T.C.
GEBZE YÜKSEK TEKNOLOJİ ENSTİTÜSÜ
MÜHENDİSLİK VE FEN BİLİMLERİ ENSTİTÜSÜ

5 SERBESTLİK DERECELİ DOKUNSAL GERİ BİLDİRİM CİHAZININ GERÇEK ZAMANLI LİNUX İŞLETİM SİSTEMİ ALTINDA UYGULANMASI VE DENETİMİ

CÜNEYT AY
YÜKSEK LİSANS TEZİ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

GEBZE
2014

T.C.
GEBZE YÜKSEK TEKNOLOJİ ENSTİTÜSÜ
MÜHENDİSLİK VE FEN BİLİMLERİ ENSTİTÜSÜ

5 SERBESTLİK DERECELİ DOKUNSAL GERİ BİLDİRİM CİHAZININ GERÇEK ZAMANLI LİNUX İŞLETİM SİSTEMİ ALTINDA UYGULANMASI VE DENETİMİ

CÜNEYT AY
YÜKSEK LİSANS TEZİ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

DANIŞMANI
DOÇ. DR. ERKAN ZERGEROĞLU

GEBZE
2014

T.R.

GEBZE INSTITUTE OF TECHNOLOGY

GRADUATE SCHOOL OF ENGINEERING AND SCIENCES

THE 5 DOF HAPTİC WAND CONTROL WITH ZENOM SIMULATION ENVIRONMENT

CÜNEYT AY

A THESIS SUBMITTED FOR THE DEGREE OF

MASTER OF SCIENCE

DEPARTMENT OF COMPUTER ENGINEERING

THESIS SUPERVISOR

ASSOC. PROF. DR. ERKAN ZERGEROĞLU

GEBZE

2014

GEBZE YÜKSEK TEKNOLOJİ ENSTİTÜSÜ	**YÜKSEK LİSANS JÜRİ ONAY FORMU**

GYTE Mühendislik ve Fen Bilimleri Enstitüsü Yönetim Kurulu'nun ……. tarih ve ..……/……..… sayılı kararıyla oluşturulan jüri tarafından …..…/…..…/..…… tarihinde tez savunma sınavı yapılan ……'ın tez çalışması ……Anabilim Dalında YÜKSEK LİSANS tezi olarak kabul edilmiştir.

JÜRİ

ÜYE
(TEZ DANIŞMANI) :

ÜYE :

ÜYE :

ONAY

GYTE Mühendislik ve Fen Bilimleri Enstitüsü Yönetim Kurulu'nun ……. tarih ve ……/…….. sayılı kararı.

İMZA/MÜHÜR

ÖZET

Bu çalışmada Quanser Inc. firmasının 5 DOF Haptic Wand cihazının gerçek zamanlı Linux işletim sistemi altında denetimi ve kontrolü yapılmıştır. 5 DOF Haptic Wand, 5 serbestlik derecesine sahip dokunsal geri bildirim verebilen bir cihazdır. Bu cihaz ile uygulama geliştirmek için döngüde donanım uygulamalarında etkili bir ara yüz sunan Zenom benzetim ortamı kullanılmıştır. Zenom, Xenomai yamalı Linux gerçek zamanlı işletim sistemi üzerinde çalışmaktadır. Ancak dokunsal geri bildirim cihazını üreten firma Linux platformuna destek vermemektedir. Bu yüzden cihaza ait sürücü ve uygulama programlama ara yüzü yapılmıştır. Bu bileşenler aracılığıyla Zenom ortamı ile cihaz haberleşmesi gerçek zamanlı olarak mümkün olmuştur. Bu iletişimin nasıl kurulacağına dair örnek teşkil etmesi için 5 DOF Haptic Wand cihazından veri okuyup, cihaza veri yazan örnek uygulamalar yapılmış ve açıklanmıştır.

Anahtar Kelimeler: Gerçek Zamanlı Benzetim, Xenomai, Zenom, Dokunsal Geri Bildirim.

SUMMARY

In this study, Quanser Inc.'s 5 DOF Haptic Wand products has been controlled on Linux based real-time operating system. 5 DOF Haptic Wand has five degree of freedom allowing for three translations and two rotations. To develop application with this device, Zenom's simulation environment which is an effective framework to develop hardware-in-the-loop simulations has been used. Zenom runs on Linux patched with Xenomai. The company that develops the haptic device does not support the Linux platform. Therefore, this thesis presents driver and application programming interface (API) for haptic programming which support a 5 DOF Haptic Wand device. A part of the thesis is a set of testing applications illustrating a basic use of these API.

Key Words: Real-Time Simulation, Xenomai, Zenom, 5 DOF Haptic Wand.

TEŞEKKÜR

Başta, yüksek lisans eğitimimde ve akademik hayatımda desteğini ve yardımlarını hiçbir zaman esirgemeyip bilgisi ile bu çalışmanın oluşmasının yolunu açan danışmanım Doç. Dr. Erkan ZERGEROĞLU'na,

Bütün çalışmam boyunca yanımda olan ve göstermiş olduğu desteklerinden dolayı Tuğçe İSAK'a en içten teşekkürlerimi sunarım.

İÇİNDEKİLER

SİMGELER ve KISALTMALAR DİZİNİ

Simgeler ve Kısaltmalar		**Açıklamalar**
Hz	:	Hertz
F	:	Kuvvet
m	:	Metre
mm		Milimetre
sn	:	Saniye
rad		Radyan
N	:	Newton
A	:	Amper
GYTE	:	Gebze Yüksek Teknoloji Enstitüsü
UPA	:	Uygulama Programlama Ara yüzü

ŞEKİLLER DİZİNİ

TABLOLAR DİZİNİ

1. GİRİŞ

Haptik teknolojisi dokunma ve kuvvet sentezi ile ilgilenir. Haptik cihaz içeren uygulamalar son yıllarda görme ve ses duyularımıza hitap eden uygulamalar kadar hızlı gelişme göstermiştir. Haptik teknolojisi kullanıcının haptik cihazla yapmış olduğu hareketleri işleyerek deri veya kas yoluyla algılayabildiği sanal dokunma hissi yaratır. Cerrahi operasyonlar gibi el ve göz koordinasyonu gerektiren ortamlar haptik cihazlar kullanarak benzetilir. Uçuş benzetimlerinde haptik cihazlar ile eğitimler verilir. Ayrıca bilgisayar oyunları ve mobil cihazlarda da önemli miktarda araştırma konusu olmuştur.

Haptik cihaz içeren benzetimlerde gerçekçi duygu alınmak isteniyorsa başarılı algoritmaların yanı sıra benzetimin gerçek zamanlı sistemler üzerinde çalışması gerekir. Aksi takdirde benzetimde insana gerçek dünyaya eş değer tepkiler verilmez. Bu da benzetimi gerçek dışı yapar. Gerçek zamanlı benzetimler gerçek zamanlı işletim sistemleri ile çalışır. QNX [14] ve VxWorks [15] yaygın olarak kullanılan gerçek zamanlı işletim sistemleridir. Ayrıca, günümüzde Linux işletim sistemi üzerine kurulan açık kaynak kodlu yamalar ile gerçek zamanlı uygulamalar çalıştırabilir hale getirilebilmektedir. Bunlardan yaygın olarak kullanılanların Xenomai [5] ve RTAI yamasıdır [16].

Xenomai dünyaca kabul görmüş Posix API ara yüzü ile uygulama geliştirmeye imkân sağlayan yapısı ile standart uygulamaların düşük bir maliyet ile gerçek zamanlı çalışabilir hale getirilmesini sağlamaktadır. Ayrıca daha yüksek performans sağlayan Xenomai API ile sıkı gerçek zaman kısıtlarını da sağlamaktadır. Bu sayede standart bir Linux işletim sistemi gerçek zamanlı uygulamaların çalıştırılabileceği bir sisteme dönüştürülebilmektedir.

Gerçek zamanlı sistemlerde robotik uygulamaları geliştirirken benzetim ortamlarına ihtiyaç duyulur. Robotik uygulamaları karmaşık modeller içerdiği için sistem dinamiklerinin ve uygun zaman aralığının bulunması gerekir. Bu parametreleri hızlı ve doğru bulmak için benzetim ortamlarının sunduğu imkânlardan yararlanılır. Sonuçlar iki boyutlu grafikler ile gösterilerek değerler yorumlanabilir ve hatalar ayıklanabilir. Ayrıca benzetim ortamları kullanıcıya karmaşık matematik işlemleri içeren kütüphaneler vererek benzetimin gerçeklenmesini kolaylaştırabilir.

RT-LAB, RTI-Lab, dSPACE ve MATLAB/Simulink günümüzde yaygın olarak kullanılan benzetim ortamlarıdır.

Biz de Linux işletim sistemini gerçek zamanlı işletim sistemine çeviren Xenomai yamasından faydalanarak, GYTE Kontrol Uygulamaları ve Robotik Laboratuvarı'nda Zenom olarak adlandırılan gerçek zamanlı bir benzetim ortamı geliştirmişdik [1]. Geliştirdiğimiz ortam değerleri iki boyutlu grafikte, sanal gerçeklik uygulamaları için üç boyutlu sahnede ve çeşitli ölçü pencerelerinde gösteren gelişmiş ara yüzlere sahiptir. Bu çalışmada da 5 DOF Haptic Wand cihazı Zenom benzetim ortamı kullanılarak kontrol edilmiştir.

1.1. Tezin Amacı, Katkısı ve İçeriği

Bu tez kapsamında Quanser Inc. firmasının 5-DOF Haptic Wand cihazının [6], [7] gerçek zamanlı Linux işletim sistemi üzerinde kontrol edilmesi amaçlanmıştır. Mevcut cihazın Windows işletim sistemi altında MATLAB/Simulink kullanarak kontrol edildiği örnekler bulunmaktadır. Windows işletim sistemi gerçek zamanlı bir işletim sistemi olmadığı için insan ile etkileşimde olan haptik cihaz uygulamalarında gerçekçi davranışlar sergilemez. Mevcut MATLAB/Simulink örnek uygulamaları gömülü bir sisteme aktarılarak gerçek zamanlı çalıştırılabilir. Ancak bunun için ekstra donanımsal bir cihaza ihtiyaç vardır. Bizim de amacımız maliyet yükü olmadan Linux işletim sistemini gerçek zamanlı bir işletim sistemine çevirerek 5-DOF Haptic Wand cihazını kontrol etmektir.

GYTE Kontrol Uygulamaları ve Robotik Laboratuvarı'nda 5-DOF Haptic Wand cihazı ile ileriye dönük yapılacak çalışmalar için cihazın kontrolünü kolaylaştıran uygulama programlama ara yüzü yapılması hedeflenmiştir. Ayrıca döngüde donanım benzetimlerini ve kontrol uygulamalarını Zenom ortamında geliştirirken izlenmesi gereken yol da anlatılmıştır.

Bu çalışmada, gerçek zamanlı sistem olarak Xenomai yaması yapılmış Linux işletim sistemi kullanılmıştır [5]. 5-DOF Haptic Wand cihazı ile uygulama geliştirmek için GYTE Kontrol Uygulamaları ve Robotik Laboratuvarı'nda yapılan Zenom benzetim ortamı kullanılmıştır [1].

Bu tez 6 bölümden oluşmaktadır. İlk bölümde genel giriş verilmiş ve tezin amacı sunulmuştur. Bölüm 2'de haptik teknolojileri ve gerçek zamanlı benzetimlere

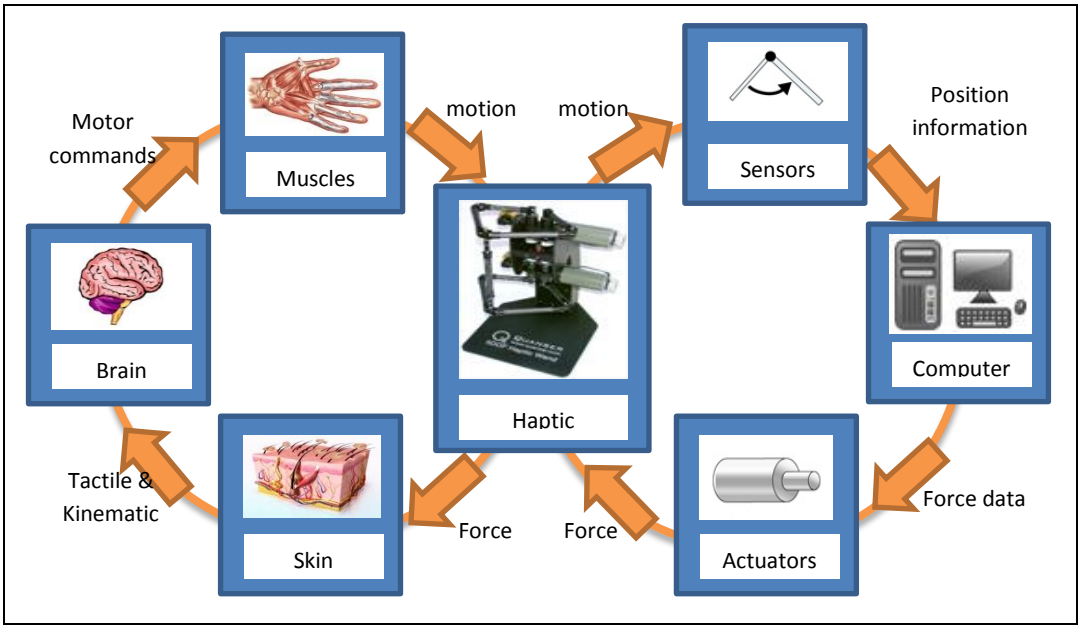
genel bakış, temel tanımlar anlatılmıştır. Döngüde donanım uygulamalarında niçin gerçek zamanlı sistemlere ihtiyaç duyulduğu tartışılmıştır. Üçüncü bölümde 5 DOF Haptic Wand cihazı hakkında genel bilgi verilmiş, teknik özellikleri ve yetenekleri anlatılmıştır. Quanser 5 DOF Haptic Wand cihazına ait sürücü ve programlama ara yüzü dördüncü bölümde anlatılmıştır. Beşinci bölümde örnek uygulamalar ile Zenom ortamı üzerinde cihazın nasıl kontrol edildiğinden bahsedilmiştir. Altıncı ve son bölümde elde edilen sistem ile ilgili sonuçlar yorumlanmıştır.

