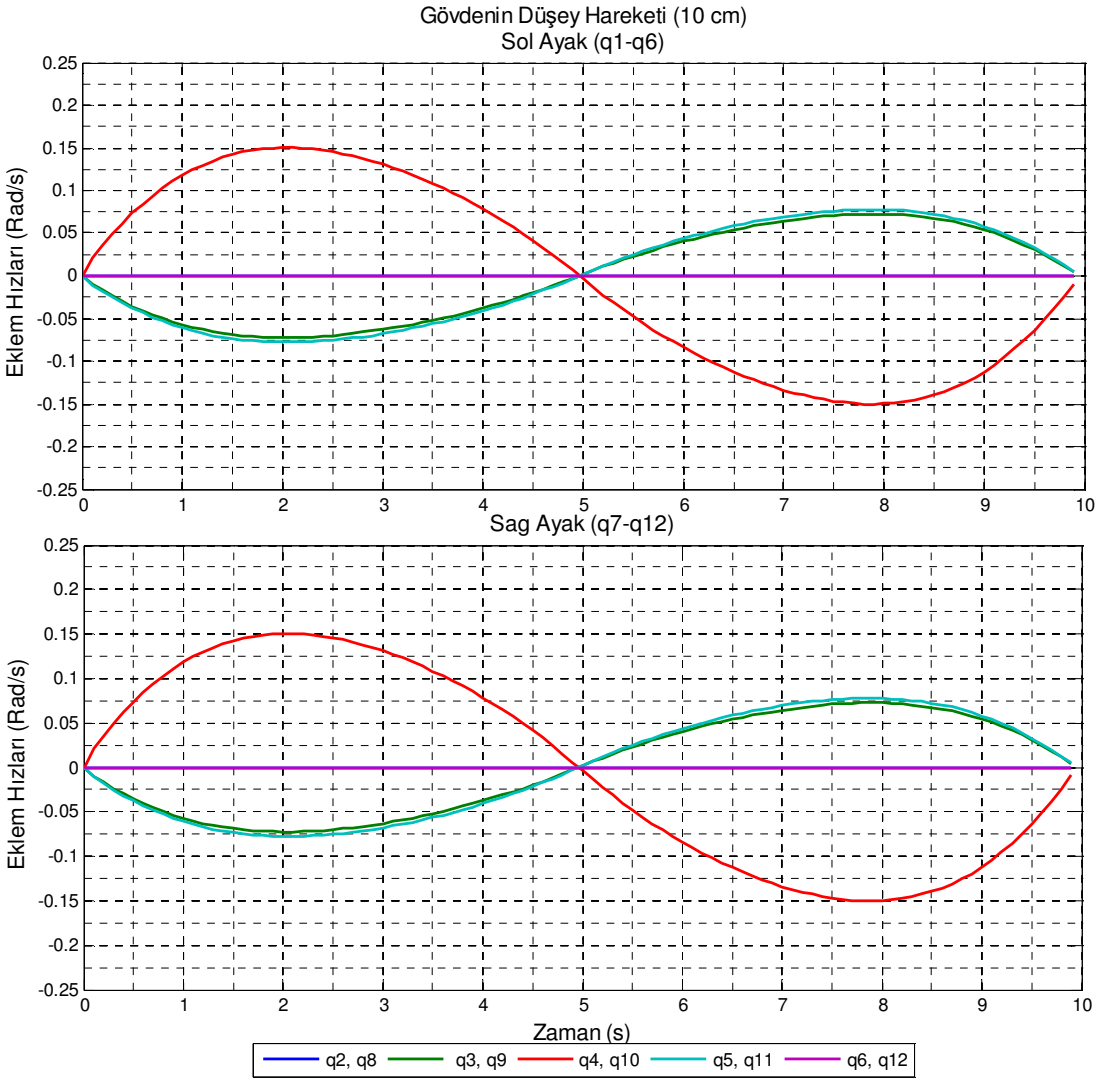
Şekil C.7 : Gövdenin 10 cm'lik düşey hareketi için referans eklem yörüngeleri.

EK C.8



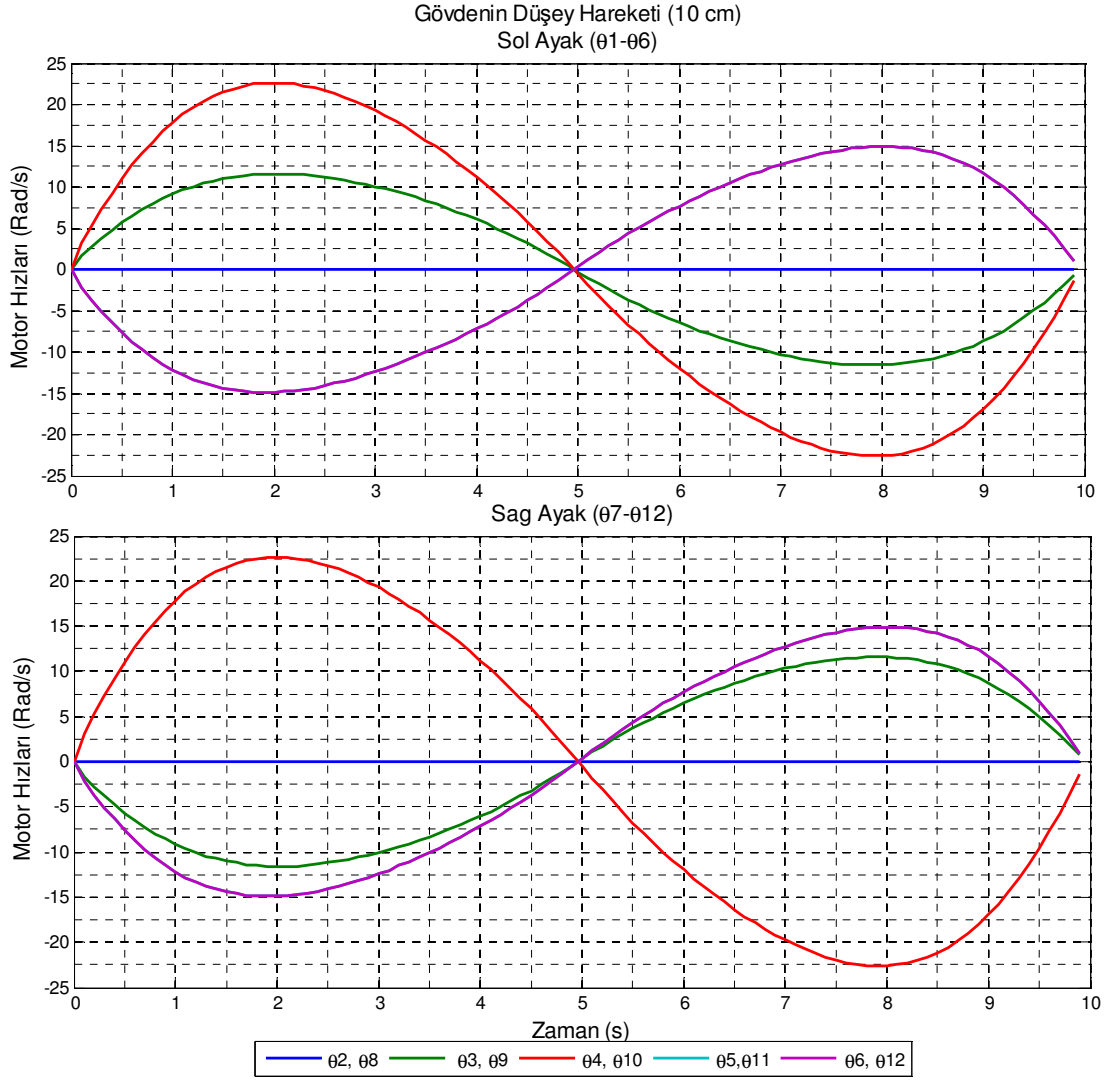
Şekil C.8 : Gövdenin 10 cm'lik düşey hareketi için referans motor yörüngeleri.