2. HAPTİK VE GERÇEK ZAMANLI SİSTEMLER

Haptik kelimesi Yunanca'dan gelmektedir ve dokunmak anlamına gelir (Oxford Dictionary). İlk olarak 19. yüzyılın sonlarında psikofizikçiler tarafından insanın dokunuşları üzerine yaptıkları çalışmalarda kullanılmıştır. Daha sonrasında büyüyen robotik çalışmalarıyla beraber kelimenin anlamı genişlemiştir.

Psikofizikçiler yıllarca dokunmayı ve hissetmeyi anlamaya çalıştılar. İnsanlar derideki sensörler, eklemler, tendonlar ve kaslar yardımıyla nesneleri algılarlar. Derideki sensörler ile sıcaklık, basınç, titreşim, sürtünme ve acı hissedilir. Kinestetik olarak yani eklem, kas ve tendonlar ile pozisyon, hareket ve kuvvet algılanır.

1970 ve 1980'lerde robotik araştırmalarında dokunarak algılamaya yönelik araştırmalar insanların ilgisi çekmiştir. Corliss & Johnson (1968) ve Mosher'in (1964) 10 serbestlik dereceli "Handyman" tasarımı ve Thring'in (1983) "Hardiman" tasarımı haptik sistemlerin ilk örneklerindendir [2], [3] [4]. Massie & Salisbury (1994) sanal nesneleri hissetmeyi mümkün kılan PHANToM ismini verdikleri haptik bir cihaz tasarlamıştır [9]. Srinivasan & Başdoğan'da (1997) haptik cihazlar ile sanal ortamların birleştirildiği çalışmalar ortaya koymuştur [10]. Bu gelişmeler bilgisayarla haptiğin ilk adımları olmuştur. Ayrıca bu keşifler karmaşık haptik teknolojileri için motivasyon kaynağı haline de gelmiştir.



Şekil 2.1: Bilgisayar ile haptikte bilgi akışı

Srinivasan & Başdoğan (1997) insanın gerçek dünya algısı ile bilgisayarla haptiğin arasındaki bilgi akışını Şekil 2.1' de ortaya koymuştur [10]. Bu akış şeması insanın hissetme döngüsü ve bilgisayarla haptiğin döngüsünden oluşur. İnsan gerçek veya sanal nesnelere dokunduğu zaman deride bulunan alıcılar uyarılır. Alıcılardan bilgiler beyine iletilir. Beyinde kasları harekete geçiren komutları yollar. Bilgisayarla haptik döngüsünde ise insan haptik cihaz ile temasa geçtiğinde cihazın pozisyonu sensörler aracılığı ile bilgisayara aktarılır. Bilgisayar motorlara uygulaması gereken tork hesaplamalarını yapar ve haptik cihaza gönderir. Haptik cihazda motorlar aracılığı ile insana kuvvet uygulayarak insan haptik cihaz etkileşim döngüsü tamamlar.

Haptik sistemlerin önemli kullanım alanlarında biri de sanal ortamlar oluşturarak eğitim amaçlı platformlar oluşturmaktır. İnsan bir haptik cihaz tutarak veya giyerek gerçeği taklit eden veya hayali sahnelerle eğitime tabii tutulabilir. Haptik cihaz uçuş benzetimlerinde levyeyi veya diş temizleme benzetimlerinde kavitronu temsil edebilir. Bu eğitimler ile farklı koşullar oluşturularak yapılan egzersizler sayesinde kişiye tecrübe kazandırılabilir. Ayrıca oluşturulan zor senaryolarda bir hata yapıldığı zaman herhangi bir zararda meydana gelmez.

Temel haptik uygulamaları genellikle çarpışma tespiti, kuvvet geri bildirim algoritması ve kontrol algoritması olmak üzere üç ana bloktan oluşur. Çarpışma tespit algoritması, insanın haptik cihaz kullanarak hareket ettirdiği sanal dünyadaki şekil ile sanal ortamdaki diğer nesneler arasında çarpışmaların nerede olduğunu ve ölçüsünü tespit eder. Kuvvet geri bildirim algoritması çarpışma tespit edildiği zaman insana uygulanması gereken kuvvetin büyüklüğünü hesaplar. Bu kuvvet mümkün olduğunca gerçek nesneye dokununca hissedilen kuvvete eş değerdir. Genellikle bu algoritmalar kuvvet veya tork vektörleri döndürür. Kontrol algoritmaları cihaz eklemlerinden konum bilgilerini alır. Her eklemden alınan konum bilgileri birleştirilerek dünya koordinat sistemindeki pozisyonu bulur. Çarpışma tespit algoritmasına pozisyonu iletir ve kuvvet geri bildirim algoritmasından aldığı vektörler ile haptik cihazı kontrol ederek insana kuvveti iletir.

Haptik cihaz uygulamaları genellikle döngüde donanım tekniği ile kontrol edilir. Döngüde donanım tekniği sayesinde fiziksel aygıtlar ile sanal cihazlar yer değiştirir. Bu değişim maliyeti azaltır, sistemi farklı koşullarda test etme imkânı sunar, sistemi daha kararlı yapar. Ayrıca tehlikeli şartları güvenli bir şekilde oluşturmak mümkündür.

Döngüde donanım tekniği gerçek zamanlı benzetimleri ile mümkündür. Gerçek zamanlı benzetimler fiziksel bir sistemin duvar saati zamanı ile aynı oranda bir bilgisayar modeli ile ifade edilmesidir. Gerçek zamanlı benzetimlerin katı kuralı gerçek dünya ile senkron çalışma zorunluluğu olmasıdır. Başka bir deyişle zamana karşı bir yarıştır. Eğer belirlenen zaman aralığında hesaplamalar tamamlanmaz ise yarış kaybedilir, benzetim gerçek dışı davranış sergiler.

Haptik cihaz içeren benzetimlerde insana gerçekçi yanıtlar verilmek isteniyorsa başarılı modellerin yanı sıra uygulamanın gerçek zamanlı sistemler üzerinde çalışması gerekir. Çünkü benzetimde insan haptik cihaz ile etkileşime girdiğinde benzetim gerçek dünya ile aynı tepkileri insana vermelidir.

İnsan etkileşimi olan benzetimler gerçek zamanlı benzetimler gerektirir. Örneğin pilot eğitimi için tasarlanan bir uçuş benzetimi gerçek zamanlı benzetim olmak zorundadır. Pilot levye şeklindeki haptik cihaz ile benzetimde uçağı kontrol ettiğini varsayalım. Benzetimde levye şeklindeki cihazdan alınan geri beslemeler gerçek dünya ile eş değer olması gereklidir. Aksi takdirde eğitim gerçek dışı olur. Geri beslemeleri gerçek dünya ile eş değer yapmanın yolu gerçek zamanlı sistemler üzerinde döngüde donanım tekniğini kullanmaktır.

Gerçek zamanlı benzetim oluşturmak sıradan benzetimlere göre daha zordur. Çünkü zamana karşı hassas olduğu için katı kısıtları vardır. Benzetimdeki bütün görevler belirlenen zaman aralığında tamamlanmalıdır. Benzetim zaman aralığı, model ile hesap üretecek, sensörlerden veri okuyacak ve eylemcilere sinyal yazacak kadar yeterli olmalıdır. Ayrıca zaman aralığı benzetimde yanlış sonuçlar üretecek kadarda yüksek olmamalıdır.

Benzetim zaman aralığının yeterli olmadığı durumlarda doğru sistem dinamiklerini yakalamak ve hesaplama miktarını en aza indirmek gerekir. Bu gibi durumlarda gerçek zaman kriterini karşılayan zaman aralığı ve sistem dinamikleri için doğru kombinasyonu bulmak gerekir. Bu kombinasyonda genellikle deneme yanılma ile yakalanır.

Benzetim performansı gerçek zamanlı sistemde kabul edilemez ise nedenleri belirlemek ve uygun bir çözüm bulmak gerekir. Sistem dinamikleri ve zaman aralığı kombinasyonlarına alternatif olarak daha hızlı bir bilgisayar kullanılarak benzetim performansı arttırılabilir. Mümkünse benzetim modelleri ayrıştırılarak birden fazla bilgisayarda çalıştırılıp paralelleştirilebilir. Hızlı ve ani değişikliklere neden olan değişkenler gözden geçirilebilir. Karmaşık eşitlikler içeren denklemleri çalışma

zamanında hesaplamak yerine daha önceden hesaplanmış sonuçlar ile doldurulmuş bir tablodan çekilebilir [13].

Katı bir kural olmasa da haptik sistemlerin kontrol döngüsü en az 1 kHz oranında olmalıdır. Başka bir deyişle 1 ms'den daha kısa süreden bütün görevler tamamlanmalıdır. Bu orandan yüksek değerler daha keskin temas ve doku hissi sağlayabilir. Ancak çoğu uygulama için 1 kHz yeterli bir orandır.

3. QUANSER 5-DOF HAPTIC WAND

Bu bölümde çalışmada kullanılan Quanser Inc. firmasına ait 5-DOF Haptic Wand cihazı anlatılmıştır. 5-DOF Haptic Wand cihazı Profesör Tim Salcudean tarafından Britanya Kolombiya Üniversitesi'nde haptik cihazlar üzerine araştırmalar yapmak için tasarlanmıştır. 5-DOF Haptic Wand ticari amaç güden bir araştırma aracıdır. Haptik ve tele operasyon uygulamalarında kullanılan bir üründür. 5-DOF Haptic Wand Şekil 3.1'de gösterilmiştir.

Şekil 3.1: Quanser 5-DOF Haptic Wand cihazı

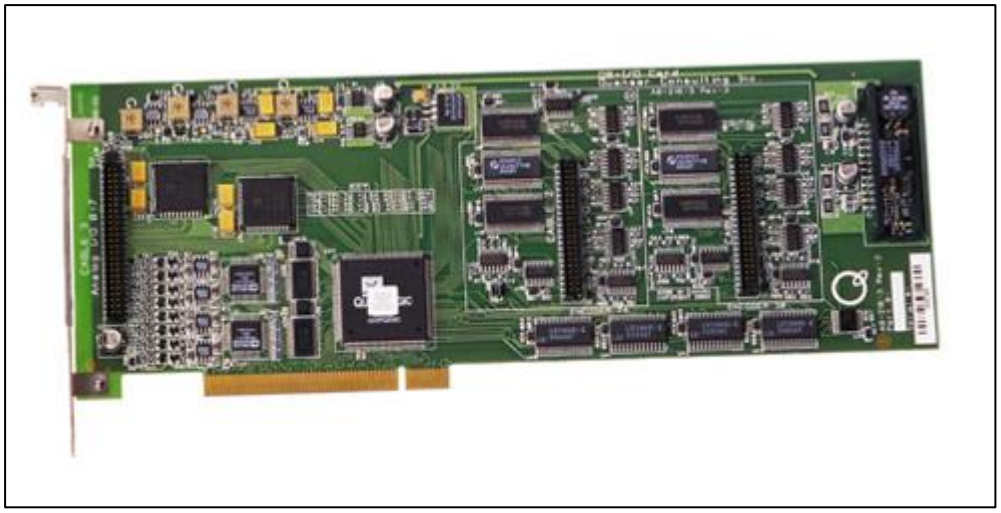
5-DOF Haptic Wand robotik uygulamalarında özellikle haptik uygulamalarda kullanılan bir cihazdır. Cihaz isminden de anlaşıldığı üzere üç öteleme (x, y, z) ve iki dönme (yuvarlanma (ing: roll), yunuslama(ing: pitch)) hareketi olmak üzere 5 serbestlik derecesine sahiptir. Çubuk ile sağa sola sapma (ing: yaw) hareketi yapılamaz. Haptik cihaz silindir şeklindeki bir sonlandırıcı çubuk ile birbirine bağlanmış iki pantograftan oluşmaktadır. Her pantografın yan yana iki DC motoru ve omuzda bir DC motoru vardır. Omuz kısmında olan motorlar daha güçlüdür. Pantograflara sonlandırıcı çubuk Universal eklem (ing: U-joint) ile birbirlerine bağlanmıştır. Çubuk hafif malzemeden yapılmış olup, ağırlığını değiştirebilmek için iki tane ayarlanabilir silindir kütle bulunur.

Cihaza altı motor güç sağlar. Motorlarının pozisyonları yüksek çözünürlüklere sahip optik pozisyon kodlayıcılar(ing: encoder) kullanılarak ölçülür. Cihaz Quanser firmasının Q8 HIL kontrol kartıyla kişisel bilgisayara bağlanarak kontrol edilebilir.