EK C.9



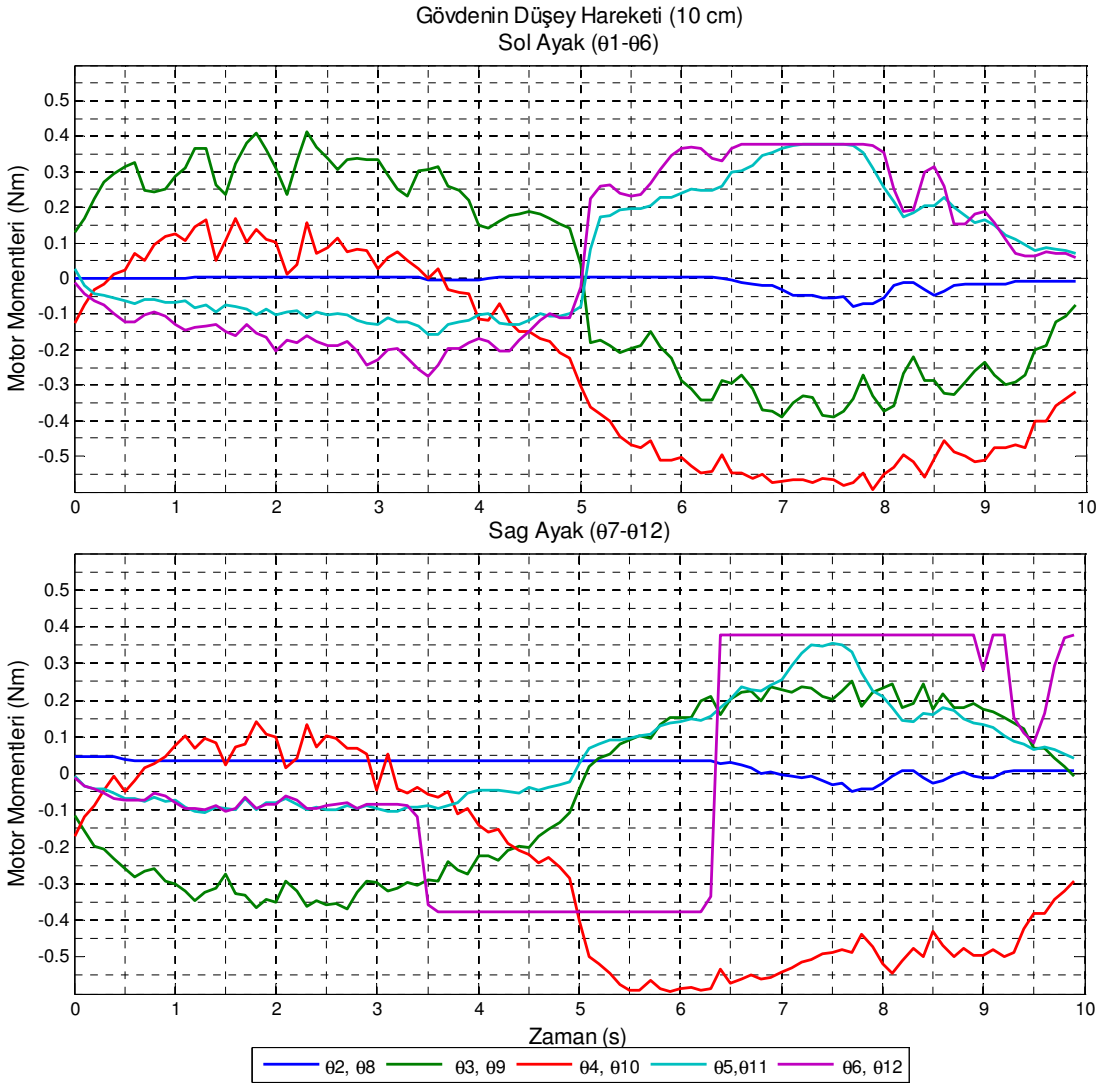
Şekil C.9 : Gövdenin 10 cm'lik düşey hareketi için referans eklem hızları.

EK C.10



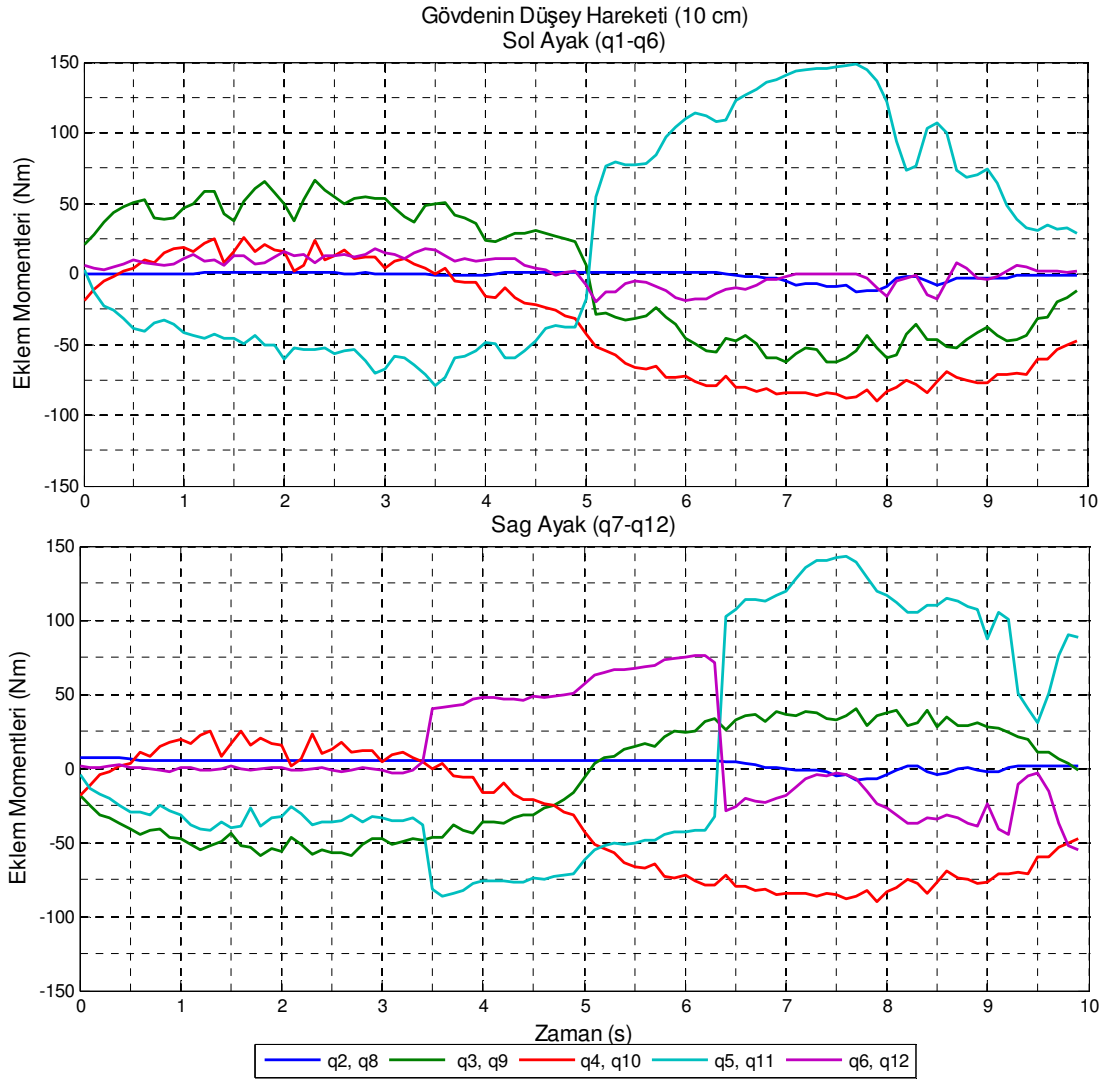
Şekil C.10 : Gövdenin 10 cm'lik düşey hareketi için referans motor hızları.