5-DOF Wand sisteminin MATLAB/Simulink üzerinde örnek uygulamaları bulunmaktadır. Ayrıca Quanser Inc. Firmasının kontrol uygulamaları için QuaRC kütüphanesi bulunmaktadır. Bu kütüphane Q8 HIL kontrol kartı ve MATLAB/Simulink kontrol uygulamaları için programlama ara yüzü sunar. QuaRC kütüphanesi C, C++, Java, .NET ortamlarında da kontrol uygulamaları geliştirme imkanı sunar. Ayrıca sistemin kinematik ve dinamik modellerin yanı sıra sistem parametrelerini de sağlamaktadır. Kütüphane katı gerçek zamanlı uygulamalar için QNX Neutrino işletim sistemi üzerinde, yumuşak gerçek zamanlı uygulamalar için ise Windows tabanlı işletim sistemlerine destek vermektedir. Linux tabanlı işletim sistemleri desteklenmemektedir.

3.1. Q8 Hil Kontrol Kartı

5-DOF Haptic Wand cihazının güç yükselteçleri ve ikiz pantografları tamamen Q8 HIL kontrol kartıyla uyumlu olacak şekilde tasarlanmıştır. Bu sayede 5-DOF Haptic Wand cihazı bu kart aracılığıyla kişisel bilgisayara bağlanarak kontrol edilebilir. Q8 HIL kartı Şekil 3.2'de gösterilmiştir.



Şekil 3.2: Q8 Hil kontrol kartı

Kart gerçek zamanlı sistemlerde önde gelen QuaRC, xPC ve RT-Lab sistemlerini desteklemektedir. Xenomai tabanlı Linux sistemini desteklememektedir. Ancak mimarisi açık olduğu için GYTE Kontrol Uygulamaları ve Robotik Laboratuvarı'nda sürücüsü yapılmıştır [17]. Q8 HIL kontrol kartı ile ilgili ayrıntılar için kullanıcı dokümanına bakınız [11].

3.2. Sistem Parametreleri

Cihazın çalışma uzayı sırasıyla x, y, z ekseni için 480mm x 250mm x 450mm'dir. Kalibrasyon yapıldıktan sonra sonlandırıcının konumu [0, 124, 0] mm'dir. Y ekseninin sıfır olmamasının sebebi kalibrasyon için kullanılan parçadan kaynaklanır. Dönme hareketinde ise yuvarlanma derecesi ±85º, yunuslama derecesi ±65º'dir. Tablo 3.1'de Quanser 5-DOF Haptic Wand cihazına ait çalışma uzayının sınırları verilmiştir.

x ekseni	±240 mm
y ekseni	85 – 335 mm
z ekseni	-215 – 235 mm
yuvarlanma	±85º
yunuslama	±65º

Tablo 3.1: Quanser 5-DOF Haptic Wand çalışma uzayı.

Quanser 5-DOf Haptic Wand cihazı ile sürekli olarak uygulanabilen kuvvet ve tork değerleri Tablo 3.2'de verilmiştir. Bu değerler haptik operasyonlardaki sertlik ve sürtünme için gerçeğe yakın bir his uyandırdığı için önemlidir.

x ekseni	2.3 N
y ekseni	2.1 N
z ekseni	3.0 N
yuvarlanma	230 N.mm
yunuslama	250 N.mm

Tablo 3.2: Quanser 5-DOF Haptic Wand kuvvet ve tork değerleri.

5-DOF Haptic Wand cihazı kontrol edilirken kademeli olarak kuvvete limit uygular. Bu kademenin omuzdaki motorlar için parametreleri; tepe güç limiti 7.0 A, tepe güç limit süresi 0.2 sn, sürekli güç limiti 2.15 A, sürekli güç limit süresi 10.0 sn'dir. Bunun anlamı cihazla 0.2 sn boyunca 7.0 A'ya eşdeğer bir kuvvet uygulanabilir demektir. 0.2 sn'den sonra 10 sn boyunca maksimum 2.15 A'e eşdeğer güç uygulanabilir. Benzer şekilde diğer motorlar için parametreler tepe güç limiti 5.0 A, tepe güç limit süresi 0.2 sn, sürekli güç limiti 1.69 A, sürekli güç limit süresi 10.0 sn'dir.

Sertlik (ing: stiffness) ve sönümleme (ing: damping) değerleri kontrol uygulamalarında geri bildirim olarak uygulanacak kuvvet hesaplamalarında kullanılan kat sayılardır. Tablo 3.3'de bu tez kapsamında kullanılan kat sayılar verilmiştir. kp değerleri sertliği, kd değerleri sönümlemeyi ifade etmektedir.

Parametre	x	y	z	pitch	roll
kp (N/m)	500	500	500	1	1
kd (N.s/m, N.s/rad)	5	5	5	0.1	0.1

Tablo 3.3: Sertlik ve sönümleme kat sayıları.

Cihaza ait daha detaylı teknik bilgiler ve parametreler Quanser 5-DOF Haptic Wand referans kılavuzunda bahsedilmektedir [7].

3.3. Kinematik Model

5-DOF Haptic Wand cihazının sağa sola sapma (ing: yaw) hareketi pasif olduğu için çubuğun konum vektörü olan X eşitlik (3.1)'deki gibi olur.

$$X = [x, \quad y, \quad z, \quad pitch, \quad roll\,] \tag{3.1}$$

Lineer yer değiştirme x, y, z metre olarak ifade edilir. $pitch$ ve $roll$ açıları radyan olarak ifade edilir. $roll$ x ekseni etrafında yapılan, $pitch$ y ekseni etrafında yapılan açıdır. Açılar saat yönünün tersine pozitiftir. Şekil 3.3'de 5-DOF Haptic Wand cihazının referans koordinat sistemi gösterilmiştir.

Şekil 3.3: Quanser 5-DOF Haptic Wand koordinat sistemi

5-DOF Haptic Wand cihazı kalibre edildiği zamanki çubuğun X vektörü eşitlik (3.2)'deki gibi olur. Y ekseninin sıfır olmamasının sebebi kalibrasyon için kullanılan parçadır. Çubuk bu parçaya tutturulduğu için 0.124 metrelik bir paya sebep olur.

$$X = [\ 0, \quad 0.124, \quad 0, \quad 0, \quad 0\ ] \tag{3.2}$$

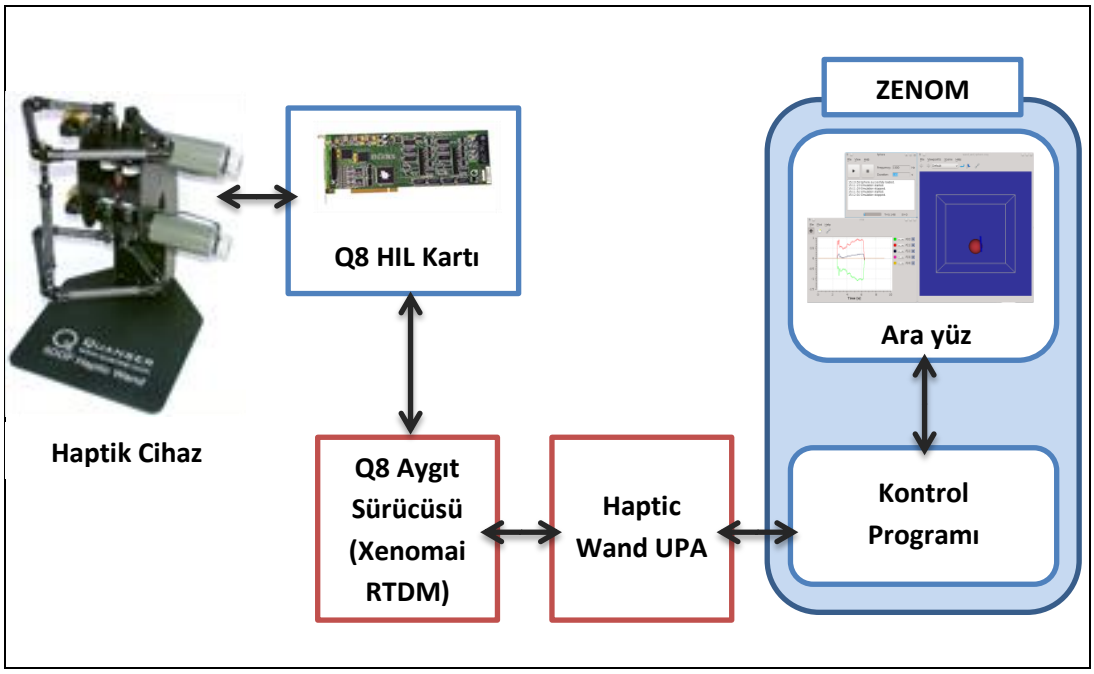
5-DOF Wand cihazına ait ileri kinematik, ters kinematik ve hız kinematik denklemleri Maple dokümanında verilmektedir [8]. Maple dokümanı model denklemlerini C ve MATLAB kodu olarak üretir. Bu çalışmadaki HapticWand uygulama programlama ara yüzü bu Maple dokümanı ile üretilen C kodu kullanılarak yapılmıştır.

4. QUANSER 5-DOF HAPTIC WAND KONTROLÜ

Bu bölümde Q8 HIL kontrol kartına erişmek ve 5-DOF Haptic Wand cihazının gerçek zamanlı kontrolünün yapıldığı sistem anlatılmıştır.

Döngüde donanım uygulamalarında insan faktörünün girmesiyle gerçek zamanlı sistemler üzerinde çalışması gerektiğini vurgulanmıştı. Tasarlanan sistem Şekil 4.1'de gösterilmiştir. Tasarımda gerçek zamanlı işletim sistemi olarak Xenomai yamalanmış Linux kullanılmıştır. Gerçek zamanlı benzetim ortamları iki boyutlu grafik gösterimi, üç boyutlu görselleme ve veri aktarma yetenekleri ile döngüde donanım uygulamalarının test edilmesi ve geliştirmesinde önemli katkılar sağlar. Bu yüzden bu çalışmada Zenom benzetim ortamı kullanılmıştır.

Q8 HIL kontrol kartına erişmek için Xenomai Real-Time Driver Module (RTDM) arabirimi kullanılarak gerçek zamanlı Q8 aygıt sürücüsü yapılmıştır. 5 DOF Haptic Wand cihazına ait ileri ve ters kinematik algoritmalarını içeren Haptic Wand uygulama programlama ara yüzü (UPA) yapılmıştır. Haptic Wand uygulama programlama kütüphanesi Zenom kontrol uygulamasında kullanılarak cihaz gerçek zamanlı kontrol edilmiştir.



Şekil 4.1: 5-DOF Haptic Wand cihazının kontrolü

4.1. Q8 Aygıt Sürücüsü

Bir uygulamanın bir donanım ile iletişim kuracağı en düşük tabaka işletim sisteminin sürücüsüdür. Genellikle bu katmandan alınan ham bilgiler işlenerek kullanılır. Haptik bir cihaz için bu katmanda cihaz açma ve kapatma, pozisyon kolayıcılardan sayım değeri okuma, analog kanallardan voltaj değeri okuma veya yazma işlemleri yapılır.

5-DOF Haptic Wand cihazının kullandığı Q8 HIL kontrol kartının Linux işletim sistemi üzerinde sürücüsü yoktur. Ayrıca cihaz gerçek zamanlı bir sistemde kontrol edileceği için aygıt sürücüsü de bu kritere göre gerçeklenmelidir. Xenomai Real-Time Driver Model ile gerçek zamanlı Linux altında aygıt sürücüleri için de bir yaklaşım sunar [12]. Xenomai, Real-Time Driver Model (RTDM) yaklaşımı ile Xenomai, sürücü katmanını da gerçek zamanlı sistemlerine dâhil etmişlerdir.

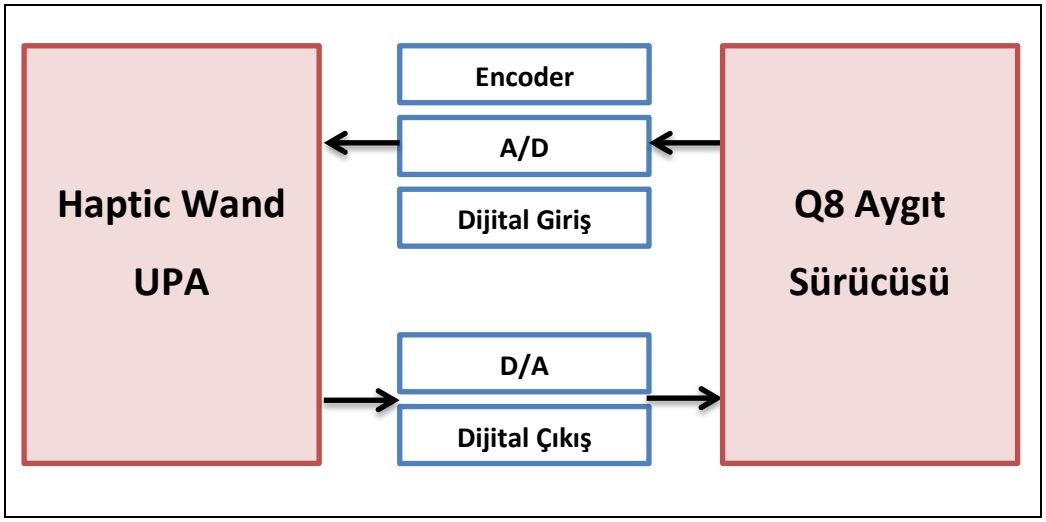
GYTE Kontrol Uygulamaları ve Robotik Laboratuvarı'nda Xenomai yamasının RTDM yaklaşımı kullanılarak Q8 HIL kartına uygun sürücü yazılmıştır [17]. Cihaz kontolünde bu sürücü kullanılmıştır.

4.2. HapticWand Uygulama Programlama Ara Yüzü

Sürücü tabakasında çalışmak en hızlı ve en hassas tepki sağlar. Ancak buradaki operasyonlar ham veri üzerinden olur. Haptik uygulamalarında cihazdan alınan ham veriler ileri kinematik algoritmaları ile işlenerek pozisyon bilgisine dönüştürülmesi ve kuvvet değerlerinin ters kinematik kullanılarak voltaj değerlerine dönüştürülmesi gereklidir. Bu yüzden kod yazarken oluşabilecek zorlukları engellemek, aynı kodları tekrar tekrar yazmamak, okunabilir ve anlaşılabilir bir program yazmak için HapticWand programlama ara yüzü tasarlanmıştır. Ara yüz c++ programlama dilini desteklemektedir.

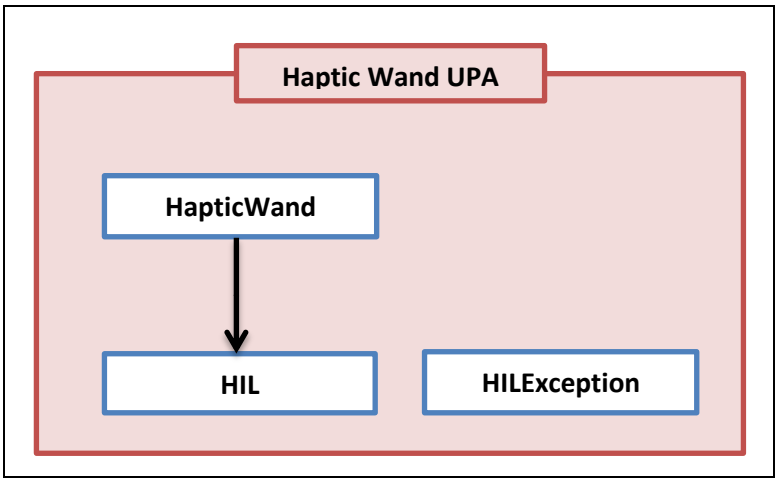
HapticWand ara yüzü 5 DOF Haptic Wand cihazı kontrolü için ihtiyaç duyulan fonksiyonları sağlar (Şekil 4.2). Bu fonksiyonlar temel düzeyde dijital giriş ve çıkışlarına ulaşma, pozisyon kodlayıcı girişlerinden sayım değerlerini okuma, analog giriş ve çıkış değerlerine erişmektir. Ayrıca bu ara yüz ile ileri düzey 5 DOF Haptic Wand cihazına ait ileri ve ters kinematik algoritmalarını da içinde barındırır. 5 DOF Haptic Wand cihazını kolayca yapılandırabilir ve çeşitli şekillerde cihazdan veri

okuyabilir ve cihaza veri yazabilirsiniz. İleri kinematik algoritması ile direk pozisyon bilgisini elde edebilir, ters kinematik fonksiyonu ile kuvvet değerlerini voltaja çevirebilirsiniz.



Şekil 4.2: Haptic Wand UPA ile Q8 aygıt sürücüsü haberleşmesi

HapticWand ara yüzü *HIL* ve *HapticWand* olmak üzere iki sınıf içerir. HIL sınıfı daha düşük seviyede Q8 HIL kontrol kartına erişim fonksiyonları içerir. HapticWand sınıfı ise 5 DOF Haptic Wand cihazına göre özelleşmiş fonksiyonlar içerir. 5 DOF Haptic Wand cihazının boyutlarını ve parametrelerini içeren HapticWand sınıfının ileri ve ters kinematik algoritmaları içeren fonksiyonlar vardır.



Şekil 4.3: Haptic Wand UPA mimarisi

4.2.1. Düşük Seviyeli Hil Sınıfı

Düşük seviyeli *Hil* sınıfı Xenomai RTDM kullanarak sürücüyle direk iletişime geçen sınıftır. Bu sınıf ile Q8 Hil kontrol kartının giriş ve çıkış kanallarına ulaşabilir, kartı yapılandırabilirsiniz. Dijital giriş ve çıkışlarına ulaşabilir, pozisyon kodlayıcılardan sayım değerlerini okuyabilir, analog giriş ve çıkış kanallarının voltaj değerlerine erişebilirsiniz. HIL sınıfı Şekil 4.4'de verilmiştir.

```
class HIL {
  public:
    HIL();

    virtual void open(const char* cardType = "Q80", const int
cardIdentifier = 0);

    virtual void close();

    double readAnalog( const unsigned int channelNumber );

    void readAnalog( const unsigned int* channelNumbers, double* voltages,
const int channelNumbersLength);

    void writeAnalog( const unsigned int channelNumber, const double
voltage );

    void writeAnalog( const unsigned int* channelNumbers, const double*
voltages, const int channelNumbersLength );

    int readEncoder( const unsigned int channelNumber );

    void readEncoder( const unsigned int* channelNumbers, int* counts,
const int channelNumbersLength  );

    void resetEncoder( const unsigned int channelNumber );

    void resetEncoder( const unsigned int* channelNumbers, const int
channelNumbersLength );

    bool readDigital( const unsigned int channelNumber );

    void readDigital( const unsigned int* inputChannels, bool* bits, const
int inputChannelsLength);

    void setDigitalOutputDirection( const unsigned int *outputChannels,
const int outputChannelsLength );

    void writeDigital( const unsigned int* outputChannels, const bool*
bits, const int outputChannelsLength);


    ...


};
```

Şekil 4.4: HIL sınıfı

Hil sınıfının fonksiyon listesi aşağıda verilmiştir:

- `void open(const char* cardType, const int cardIdentifier)`

Bu fonksiyon döngüde donanım kartını açar. Donanım ile bilgi alış-veriş fonksiyonları çağrılmadan önce muhakkak bu fonksiyon ile kart açılmalıdır. Ayrıca kart ile iş bittiğinde `close` fonksiyonu ile kapatılması gerekir. `cardType` parametresi kartın tipi, `cardIdentifier` parametresi ile kartın tanımlayıcısıdır.

- `void close()`

Bu fonksiyon kartı kapatır. Parametresi yoktur.

- `double readAnalog( const unsigned int channelNumber )`

Belirtilen analog giriş kanalındaki voltaj değerini okur. `channelNumber` parametresi voltaj değeri okunacak analog kanal numarasıdır. Fonksiyon geriye analog kanalındaki voltaj değerini getirir. Analog kanal numaraları 0-7 arasında değişmektedir.

- `void readAnalog( const unsigned int* channelNumbers, double* voltages, const int channelNumbersLength)`

Bu fonksiyon analog giriş kanallarındaki voltaj değerlerini okur. Yukarıdaki `readAnalog` fonksiyonundan farkı aynı andan birden fazla kanaldaki voltaj değerini okumasıdır. `channelNumbers` parametresi değerleri okunacak kanal numaraları dizisi, `voltages` parametresi okunan değerlerin yazılacağı dizi, `channelNumbersLength` parametresi dizinin boyutudur. Analog kanal numaraları 0-7 arasında değişmektedir.

- `void writeAnalog( const unsigned int channelNumber, const double voltage)`

Belirtilen analog çıkış kanalına tanımlanan voltaj değerini yazar. `channelNumber` parametresi voltaj değeri yazılacak analog kanal numarası, `voltage` parametresi kanala yazılacak voltaj değeridir. Analog kanal numaraları 0-7 arasında değişmektedir.

• `void writeAnalog( const unsigned int* channelNumbers, const double* voltages, const int channelNumbersLength )`

Bu fonksiyon analog çıkış kanallarına tanımlanan voltaj değerlerini yazar. `channelNumber` parametresi voltaj değeri yazılacak analog kanal numaraları dizisi, `voltages` parametresi kanala yazılacak voltaj değerleri dizisi, `channelNumbersLength` parametresi dizinin boyutudur. Analog kanal numaraları 0-7 arasında değişmektedir.

• `int readEncoder( const unsigned int channelNumber )`

Belirtilen pozisyon kodlayıcı kanalındaki sayım değerini okur. `channelNumber` parametresi sayım değeri okunacak pozisyon kodlayıcı kanal numarasıdır. Fonksiyon geriye pozisyon kodlayıcı kanalındaki sayım değerini getirir. Pozisyon kodlayıcılarının kanal numaraları 0-7 arasında değişmektedir.

• `void readEncoder( const unsigned int* channelNumbers, int* counts, const int channelNumbersLength )`

Bu fonksiyon pozisyon kodlayıcı kanallarındaki sayım değerlerini okur. `channelNumber` parametresi sayım değeri okunacak pozisyon kodlayıcı kanal numaraları dizisi, `counts` parametresi okunan değerlerin yazılacağı dizi, `channelNumbersLength` parametresi dizinin boyutudur. Pozisyon kodlayıcılarının kanal numaraları 0-7 arasında değişmektedir.

• `void resetEncoder( const unsigned int channelNumber)`

Belirtilen pozisyon kodlayıcı kanalını sıfırlar. `channelNumber` parametresi sıfırlanacak pozisyon kodlayıcı kanal numarasıdır. Pozisyon kodlayıcılarının kanal numaraları 0-7 arasında değişmektedir.

• `void resetEncoder( const unsigned int* channelNumbers, const int channelNumbersLength )`

Bu fonksiyon pozisyon kodlayıcı kanallarını sıfırlar. `channelNumber` parametresi sıfırlanacak pozisyon kodlayıcı kanal numaraları dizisi, `channelNumbersLength` parametresi dizinin boyutudur. Encoder kanal numaraları 0-7 arasında değişmektedir.

• `bool readDigital( const unsigned int channelNumber )`

Belirtilen dijital kanalın değerini okur. `channelNumber` parametresi değeri okunacak dijital kanal numarasıdır. Fonksiyon geriye dijital kanalın değerini getirir. Dijital kanal numaraları 0-31 arasında değişmektedir.

• `void readDigital( const unsigned int* inputChannels, bool* bits, const int inputChannelsLength)`

Bu fonksiyon dijital kanalların değerlerini okur. `inputChannels` parametresi değeri okunacak dijital kanal numaraları dizisi, `bits` parametresi okunan değerlerin yazılacağı dizi, `inputChannelsLength` parametresi dizinin boyutudur. Dijital kanal numaraları 0-31 arasında değişmektedir.

• `void setDigitalOutputDirection( const unsigned int *outputChannels, const int outputChannelsLength)`

Bu fonksiyon dijital çıkış kanallarını belirlemek için kullanılır. `outputChannels` parametresi çıkış kanalı olarak tanımlanacak dijital kanal numaraları dizisi, `outputChannelsLength` parametresi dizinin boyutudur. Varsayılan olarak dijital kanalların hepsi giriş kanalıdır. Bu fonksiyon kullanılarak dijital çıkış kanalları tanımlanır.

• `void writeDigital( const unsigned int* outputChannels, const bool* bits, const int outputChannelsLength)`

Bu fonksiyon digital çıkış kanallarına tanımlanan değerleri yazar. `outputChannels` parametresi yazılacak dijital kanal numaraları dizisi, *bits* parametresi digital kanallara yazılacak değerler dizisi, `outputChannelsLength` parametresi dizinin boyutudur. Dijital kanal numaraları 0-31 arasında değişmektedir.

4.2.2. Yüksek Seviyeli HapticWand Sınıfı

Robotikte gelişmiş kontrol uygulamaları ileri ve ters kinematik gibi karmaşık matematik algoritmaları içerir. Bu algoritmaları başarılı bir şekilde elde etmek kolay olmadığı gibi robotun boyutları ve parametreleri hakkında bilgi sahibi olmak gerekir. 5-DOF Haptic Wand cihazında bu karmaşık işlemleri kolaylaştırmak için

HapticWand sınıfı yapıldı. Bu sınıf kullanılarak kontrol uygulaması tasarımları önemli ölçüde kolaylaştırılır.

5 DOF Haptic Wand cihazı ile kontrol uygulamasına başlarken ilk yapılacak iş pozisyon kodlayıcıdaki sayım değerlerini dünya koordinat sistemine çevirmektir. *jointAngles* fonksiyonu çağrılarak robotun eklem değerleri 6 boyutlu dizi ile radyan cinsinden alınır.

Eklem açıları elde edildikten sonra *forwardKinematics* fonksiyonu kullanılarak cihazın dünya koordinatları elde edilir. Bu fonksiyon *theta* ve *worldCoordinates* parametrelerini alır. *theta* giriş, *worldCoordinates* çıkış parametresidir. *theta* giriş parametresi her biri robot eklemlerinin radyan cinsinden açılarını gösteren 6 boyutlu dizidir. *worldCoordinates* çıkış parametresi haptik cihazın sonlandırıcı kısmının konumu ve yönünü içeren 5 elemanlı dizidir. *worldCoordinates* dizisinin ilk üç elemanı x, y, z konumunu metre cinsinde ifade ederken, son iki eleman sırasıyla radyan cinsinde yunuslama (ing: pitch) ve yuvarlanma (ing: roll) değerleridir.

Cihazın konumu bulunduktan sonra bir pozisyon kontrolü ile arzu edilen konum ve yön ile beraber uygulanması gereken kuvvet hesaplanır. Bununla birlikte bu kuvvetin motorlara uygulanabilmesi için voltaj değerlerine dönüştürülmesi gerekir. İşte bu kuvvet voltaj dönüşümünü *generateForces* fonksiyonu kullanılarak yapılır. Bu fonksiyon ile gerçek dünya koordinat sistemindeki kuvvet değerleri cihaz eklemleri için tork değerlerine dönüştürülür. Bu fonksiyon *period, joint_angles* ve *world_forces* olmak üzere 3 parametre alır. *period* parametresi kontrol uygulamasının çalışma periyot değeridir. *joint_angles* cihaz eklemlerinin radyan cinsinden açılarını gösteren 6 boyutlu dizidir. *world_forces* parametresi cihazın uygulaması istenen kuvvet değerlerinin olduğu 5 boyutlu dizidir. İlk üç eleman x, y, z konumları için Newton cinsinde değerler, son iki eleman yunuslama ve yuvarlanma değişimleri için Newton-metre cinsinden değerlerdir.