EK C.11



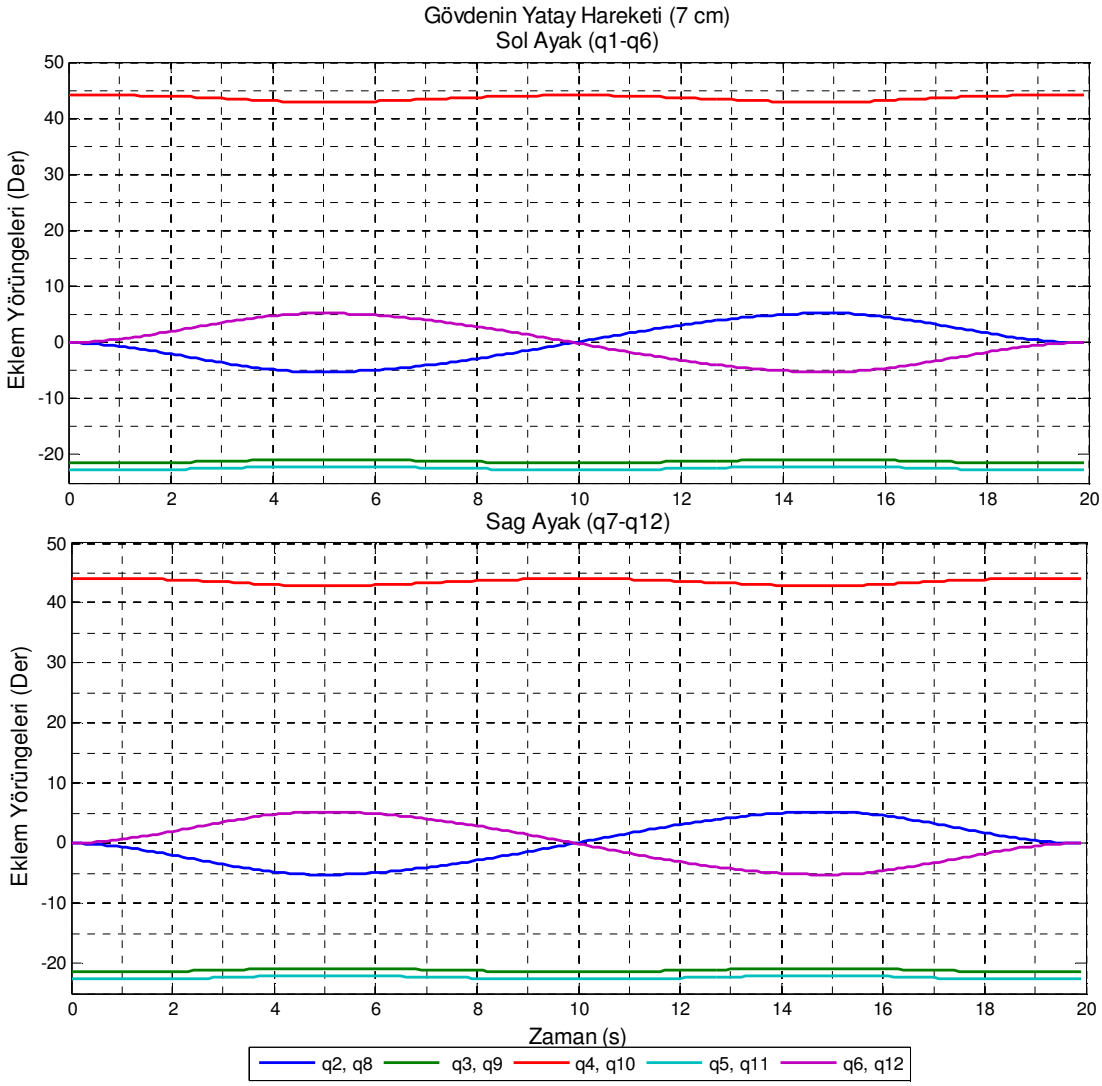
Şekil C.11 : Gövdenin 10 cm'lik düşey hareketinde ölçülen motor momentleri.

EK C.12



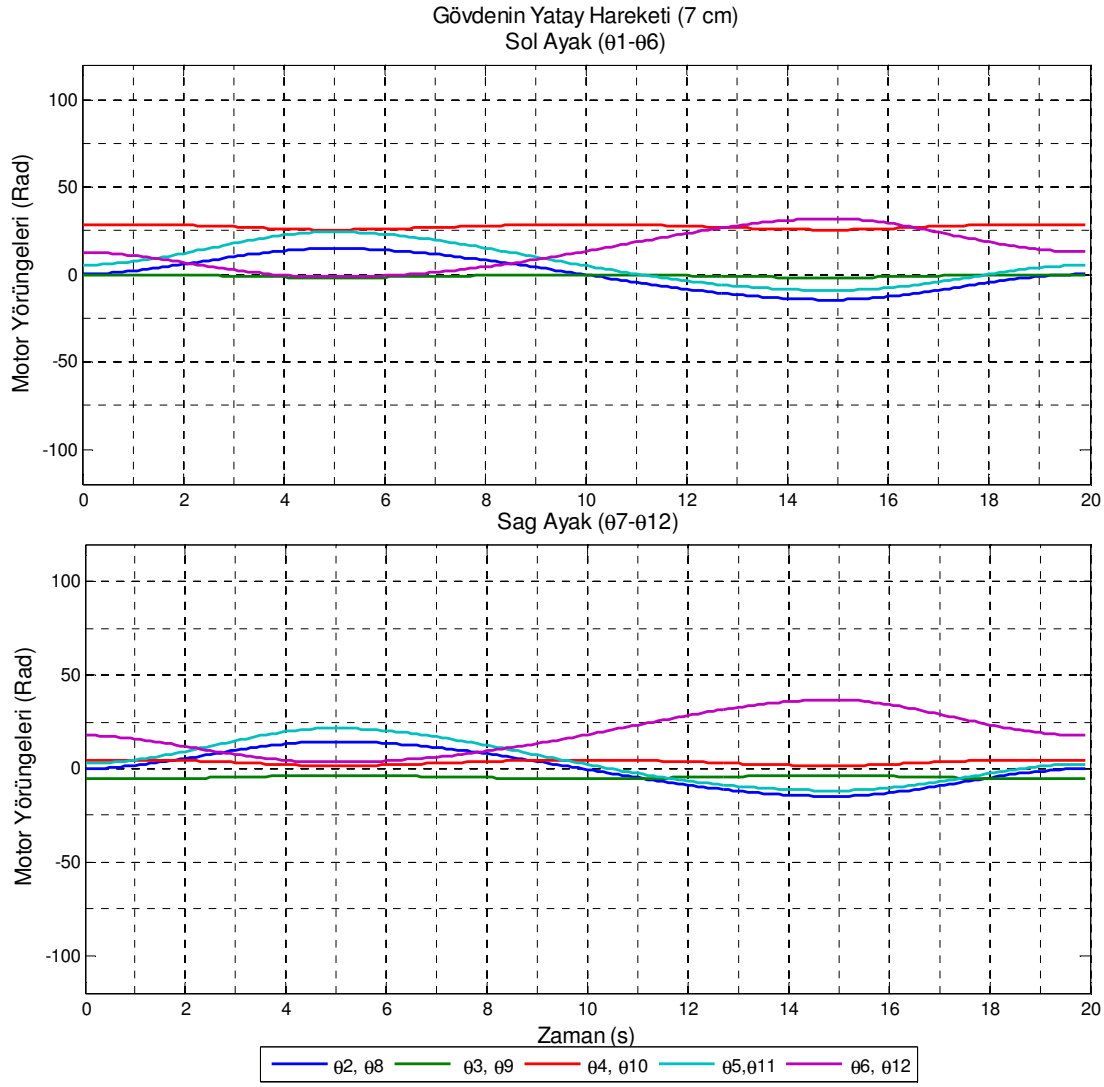
Şekil C.12 : Gövdenin 10 cm'lik düşey hareketindeki eklem momentleri.