5 DOF Haptic Wand cihazı için ileri kinematikten ters kinematiğe kadar HapticWand ara yüzü kullanılarak ile izlenmesi gereken yol anlatıldı. Bu adımların zenom programı ile örnek kullanımı beşinci bölümde bahsedilmiştir. Şekil. 4.4'de Haptic Wand sınıfı verilmiştir.

```
class HapticWand : public HIL
{
public:
    HapticWand();

    virtual void open(const char* cardType = "Q80", const int
cardIdentifier = 0);

    void enableWand();

    void disableWand();

    void calibrateWand();

    void resetEncoders();

    void readEncoders( int* counts );

    void writeAnalogs( const double* voltages );

    void jointAngles( double* jointAngles );

    void forwardKinematics( double* theta, double* worldCoordinates );

    void generateForces( double period, const double joint_angles[], const
double world_forces[] );

    inline const double* firstSample() { return mFirstSample; }



    ...


};
```

Şekil 4.5: HapticWand sınıfı

Aşağıdaki listede sınıfa ait fonksiyon tanımları yapılmıştır.

- `void open(const char* cardType, const int cardIdentifier)`

Bu fonksiyon 5 DOF Haptic Wand cihazına ait kartı açar. Ayrıca 0, 1, 2, 3, 16 ve 17 numaralı dijital kanalları çıkış kanalları olarak yapılandırır. Cihaz ile bilgi alış-veriş fonksiyonları çağrılmadan önce muhakkak bu fonksiyon ile kart açılmalıdır. Ayrıca kart ile iş bittiğinde `close` fonksiyonu ile kapatılması gerekir. `cardType` parametresi kartın tipi, `cardIdentifier` parametresi ile kartın tanımlayıcısıdır.

- `void enableWand()`

Cihazın kontrol edilebilmesi için 0, 1, 2, 3, 16 ve 17 numaralı dijital çıkış kanallarına 1 değeri yazar. Bu fonksiyon çağrılmadan cihaza veri yazılamaz. Ayrıca

bu fonksiyon çağrıldıktan sonra `firstSample` fonksiyonu ile cihazın ilk konumu alınabilir.

- `void disableWand()`

Cihazın 0, 1, 2, 3, 16 ve 17 numaralı dijital çıkış kanallarına 0 değeri yazar. Ayrıca kullanılan 0, 1, 2, 3, 4 ve 5 numaralı analog kanallarına 0 değerini yazar.

- `void calibrateWand()`

Bu fonksiyon ile 6 DOF Haptic Wand cihazı kalibre edilir. Dijital çıkış kanallarına 0 yazılır ve pozisyon kodlayıcı sayım değerleri sıfırlanır. Bu fonksiyon `disableWand()` ve `resetEncoders()` fonksiyonlarını peş peşe çağırır.

- `void resetEncoders()`

Cihazın ve pozisyon kodlayıcı sayım değerlerini sıfırlar. 0, 1, 2, 3, 4 ve 5 numaralı ve pozisyon kodlayıcı kanallarına 0 değeri yazar.

- `void readEncoders( int* counts )`

Bu fonksiyon 0, 1, 2, 3, 4 ve 5 numaralı ve pozisyon kodlayıcı kanallarının sayım değerlerini okur, `counts` parametresine değerleri yazar. `counts` parametresi 6 boyutlu bir dizi olmak zorundadır.

- `void writeAnalogs( const double* voltages )`

Cihazın 0, 1, 2, 3, 4 ve 5 numaralı analog kanallarına `voltages` parametresindeki değerleri yazar. `voltages` parametresi 6 boyutlu bir dizi olmak zorundadır.

- `void jointAngles( double* jointAngles )`

Cihazın 0, 1, 2, 3, 4 ve 5, numaralı ve pozisyon kodlayıcı sayım değerlerini radyan cinsinde açı değerlerine çevirir ve `jointAngles` parametresine yazar. `jointAngles` parametresi 6 boyutlu bir dizi olmak zorundadır.

- `void forwardKinematics( double* theta, double* worldCoordinates )`

Bu fonksiyon `theta` parametresindeki radyan cinsinden encoder sayım değerlerini dünya koordinat sistemine çevirerek `worldCoordinates` parametresine yazar. `theta` parametresi 6 boyutlu, `worldCoordinates` parametresi 5 boyutlu bir dizi olmak zorundadır. `worldCoordinates` parametresinin ilk üç elemanı x,y,z konumunu metre cinsinden değeri, dördüncü ve beşinci elemanlar ise sırasıyla radyan cinsinde yunuslama (ing: pitch) ve yuvarlanma (ing: roll) değerleridir.

- `void generateForces( double period, const double* joint_angles, const double* world_forces )`

Bu fonksiyon ile gerçek dünya koordinat sistemindeki kuvvet değerleri cihaz eklemleri için tork değerlerine dönüştürülür. Bu fonksiyon `period`, *joint*`_angles` ve *world*`_forces` olmak üzere 3 parametre alır. `period` parametresi kontrol uygulamasının çalışma periyot değeridir. `joint_angles` cihaz eklemlerinin radyan cinsinden açılarını gösteren 6 boyutlu dizidir. `world_forces` parametresi cihazın uygulaması istenen kuvvet değerlerinin olduğu 5 boyutlu dizidir. İlk üç eleman x, y, z konumları için Newton cinsinde değerler, son iki eleman yunuslama ve yuvarlanma değişimleri için Newton-metre cinsinden değerlerdir.

- `const double* firstSample()`

Bu fonksiyon enableWand fonksiyonu çağrıldıktan sonraki cihazın konumunu dünya koordinat sisteminde getirir.

5. KONTROL UYGULAMALARI

Bu bölümde Zenom ortamında 5-DOF Haptic Wand cihazının HapticWand kütüphanesi ile kontrol edildiği örnekler anlatılmıştır. Tracker ve Sphere olmak üzere iki örnek yapılmıştır. Tracker örneği 3 boyutlu dünya koordinat sisteminde pozisyon kontrolüdür. Sphere örneği ise sanal gerçeklik uygulamasıdır. Sanal bir küreye dokunulduğu zaman, geri bildirim kuvveti uygulanmaktadır.

Örnekler yapılırken Zenom ile birlikte gelen QMath kütüphanesi kullanılmıştır. QMath robotikte yaygın olarak kullanılan matematik fonksiyonlarını basit ve hafif olarak sunan bir kütüphanedir. İçerisinde matris, vektör, sayısal filtreleme, integral ve türev işlemlerini kolaylaştıran sınıflar vardır.

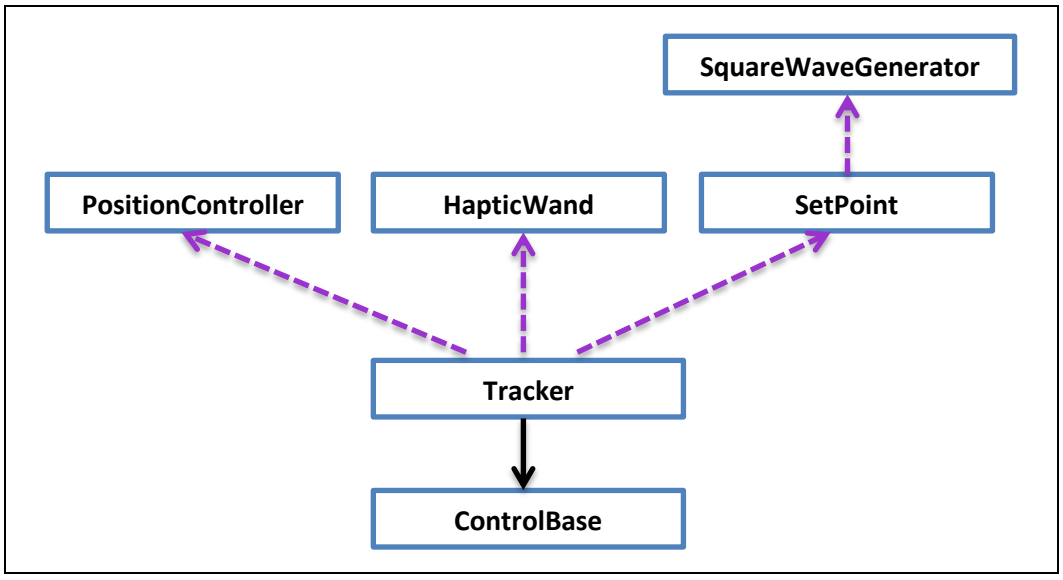
QMath Euler tabanlı ikinci dereceden filtreleme sınıfı da içerir. Yapılan örneklerde gürültüyü azaltmak için bu metot kullanılmıştır.

5.1. Pozisyon Kontrolü: Tracker

5.1.1. Tasarım

Tracker örneği 3 boyutlu dünya koordinat sisteminde pozisyon kontrolüdür. Bu örnek ile çubuk konumunun arzulanan çubuk konumuna ulaşması için uygulanması gereken kuvvetin nasıl hesaplanacağı ve Zenom ortamına döngüde donanım uygulamalarının nasıl entegre edileceği anlatılmıştır.

Tracker örneği Şekil 5.1'de gösterilen bileşenlerden oluşur. Kesik çizgiler kullandığı sınıfları, düz çizgiler türetildiği sınıfı temsil etmektedir. Tracker, Zenom'un sunduğu ara yüz gereği ControlBase sınıfından türetilen gerçek zamanlı program akışının yönetildiği sınıftır. SetPoint sınıfı bir sonraki adımda cihazın ulaşması istenen pozisyon bilgilerini üretir. PositionController sınıfı cihaz konumunu istenilen konuma ulaştırmak için gereken gücü hesaplar. HapticWand ise bu tez kapsamında yapılan ve cihaz ile haberleşmeyi mümkün kılan uygulama programlama ara yüzüdür.

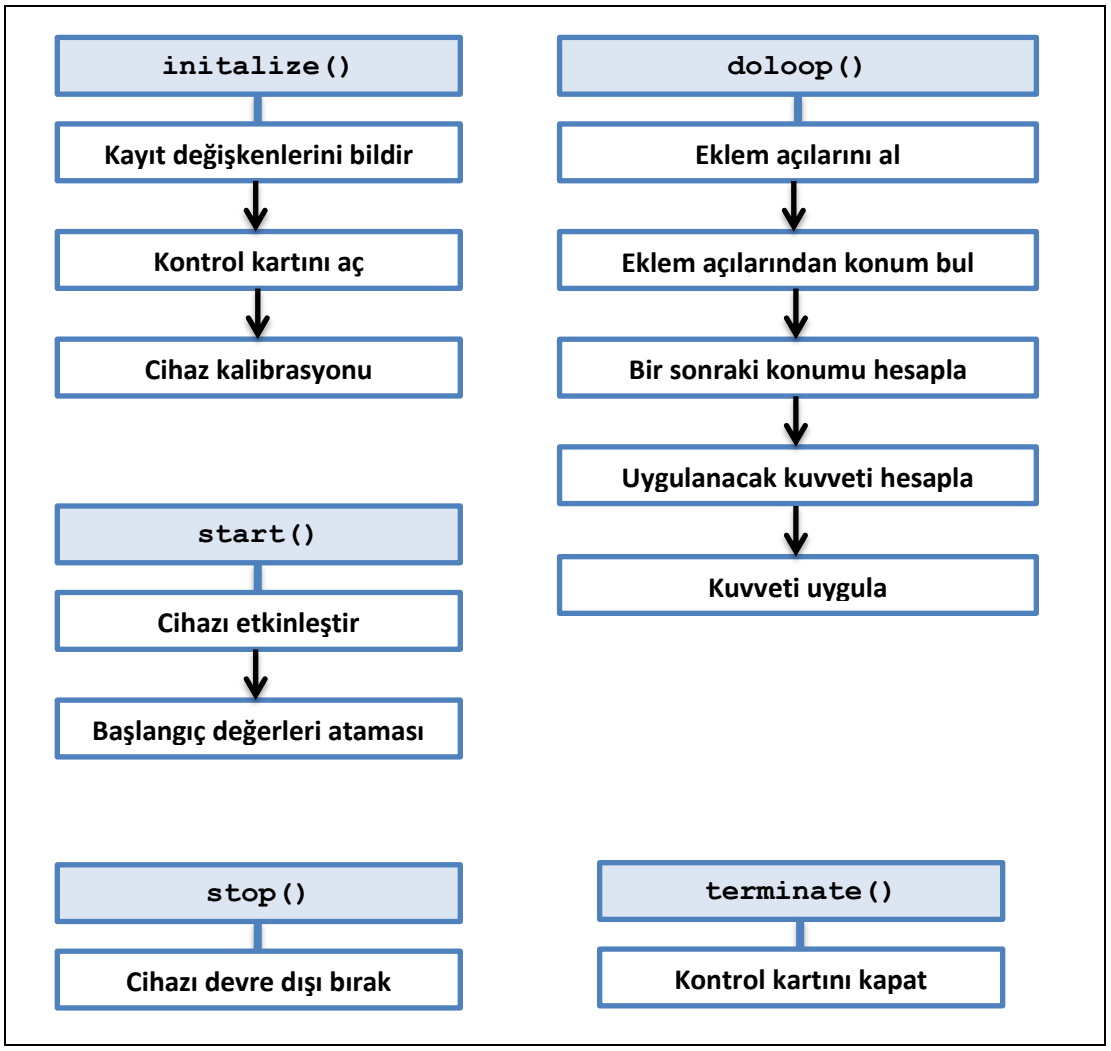


Şekil 5.1: Tracker bileşenleri

Tracker, Zenom'da kontrol programı geliştirmek için kullanılan ControlBase sınıfında türetilerek, doldurulması gereken "initialize", "start", "doloop", "stop" ve "terminate" sanal fonksiyonlar gerçeklenmiştir. Şekil 5.2'de Tracker sınıfının fonksiyon ve değişken tanımlamaları gösterilmiştir. Şekil 5.3'de ise sanal fonksiyonların akış şemaları verilmiştir.

```
class Tracker : public ControlBase
{
public:

    // ----- User Functions -----
    // This functions need to be implemented by the user.
    int initialize();
    int start();
    int doloop();
    int stop();
    int terminate();

private:
    // ----- Log Variables -----
    ColumnVector<5> w;      // current world position.
    ColumnVector<5> wd;     // desired world position.

    // ----- Variables -----
    HapticWand hapticWand; // controls Quanser's 5DOF Haptic Wand.
    SetPoint setPoint;     // generates position.
    PositionController positionController;   // calculates forces.
    ColumnVector<5> firstSample;   // first position of wand
};
```

Şekil 5.2: Tracker sınıfı



Şekil 5.3: Tracker - Kontrol fonksiyonları akış şemaları

`initialize()` fonksiyonu kontrol programı Zenom'a yüklendiği zaman çağrılır ve kayıt ve kontrol değişkenlerini Zenom'a bildirmek için kullanılır. Bu fonksiyonda *w* ve *wd* değişkenleri kayıt değişkeni olarak kaydedilmiştir. *w* değişkeni cihaz konumunu, *wd* arzu edilen cihaz konumunu saklar. Bu değişkenler kayıt değişkeni olarak Zenom'a bildirilmiştir; çünkü Zenom ortamındaki iki boyutlu grafik ekranı kullanılarak bu iki değişken arasındaki hata payı izlenecektir. Ayrıca bu fonksiyonda 5-DOF Haptic Wand cihazı ile iletişim kurmak için Q8 HIL kontrol kartı açılır ve cihazın kalibrasyonu yapılır.

```
int Tracker::initialize()
{
    registerLogVariable( w.getElementsPointer(), "w", 1, 5 );
    registerLogVariable( wd.getElementsPointer(), "wd", 1, 5 );

    hapticWand.open();              // Open the q8 card
    hapticWand.calibrateWand();     // Calibrate the haptic wand

    return 0;
}
```

Şekil 5.4: Tracker - initalize metodu görüntüsü

`start()` fonksiyonunda sınıfların başlangıç değerleri ataması, sıfırlama işlemleri ve donanımı etkinleştirme adımı yapılır. Şekil 5.5'de `start()` fonksiyonu gösterilmiştir. `enableWand()` fonksiyon çağrısı ile 5 DOF Haptic Wand cihazının kullanıma alındığı bilgisi gönderilir. Bu fonksiyonu çağırmadan cihaza bilgi yazılamayacağını unutmamak gerekir.

```
int Tracker::start()
{
    hapticWand.enableWand();        // enable haptic wand

    setPoint.reset();               // reset set point class

    firstSample =
            hapticWand.firstSample()[0],
            hapticWand.firstSample()[1],
            hapticWand.firstSample()[2],
            hapticWand.firstSample()[3],
            hapticWand.firstSample()[4];

    // reset position controller
    positionController.reset( firstSample, period() );

    return 0;
}
```

Şekil 5.5: Tracker - start metodu görüntüsü

`doloop()` fonksiyonu Zenom kontrol uygulamasında belirtilen frekans aralığında periyodik olarak çağrılan fonksiyondur. Bu fonksiyon bloğunda 5 DOF Haptic Wand cihazından eklem açıları radyan cinsinden alınır. Eklem açıları koordinat sistemine çevirilerek *w* değişkenine yazılır. *SetPoint* sınıfı yardımıyla cihazın bir sonraki istenen konumu olan *wd* hesaplanır. PositionController ile *w*

konumundaki cihazın *wd* konumuna ulaşması için gereken kuvvet hesaplanır. Hesaplanan *F* kuvveti HapticWand sınıfının *generateForce* fonksiyonu kullanılarak cihaza iletilir.

```
int Tracker::doloop()
{
    double jointAngles[6];   // joint angles in radians
    hapticWand.jointAngles( jointAngles );
    hapticWand.forwardKinematics( jointAngles,
w.getElementsPointer() );    // current world position.

    // Desired position.
    wd = setPoint.wd( simTimeInSec() , period() ) + firstSample;

    // calculate forces.
    ColumnVector<5> F;
    F = positionController.force( w, wd );

    // convert forces to joint torques
    hapticWand.generateForces( period(), jointAngles,
F.getElementsPointer() );

    return 0;
}
```

Şekil 5.6: Tracker - doloop metodu görüntüsü

Zenom kontrol uygulaması durdurulduğu zaman veya süresi dolduğunda `stop()` fonksiyonu çağrılır. Bu aşamada HapticWand ara yüzünün `disableWand()` fonksiyonu çağrılarak 5-DOF Haptic Wand cihazı devre dışı kalır. Bu andan itibaren cihazdan veri okunma ve cihaza veri yazma yapılmayacağı için cihaz devre dışı bırakılır.

```
int Tracker::stop()
{
    hapticWand.disableWand();   // disable haptic wand

    return 0;
}
```

Şekil 5.7: Tracker - stop metodu görüntüsü

`terminate()` fonksiyonu Zenom ortamından çıkış yapılınca çağrılan fonksiyondur. Bu fonksiyonda erişime açılan donanımsal kartların kapatılması

gerekir. Bu yüzden `initalize()` fonksiyonunda açılan Q8 HIL kontrol kartı `close()` fonksiyonu çağrılarak kapatılır.

```
int Tracker::terminate()
{
    hapticWand.close();   // close the q8 card

    return 0;
}
```

Şekil 5.8: Tracker - terminate metodu görüntüsü

SetPoint sınıfı cihazın ulaşması istenen pozisyonu üretir. Bunun için SquareWaveGenerator sınıfını kullanır. SquareWaveGenerator istenilen frekans ve zaman aralığında kare dalga formu üreten bir sınıftır. SetPoint sınıfında bir sonraki cihaz konumu *wd* fonksiyonu çağrısı ile getirilir. *wd* fonksiyonuna benzetim zamanı verilerek cihazın olması istenen *x, y, z, pitch* ve *roll* konumları hesaplanır. Buradaki katsayılar cihazın hangi aralıkta hareket edeceğini belirler. Bu örnek için cihazın *x, y, z* ±30 mm, *pitch, roll* ±20º aralığında hareket etmesi istenmiştir. Ayrıca yörüngeye hız limiti koymak için lineer interpolasyon uygulanmıştır.

```
ColumnVector<5> SetPoint::wd(double pTime, double pSamplingPeriod)
{
    ColumnVector<5> wd;
    wd =
         squareWave_x( pTime ) * 30,
         squareWave_y( pTime ) * 30,
         squareWave_z( pTime ) * 30,
         squareWave_yaw( pTime ) * 20,
         squareWave_roll( pTime ) * 20,

    wd = elementProduct( wd, toSI );

    current = linear_interpolator( wd, current, v_lim,
pSamplingPeriod );

    return current;
}
```

Şekil 5.9: Tracker - SetPoint sınıfı wd metodu görüntüsü

PositionController sınıfı şu anki cihaz konumundan SetPoint ile hesaplanan konuma ulaştırmak için gereken gücü hesaplar. Kuvvet hesabında ikinci dereceden

Euler tabanlı filtreleme kullanılmıştır. Sertlik ve sönümleme kat sayıları olarak Tablo 3.3'de verilen değerler kullanılmıştır. *force* fonksiyonu şu anki konumu ve arzu edilen konumu parametre olarak alır ve uygulanması gereken kuvvet vektörünü döndürür.

```
ColumnVector<5> PositionController::force( ColumnVector<5>& w,
ColumnVector<5>& wd )
{
    ColumnVector<5> F_stiff = wd - w;
    F_stiff = elementProduct(F_stiff, stiffness);

    ColumnVector<5> F_damp = digitalFilter.integrate( w );
    F_damp *= -1;
    F_damp = elementProduct( F_damp, damping );

    return F_stiff + F_damp;
}
```

Şekil 5.10: Tracker - PositionController sınıfı force metodu görüntüsü

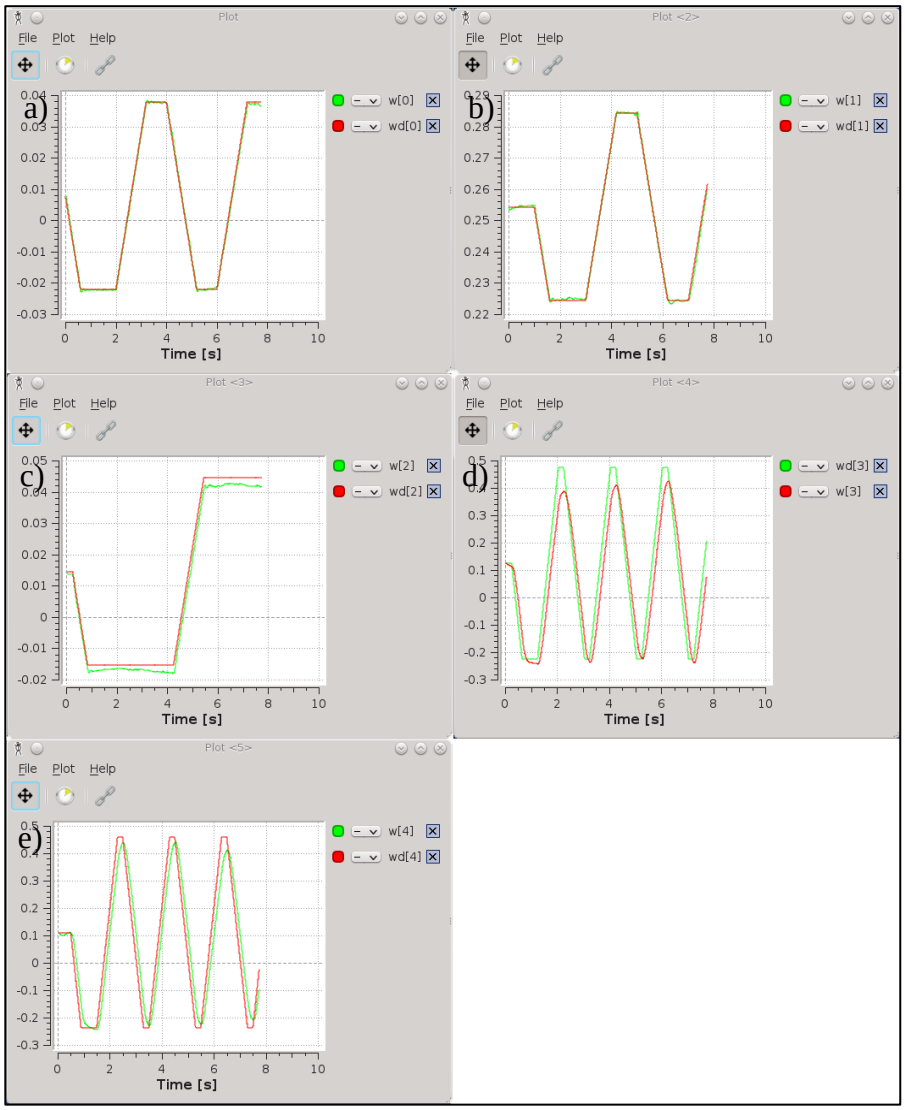
5.1.2. Çalıştırma Prosedürü

Örneği çalıştırmadan önce bazı ön koşulların yerine getirilmelidir. *q8driver* sürücü modülü işletim sistemine yüklenmelidir. Bu yüzden Bölüm 4.1'de anlatılan şekilde sürücü işletim sistemi çekirdeğine yüklenmiş olmalıdır. 5 DOF Haptic Wand cihazı kalibrasyon etmek için kullanılan parçanın takılması ve çubuğun tutturulmuş olması gerekmektedir.

Ön koşulları yerine getirdikten sonra aşağıdaki adımları izleyin:

- Adım 1: Konsol penceresi açılır ve *sudo zenom* komutu yazılır.
- Adım 2: Zenom programı açılır. File menüsünden Open Project seçeneği seçilir. Açılan pencereden Tracker örneğinin bulunduğu adrese gidilir. Bu adresten Tracker.znm dosyası seçilir.
- Adım 3: Tracker uygulaması Zenom'a yüklendikten sonra 5 adet grafik penceresi açılır. Cihazı kalibrasyon etmek için kullanılan parça çıkartılır.
- Adım 4: Cihazın çubuk kısmı yaklaşık olarak çalışması istenilen alana getirilir.
- Adım 5: Zenom penceresinden Start düğmesine basılır. Cihaz çubuğu x, y, z eksenleri boyunca ±30 mm'lik bir yörüngede, yuvarlanma ve yunuslama

±20º'lik dönüşlerle bir yol çizer. Grafik pencerelerinde *w* ve *wd* kayıt değişkenlerinin grafikleri görüntülenir. Grafik pencerelerindeki iki eğri arasındaki farklılıklar hata payını gösterir. Grafiklerin görüntüsü Şekil 5.11'de gösterilen grafiklere benzer.



Şekil 5.11: Tracker - a) x ekseni grafiği, b) y ekseni grafiği, c) z ekseni grafiği, d) yunuslama grafiği, e) yuvarlanma grafiği.

- Adım 7: Uygulamayı durdurmak için çubuğu tutmaya hazır olmalısınız. Zenom ana pencereden Stop düğmesine bastığınız andan itibaren cihaza kuvvet uygulanmayacağı için çubuk yere düşecektir.
- Adım 8: Kalibrasyon cihazı takılarak çubuk cihaza tutturulur.