EK C.13



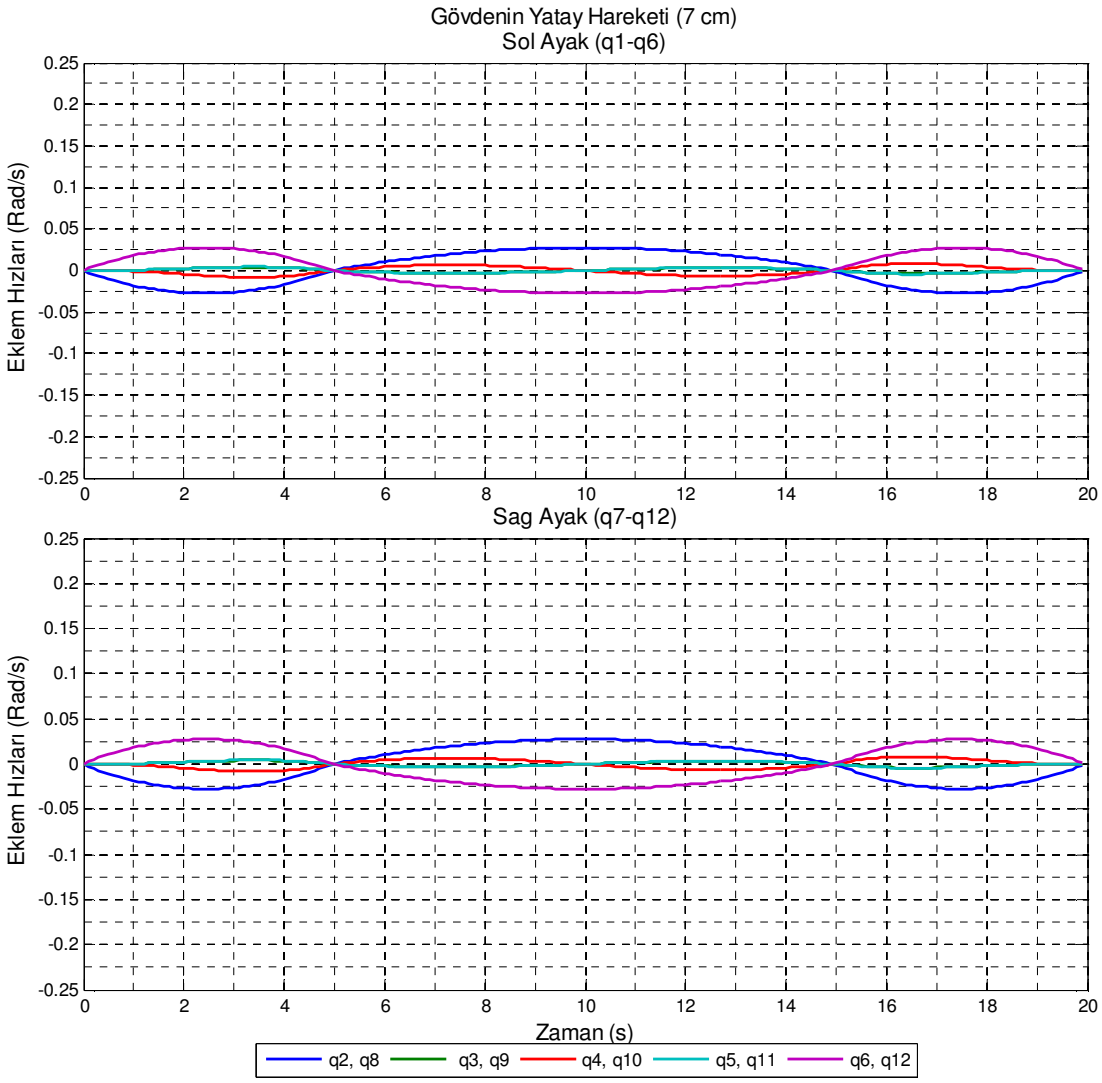
Şekil C.13 : Gövdenin 7 cm'lik yatay hareketi için referans eklem yörüngeleri.

EK C.14



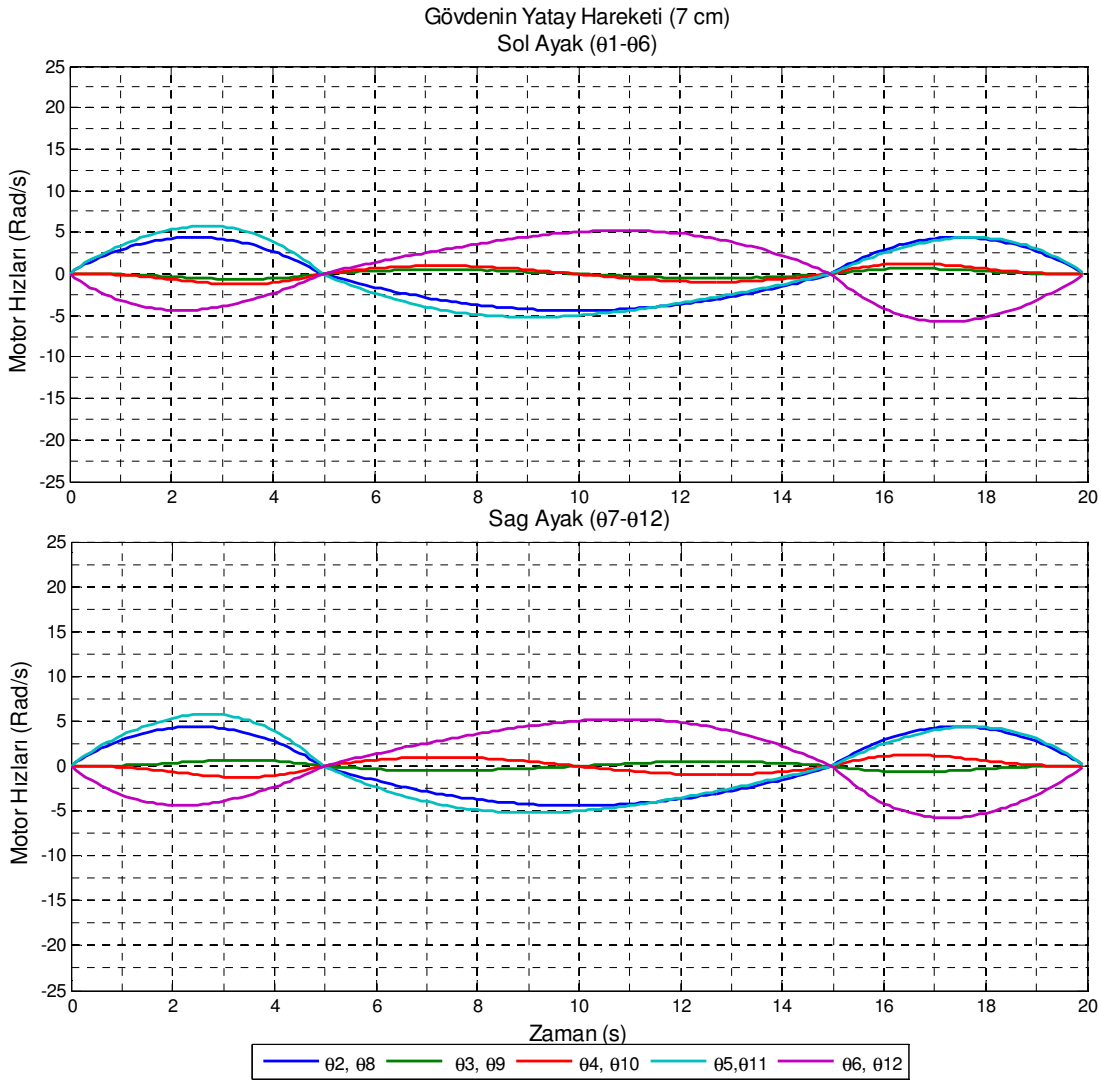
Şekil C.14 : Gövdenin 7 cm'lik yatay hareketi için referans motor yörüngeleri.