5.2. Sanal Gerçeklik: Sphere

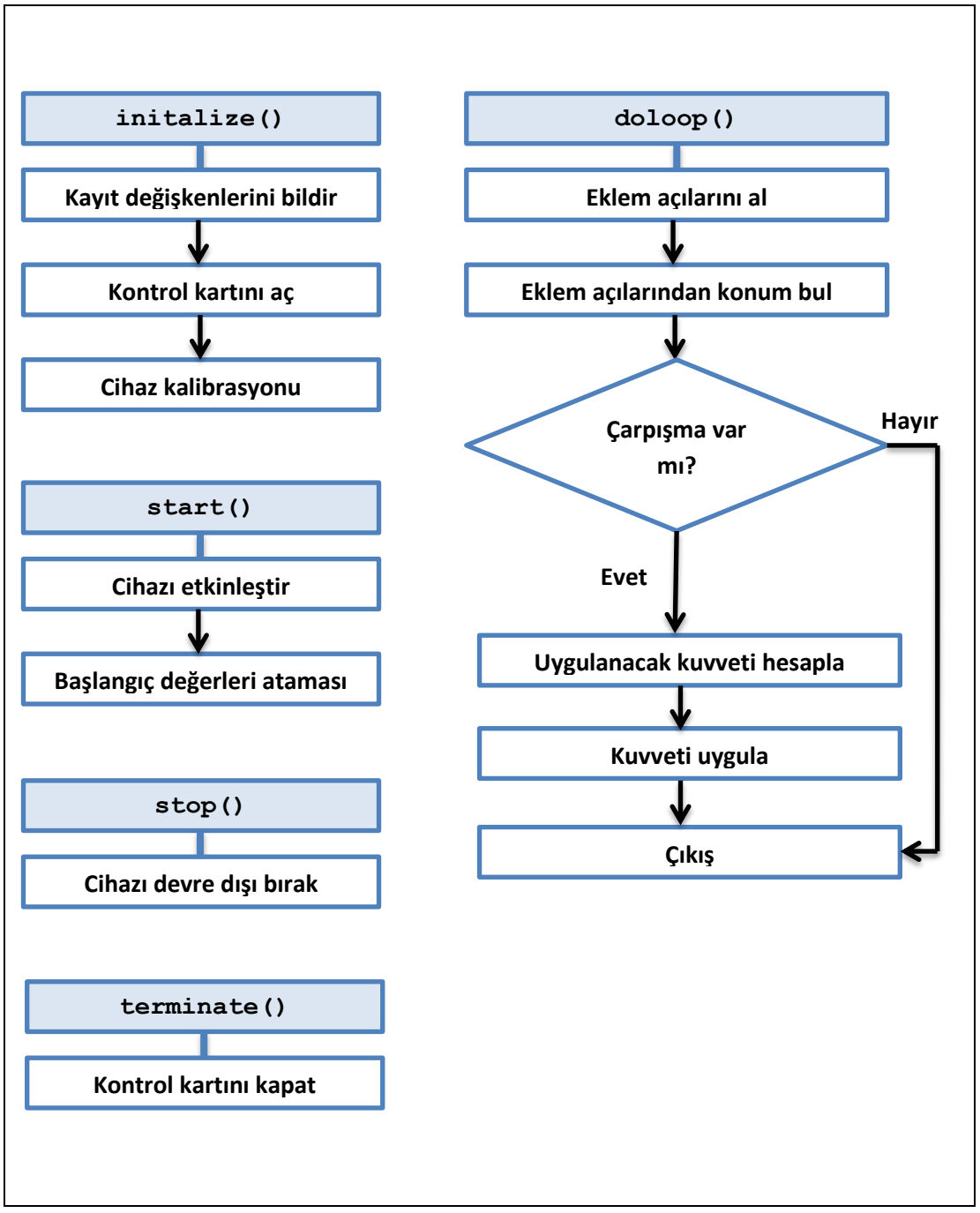
5.2.1. Tasarım

Sphere örneği sanal gerçeklik uygulamasıdır. Sanal dünyada bir küre yer almaktadır ve bu küreye dokunulduğu zaman, geri bildirim kuvveti uygulanarak süngerimsi bir küre hissedilmesini sağlar. Bu örnekte sanal gerçeklik için Zenom'da bulunan Scene penceresi kullanılmıştır. OpenSceneGraph kullanılarak bir küre ve çubuk içeren *wand_and_sphere.osg* model dosyası oluşturulmuştur. Scene ekranında bu model dosyası açılarak, çubuk şekline cihaz konumlarını tutan ilgili kayıt değişkenleri bağlanmıştır. Bu sayede cihaz ile yapılan hareketler sanal ortamda izlenebilmiştir.

Sphere Zenom kontrol programı ara yüz sınıfı olan ControlBase'den türetilen tek sınıf içerir. Bu sınıf içerisinde cihazın konum bilgileri alınarak çarpışma tespiti yapılır. Şekil 5.12'de Spere sınıfı gösterilmiştir.

```
class Sphere : public ControlBase
{
public:

    // ----- User Functions -----
    // This functions need to be implemented by the user.
    int initialize();
    int start();
    int doloop();
    int stop();
    int terminate();

private:
    ColumnVector<5, double> virtual_object( ColumnVector<3,
double>& center, double radius, ColumnVector<5, double>& k_stiff,
ColumnVector<5, double>& k_damp, ColumnVector<3, double>& w_xyz,
ColumnVector<5, double>& w_dot );

    // ----- Log Variables -----
    double position[3];    // position of the wand
    double rotation[4];    // rotation of the wand
    ColumnVector<5, double> F;    // world forces

    // ----- Variables -----
    HapticWand hapticWand;  // 5-DOF Haptic Wand
    Euler2DigitalFilter< ColumnVector<5> > digitalFilter;
};
```

Şekil 5.12: Sphere Sınıfı



Şekil 5.13: Sphere - Kontrol fonksiyonları akış şemaları

`initialize()` fonksiyonunda *position*, *rotation* ve *F* değişkenleri kayıt değişkeni olarak Zenom ortamına bildirilir. *position*, *rotation* değişkenleri Zenom Scene ekranında çubuğa bağlanarak cihaz ile yapılan hareketleri sanal ortamda göstermek için kullanılacaktır. 5-DOF Haptic Wand cihazı ile iletişim kurmak için Q8 HIL kontrol kartı açılır ve cihazın kalibrasyonu yapılır.

```
int Sphere::initialize()
{
    registerLogVariable( position, "position", 1, 3 );
    registerLogVariable( rotation, "rotation", 1, 4 );
    registerLogVariable( F.getElementsPointer(), "F", 1, 5 );

    hapticWand.open();              // Open the q8 card
    hapticWand.calibrateWand();     // Calibrate the haptic wand

    return 0;
}
```

Şekil 5.14: Sphere - initalize metodu görüntüsü

`start()` fonksiyonunda Tracker sınıfına benzer şekilde başlangıç değerleri ataması, sıfırlama işlemleri ve donanımı etkinleştirme adımı yapılır. HapticWand sınıfının `enableWand()` fonksiyon çağrısı ile cihaz kullanıma alınır.

```
int Sphere::start()
{
    hapticWand.enableWand();

    double wn_f = 150;    // Filter Cutoff (rad/s)
    double zeta_f = 1;    // Filter Damping Ratio

    digitalFilter.setSamplingPeriod( period() );
    digitalFilter.setCutOffFrequencyRad( wn_f );
    digitalFilter.setDampingRatio( zeta_f );

    ColumnVector<5> first_sample;
    first_sample =
            hapticWand.firstSample()[0],
            hapticWand.firstSample()[1],
            hapticWand.firstSample()[2],
            hapticWand.firstSample()[3],
            hapticWand.firstSample()[4];
    digitalFilter.reset( first_sample );

    return 0;
}
```

Şekil 5.15: Sphere - start metodu görüntüsü

`doloop()` fonksiyon bloğunda 5 DOF Haptic Wand cihazından konum bilgisi alınıp, küre ile temas edip etmediğine hesaplanır. 5 DOF Haptic Wand cihazından eklem açıları radyan cinsinden alınır. Eklem açıları koordinat sistemine çevirilerek *worldCoordinates* değişkenine yazılır. Cihaz koordinatları *transformToOSG* fonksiyonu ile OpenSceneGraph koordinat sistemine çevrilir. Cihazın küreye teması

olup olmadığını *virtual_object* fonksiyonu kontrol eder. *vo_c* kürenin bulunduğu konum, *vo_r* kürenin yarıçapı, *kp* küreye temas olduğundan uygulanan sertlik kat sayısı, *kd* küreye temas olduğundan uygulanan sönümleme kat sayısı, *w_xyz* çubuğun konumu, *w_dot* Euler filtresinin sonuç değeridir. Çarpışma tespitine göre hesaplanan *F* kuvveti HapticWand sınıfının *generateForce* fonksiyonu kullanılarak cihaza iletilir.

```
int Sphere::doloop()
{
    ColumnVector<5> worldCoordinates;
    double jointAngles[6];        // joint angles in radians
    hapticWand.jointAngles( jointAngles );
    hapticWand.forwardKinematics( jointAngles,
worldCoordinates.getElementsPointer() );

    // transform to openscenegraph coordinate system.
    transformToOSG( worldCoordinates.getElementsPointer(),
position, rotation );

    ColumnVector<3> vo_c;    // Center of Ball (m)
    vo_c = 0, 0.124, -0.075;

    double vo_r = 0.05;      // Radius of Ball (m)

    ColumnVector<5> kp;      // Stiffness (N/m)
    kp = 500, 500, 500, 1, 1;

    ColumnVector<5> kd;      // Damping (N.s/m, N.s/rad)
    kd = 5, 5, 5, 0.1, 0.1;

    ColumnVector<3> w_xyz;     // Posion of Wand
    w_xyz = worldCoordinates(1), worldCoordinates(2),
worldCoordinates(3);

    // world rate of wand (m/s,rad/s)
    ColumnVector<5> w_dot = digitalFilter.integrate(
worldCoordinates );

    // check collision detection
    F = virtual_object( vo_c, vo_r, kp, kd, w_xyz, w_dot );

    // Generate the forces and torques for the Haptic Wand
    hapticWand.generateForces( period(), jointAngles,
F.getElementsPointer() );

    return 0;
}
```

Şekil 5.16: Sphere - doloop metodu görüntüsü

`stop()` fonksiyonundan itibaren cihazdan veri okunma ve cihaza veri yazma yapılmayacağı için cihaz devre dışı bırakılır.

```
int Sphere::stop()
{
    hapticWand.disableWand();   // disable haptic wand

    return 0;
}
```

Şekil 5.17: Sphere - stop metodu görüntüsü

`terminate()` fonksiyonunda `initalize()` fonksiyonunda açılan Q8 HIL kontrol kartı `close()` fonksiyonu çağrılarak kapatılır.

```
int Sphere::terminate()
{
    hapticWand.close();   // close the q8 card

    return 0;
}
```

Şekil 5.18: Sphere - terminate metodu görüntüsü

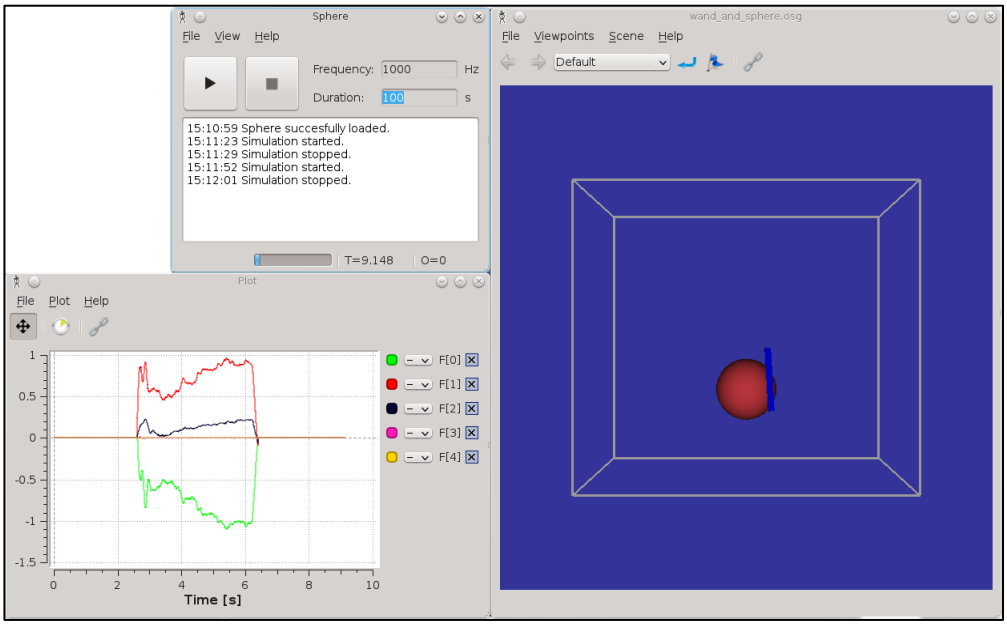
5.1.2. Çalıştırma Prosedürü

Örneği çalıştırmadan önce bazı ön koşulların yerine getirilmelidir. *q8driver* sürücü modülü işletim sistemine yüklenmelidir. Bu yüzden Bölüm 4.1'de anlatılan şekilde sürücü işletim sistemi çekirdeğine yüklenmiş olmalıdır. 5 DOF Haptic Wand cihazı kalibrasyon etmek için kullanılan parçanın takılması ve çubuğun tutturulmuş olması gerekmektedir.