EK C.15



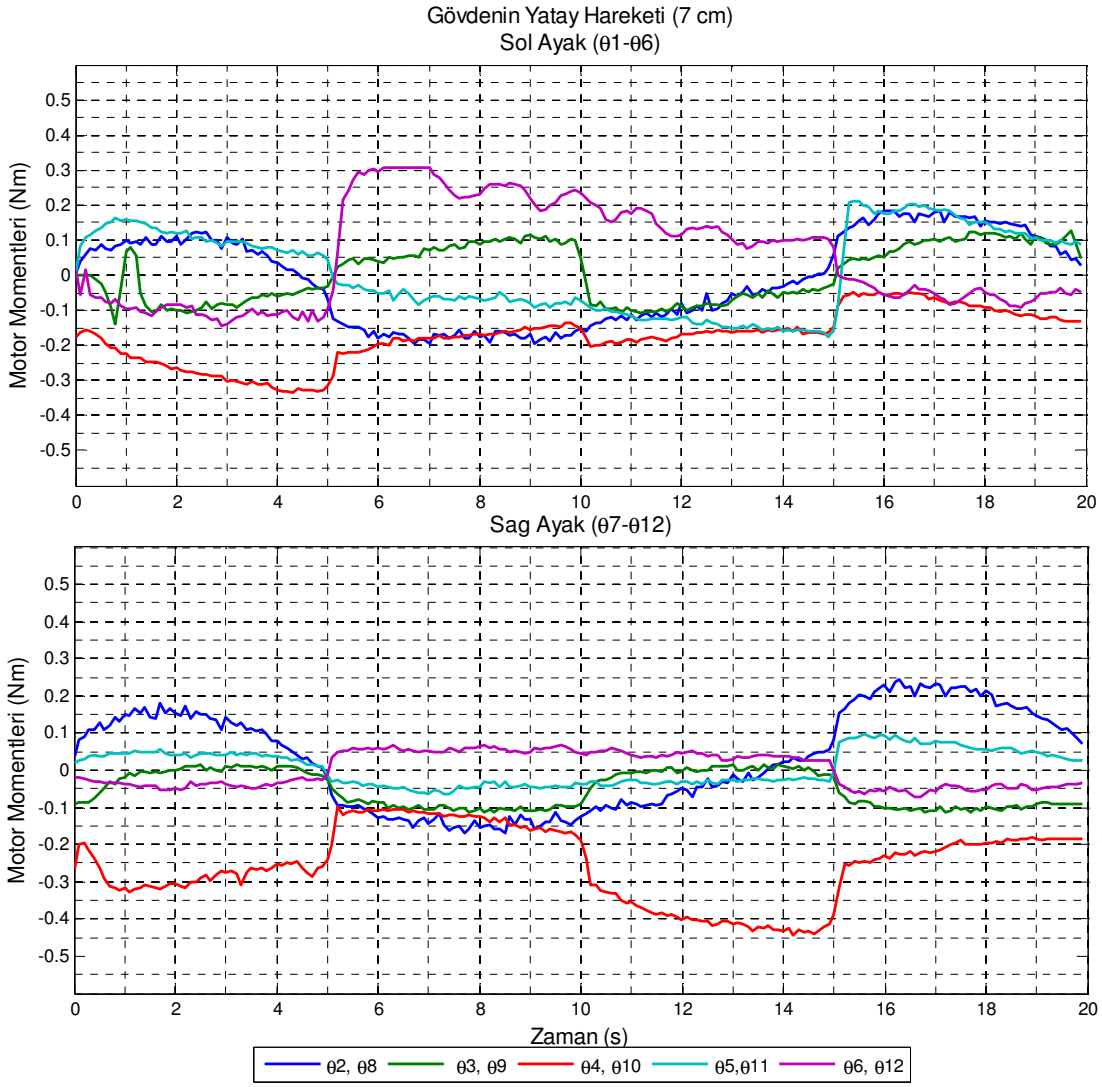
Şekil C.15 : Gövdenin 7 cm'lik yatay hareketi için referans eklem hızları.

EK C.16



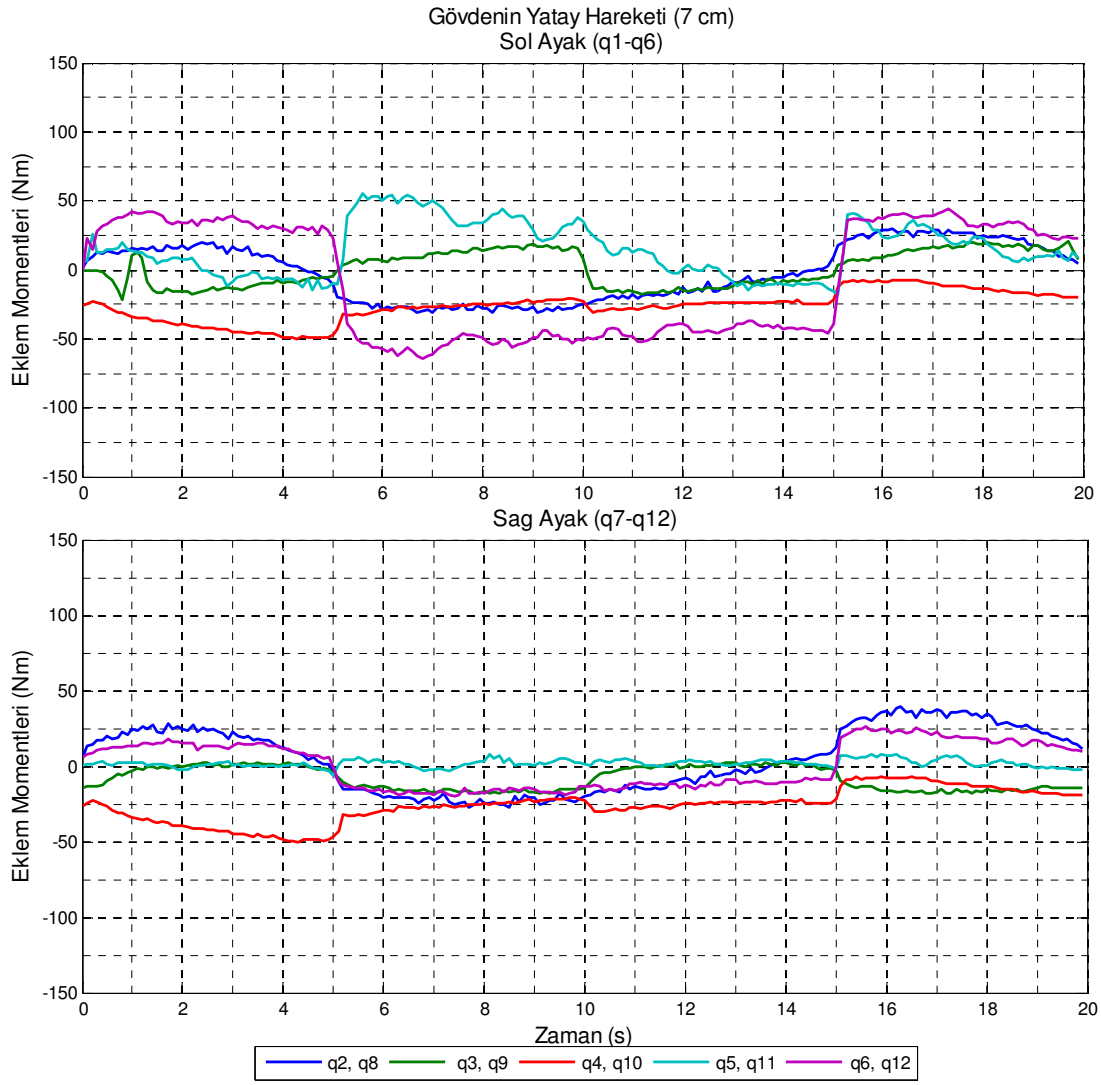
Şekil C.16 : Gövdenin 7 cm'lik yatay hareketi için referans motor hızları.

EK C.17



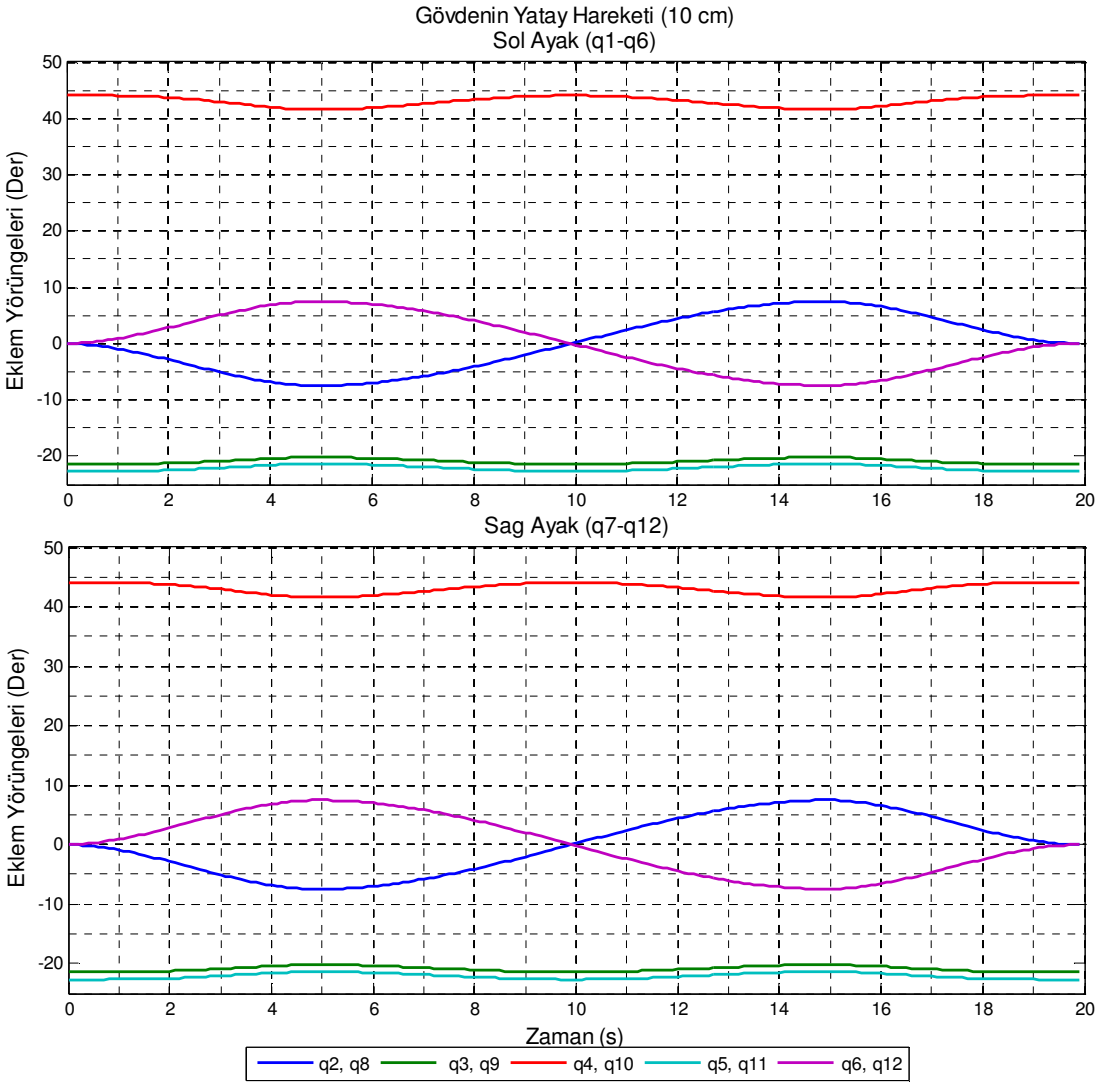
Şekil C.17 : Gövdenin 7 cm'lik yatay hareketinde ölçülen motor momentleri.

EK C.18



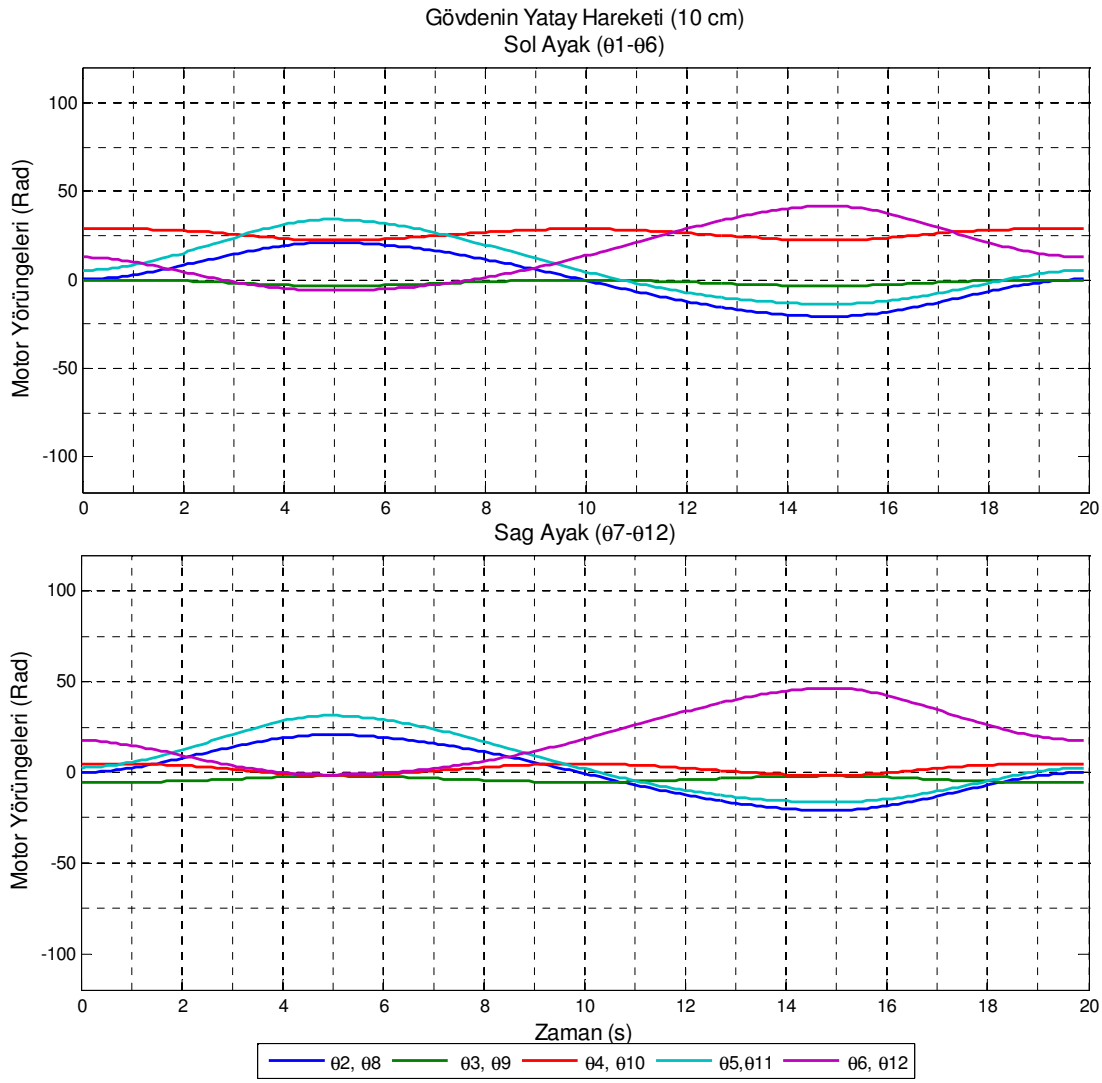
Şekil C.18 : Gövdenin 7 cm'lik yatay hareketindeki eklem momentleri.

EK C.19



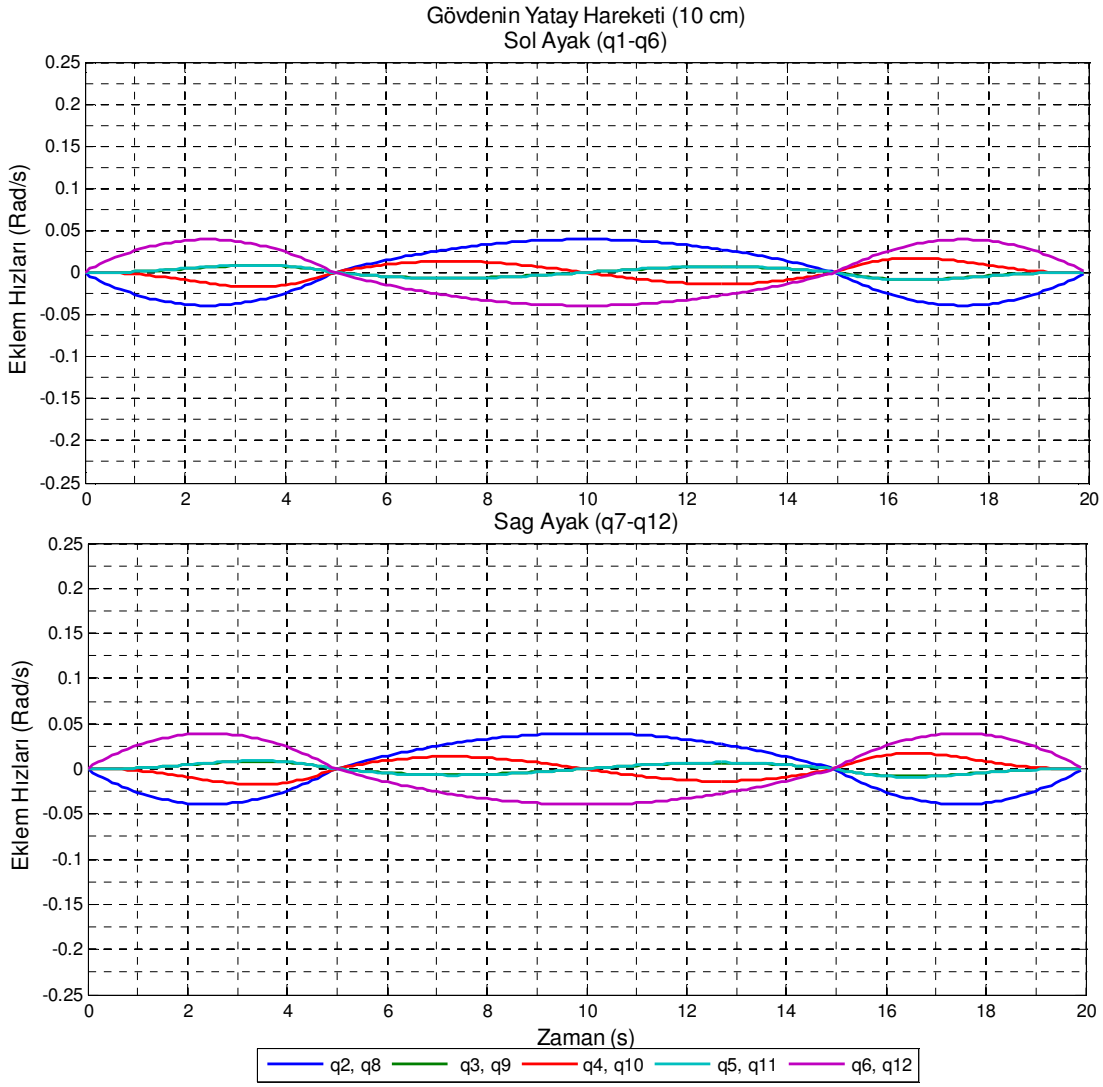
Şekil C.19 : Gövdenin 10 cm'lik yatay hareketi için referans eklem yörüngeleri.

EK C.20



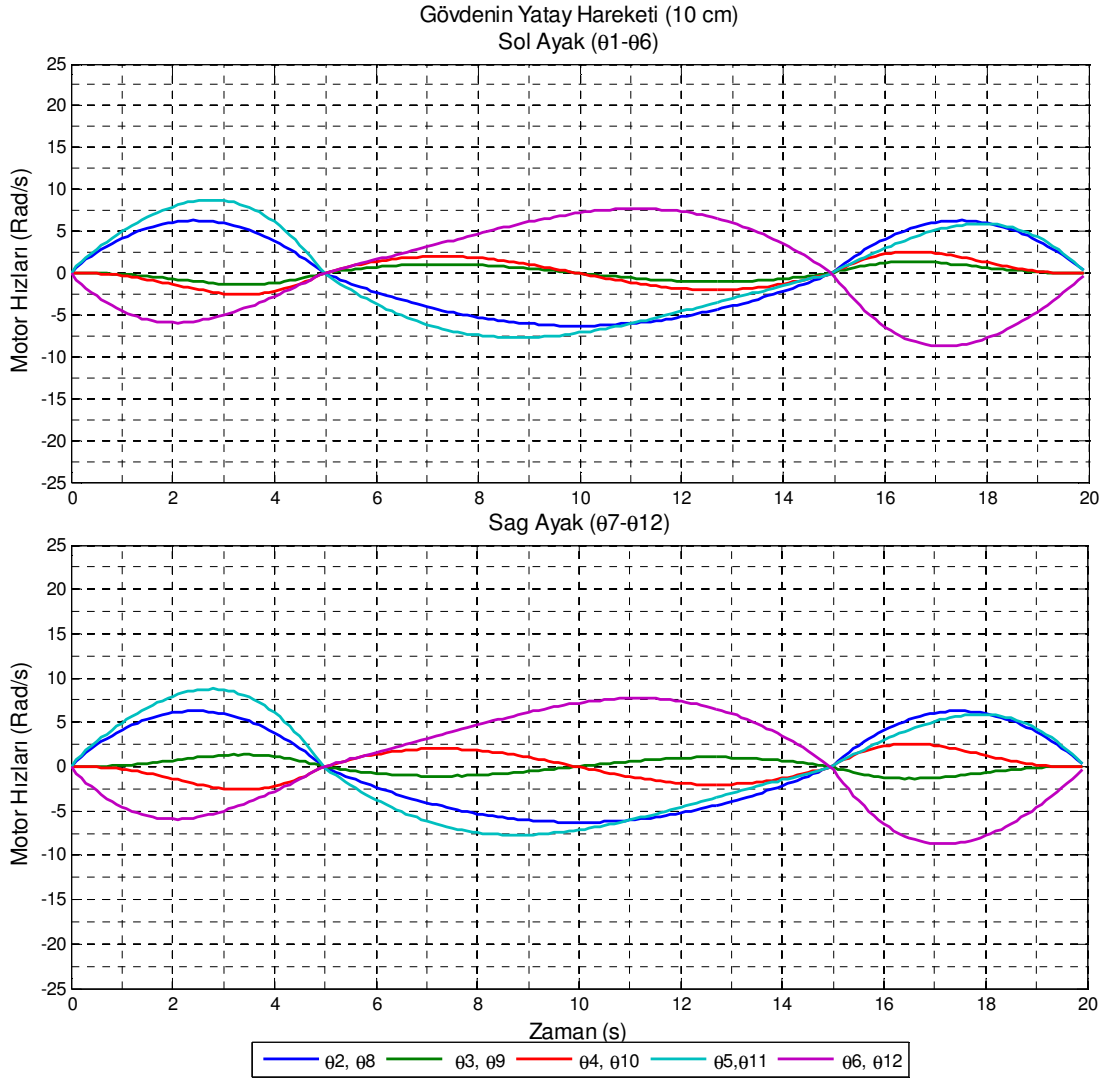
Şekil C.20 : Gövdenin 10 cm'lik yatay hareketi için referans motor yörüngeleri.

EK C.21



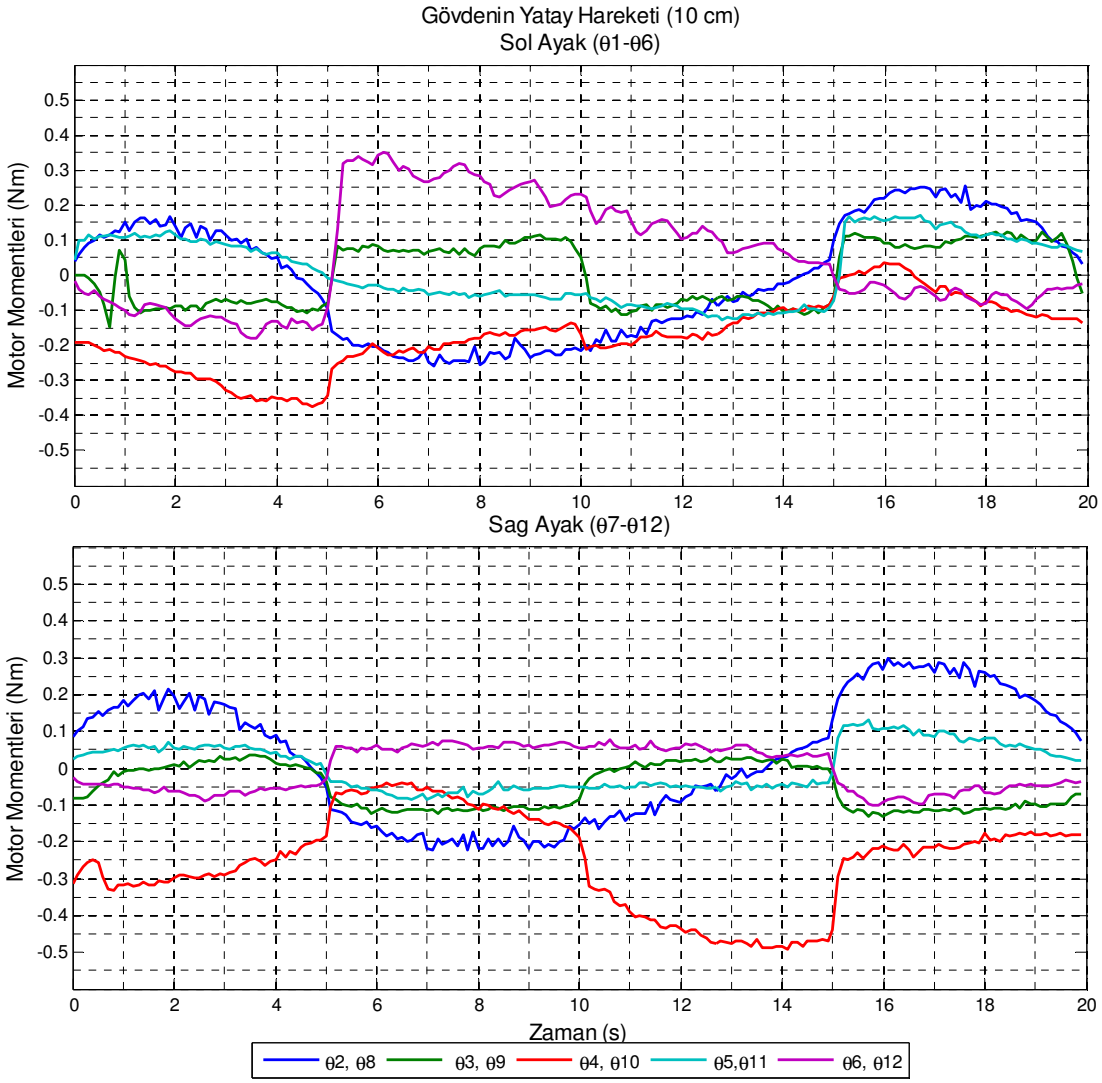
Şekil C.21 : Gövdenin 10 cm'lik yatay hareketi için referans eklem hızları.

EK C.22



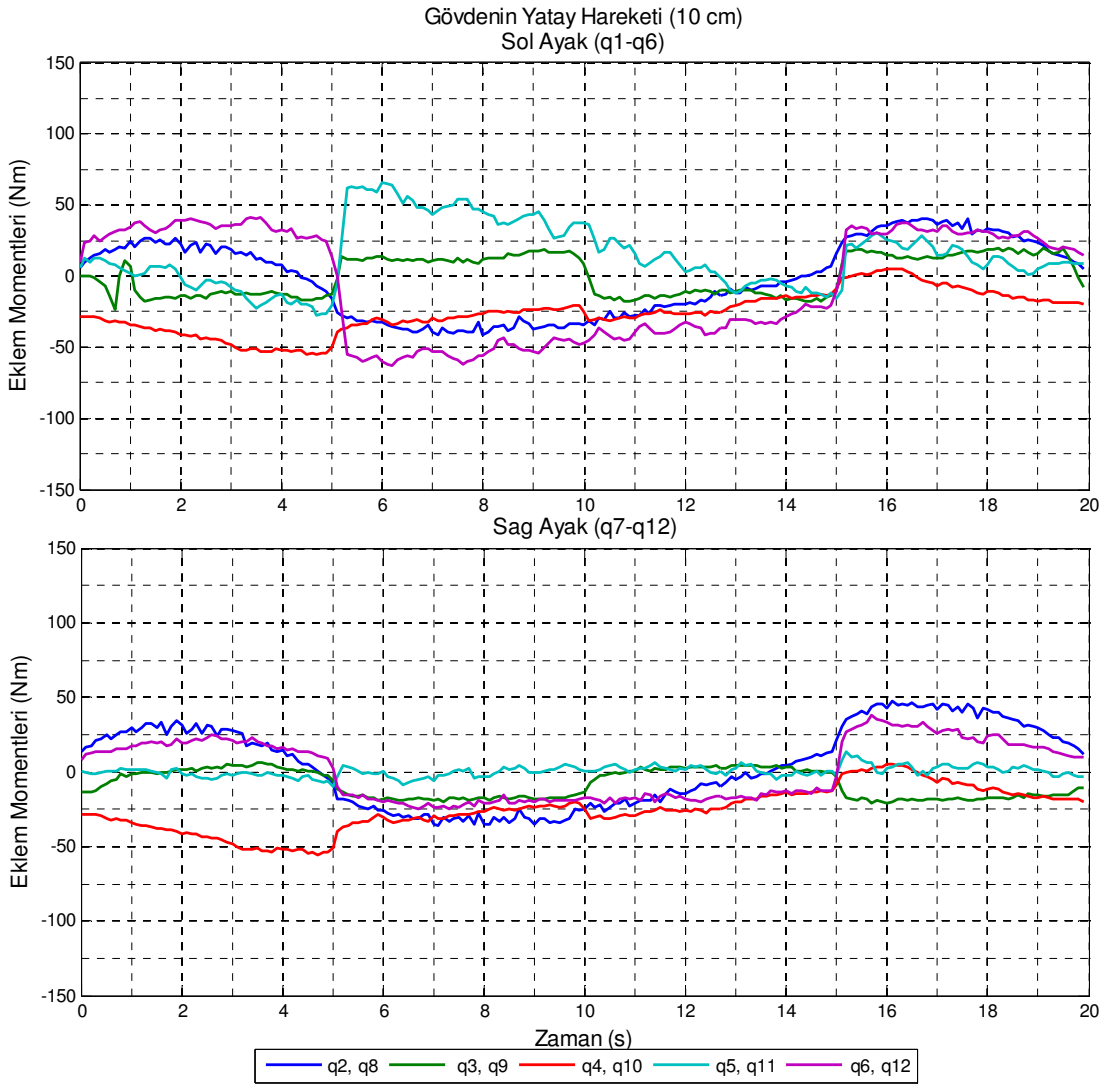
Şekil C.22 : Gövdenin 10 cm'lik yatay hareketi için referans motor hızları.

EK C.23



Şekil C.23 : Gövdenin 10 cm'lik yatay hareketinde ölçülen motor momentleri.

EK C.24



Şekil C.24 : Gövdenin 10 cm'lik yatay hareketindeki eklem momentleri.

ÖZGEÇMİŞ



Ad Soyad: Burak Tasasız

Doğum Yeri ve Tarihi: Üsküdar – 07/11/1986

Adres: Üsküdar/İST

Lisans Üniversite: İTÜ Uçak Mühendisliği

Yayın Listesi:

▪ **Acar, M., Gercek A., Tasasız, B., Bayraktaroglu, Z. Y., Kocabaş, H.,** 2010. Design of a 12-Dof Biped Robot, *MACRo'2010- International Conference on Recent Achievements in Mechatronics, Automation, Computer Science and Robotics*, Targu Mureş, Romania, 215 - 227.