Ön koşulları yerine getirdikten sonra aşağıdaki adımları izleyin:

- Adım 1: Konsol penceresi açılır ve *sudo zenom* komutu yazılır.
- Adım 2: Zenom programı açılır. File menüsünden Open Project seçeneği seçilir. Açılan pencereden Sphere örneğinin bulunduğu adrese gidilir. Bu adresten Sphere.znm dosyası seçilir.
- Adım 3: Sphere uygulaması Zenom'a yüklendikten Şekil 5.15'de gösterilen Scene penceresi açılır. Cihazı kalibrasyon etmek için kullanılan parça çıkartılır.

- Adım 4: Zenom penceresinden Start düğmesine basılır. Cihaz çubuğu hareket ettirildiği zaman Zenom Scene ekranında da sanal çubuğun hareket ettiği görülür.
- Adım 5: Küreye dokunmak için çubuğu yaklaştırın. Küre ile temasa geçtiğinizde bir kuvvet, direnç hissetmeniz gerekir. Eğer hissetmiyorsanız, çubuğun orta noktasının küreye temas ettiğine dikkat edin. Çarpışma tespiti çubuğun orta noktası esas alınarak yapılmıştır.
- Adım 6: Uygulamayı durdurmak için Zenom ana pencereden Stop düğmesine basılır.
- Adım 7: Kalibrasyon cihazı takılarak çubuk cihaza tutturulur.



Şekil 5.19: Sphere - Zenom ekran görüntüsü

6. SONUÇLAR ve YORUMLAR

Tez araştırmasında gerçek zamanlı Linux işletim sistemi üzerinde çalışan Zenom benzetim ortamı kullanılarak 5-DOF Haptic Wand cihazının kontrolü yapılmış ve cihaz için kullanıcı programlama ara yüzü sunulmuştur. Örnek uygulamalar cihaz için tasarlanan kullanıcı programlama ara yüzünün temel kullanımını göstermektedir.

Cihazın Windows işletim sistemi üzerinde MATLAB/Simulink ortamında kontrol edildiği örnekler 500 Hz frekansta çalışmaktadır. Windows işletim sistemi gerçek zamanlı bir işletim sistemi değildir. Bizim tasarladığımız yapıda insan ile etkileşimde olan sistemlerin gerekliliği dikkate alınarak gerçek zamanlı işletim sisteminde çalışmaktadır. Ayrıca Quanser firmasının destek vermediği Linux sistemi üzerinde cihaz kontrol edilmiştir.

Tasarladığımız gerçek zamanlı sistemin performansı kullanılan bilgisayarın hesaplama gücüne bağlıdır. Örneklerde kontrol döngüsünün çalışma frekansı 1 kHz alınmıştır. Ayrıca çalışma frekansının 3 kHz değerine kadar ulaşabildiğini gördük. Ancak 3 kHz üstündeki frekanslarda zaman aşımı oluşmaya başlamıştır. Örnekler bu frekans değerlerinin gerçek zamanlı bir sistem için yeterli performansta olduğunu göstermiştir.

KAYNAKLAR

[1] C. Ay, H. Karaküçük, E. Kaleli, E. Zergeroğlu, (2014), "Linux/Xenomai Tabanlı Gerçek Zamanlı Altyapının Tasarım ve Gerçeklenmesi: Zenom", TOK, Eylül 2014.

[2] Corliss, W.R., Johnson, E.G., (1968) "Teleoperator Controls"; AEC-NASA Technology Survey; NASA, Washington DC, Ref. NASA SP-5070; 1968.

[3] Mosher, R.S., (1964), "Industrial Manipulators"; Scientific American; 1964; 211(4); pp. 88-96.

[4] Robert J. Stone, (2000), "Haptic Feedback: A Potted History, From Telepresence to Virtual Reality", Haptic Human-Computer Interaction, First International Workshop, Glasgow, UK, Proceedings 1-16, August 31 - September 1, 2000

[5] Web 1, (2014), http://www.xenomai.org, (Erişim Tarihi: 02/12/2014).

[6] Web 2,(2014), http://www.quanser.com/Products/haptics, (Erişim Tarihi: 02/12/2014).

[7] 5 DOF Haptic Wand Referans Kılavuzu, (2008), Quanser Inc.

[8] 5 DOF Wand Dinamik Model Kılavuzu, (2008), Maple Worksheet, Quanser Inc.

[9] Massie, T. & Salisbury, J. K. (1994). "The phantom haptic interface: A device for probing virtual objects", Proceedings of the ASME Winter Annual Meeting, Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems, Vol. 55-1, Chicago, IL., 1994, pp. 295-300.

[10] Srinivasan, M.A., Basdogan, C., 1997, "Haptics in Virtual Environments: Taxonomy, Research Status, and Challenges", Computers and Graphics, (Special issue on "Haptic Displays in Virtual Environments"), Vol. 21, No. 4, pp. 393-404.

[11] Web 3,(2014), http://www.quarcservice.com/ReleaseNotes/files/q8.html, (Erişim Tarihi: 02/12/2014).

[12] Kiszka J. (2007), The real-time driver model and first applications, University of Hannover, Germany, Hannover.

[13] Web 4,(2014), http://www.mathworks.com/help/physmod/simscape/ug/real-time-simulation.html, (Erişim Tarihi: 02/12/2014).

[14] Web 5,(2014), http://www.qnx.com, (Erişim Tarihi: 02/12/2014).

[15] Behnam M., Nolte T., Shin I., Asberg M., Bril R., (2008), “Towards hierarchical scheduling in VxWorks.”, 4th International Workshop on Operating Systems Platforms for Embedded Real-Time Applications, 63-72, Czech Republic, 2-4 July.

[16] Mantegazza P., Dozio E. L., Papacharalambous S. (2000), “RTAI: Real time application interface”, Linux Journal, 72, 10-11.

[17] Kaleli, E., Zergeroğlu E. (2014, August). Linux-Xenomai Target: A Real-Time Hardware-In-The-Loop Simulatıon Framework Based on Simulink. (2014 ICCSS) 2nd International Conference on Computational and Social Sciences, Turkey.

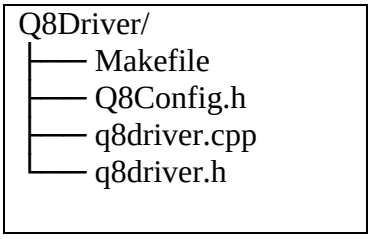
ÖZGEÇMİŞ

Cüneyt AY 1986 yılında İstanbul'da doğdu. 2005 yılında başladığı Gebze Yüksek Teknoloji Enstitüsü (GYTE) Mühendislik Fakültesi Bilgisayar Mühendisliği Bölümünü 2010 yılında tamamlayarak, 2011 yılında yüksek lisans eğitimine GYTE Mühendislik ve Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Anabilim Dalında başladı. 2009 yılından 2013 yılına kadar TÜBİTAK BİLGEM 'de Araştırmacı ünvanı ile yazılım mühendisi olarak çalışmıştır.

EKLER

Ek A: Q8 Aygıt Sürücüsün Derlenmesi

Bu kısımda Q8 aygıt sürücüsünün derlenmesi ve sürücünün işletim sistemi çekirdeğine yüklenmesi için yapılacak adımlar anlatılmıştır. Q8 aygıt sürücüsünün Şekil A1.1'de klasör yapısı gösterilmiştir.



Şekil A1.1: Q8 sürücü modülü klasör yapısı.

Sürücüyü derlemek için *Q8Driver* klasörünün bulunduğu konumda konsol penceresi açın ve aşağıdaki komutları girin:

$$make \tag{A1.1.a}$$

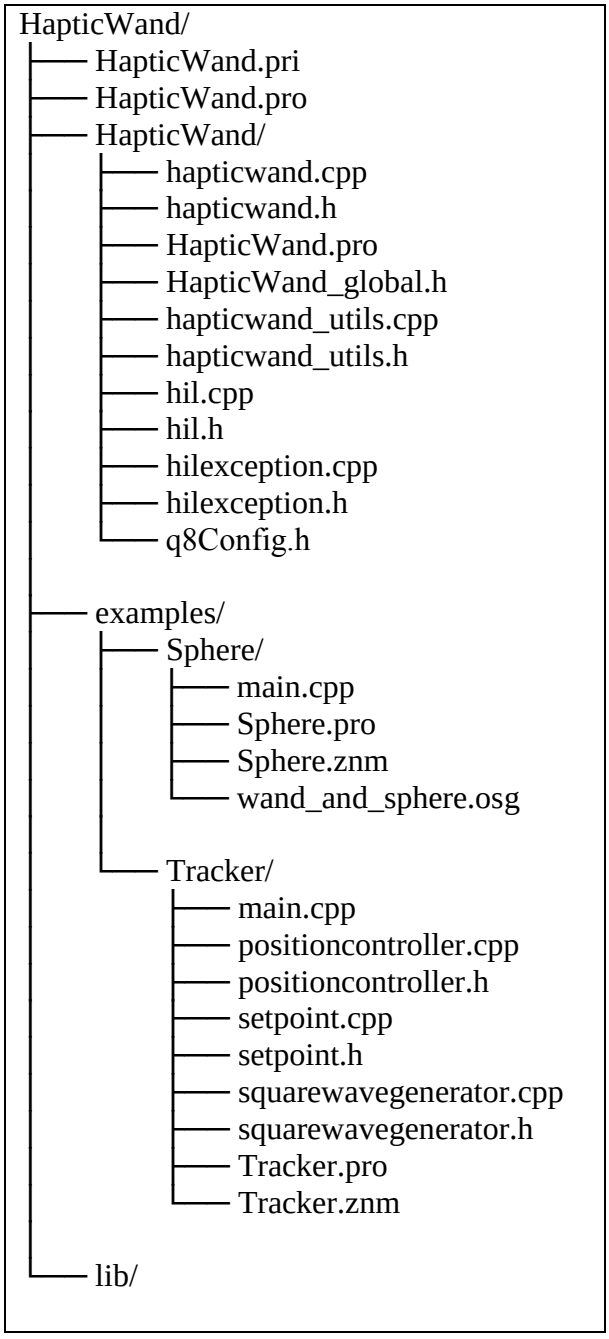
$$insmod\ q8driver.ko \tag{A1.1.b}$$

$$lsmod \tag{A1.1.c}$$

Komut (A1.1.a) ile sürücü derlenir ve *q8driver.ko* isimli sürücü modülü oluşur. Modülü işletim sistemi çekirdeğine yüklemek için komut (A1.1.b) kullanılır. Komut (A1.1.c) ile modülün çekirdeğe yüklenip yüklenmediğini anlamak içindir. Konsol ekranında çıkan listede *q8driver* ismini görüyorsanız sürücüyü başarıyla yüklemişsiniz demektir.

Ek B: HapticWand Uygulama Programlama Arayüzü Derlenmesi

Bu kısımda HapticWand UPA'nın nasıl derlenmesi gerektiği anlatılmıştır. HapticWand UPA'nın Şekil B1.1'de klasör yapısı gösterilmiştir.



Şekil B1.1: HapticWand UPA klasör yapısı.

HapticWand ara yüzü *qmake* kullanılarak yapılmıştır. Bu yüzden ara yüzü derlemek için HapticWand klasörünün bulunduğu konumda konsol penceresi açılır ve aşağıdaki komutları girin:

$$qmake - qt4 \quad \text{(B1.1.a)}$$

$$make \quad \text{(B1.1.b)}$$

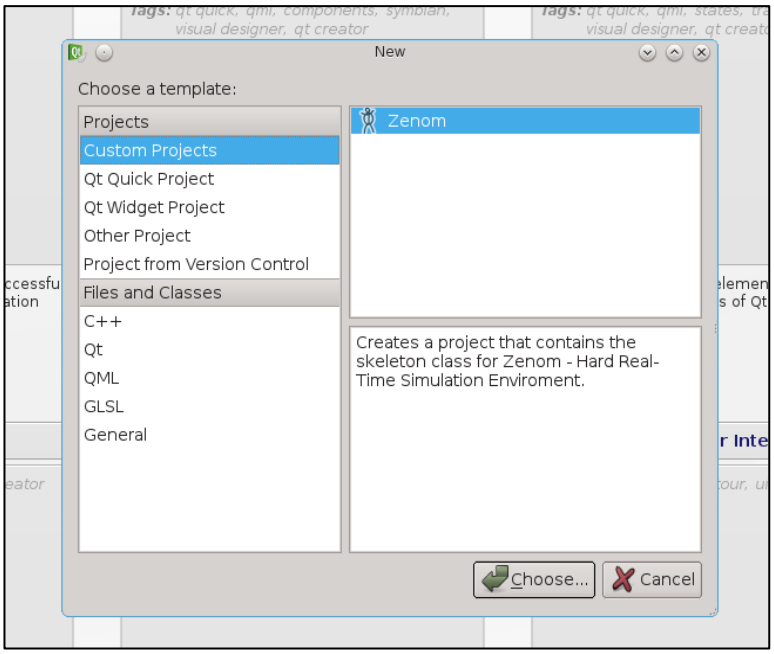
$$sudo\ make\ install \quad \text{(B1.1.c)}$$

Komut (B1.1.a) derlemek için kullanılacak dosyalar hazırlanır ve komut (B1.1.b) ile kütüphane derlenir. Derleme işlemi tamamlandığında *lib* klasörü altında so uzantılı dinamik kütüphaneler oluşur. Komut (B1.1.c) ile kütüphane sisteme yüklenir. Ancak kütüphanenin yüklendiği adresin işletim sisteminin çalışma zamanında bağlanan adresler arasında olmasına dikkat edilmelidir. Daha detaylı bilgi için *man ldconfig* komutunu araştırınız.

Ek C: QtCreator Kullanarak Zenom Projesi Oluşturma

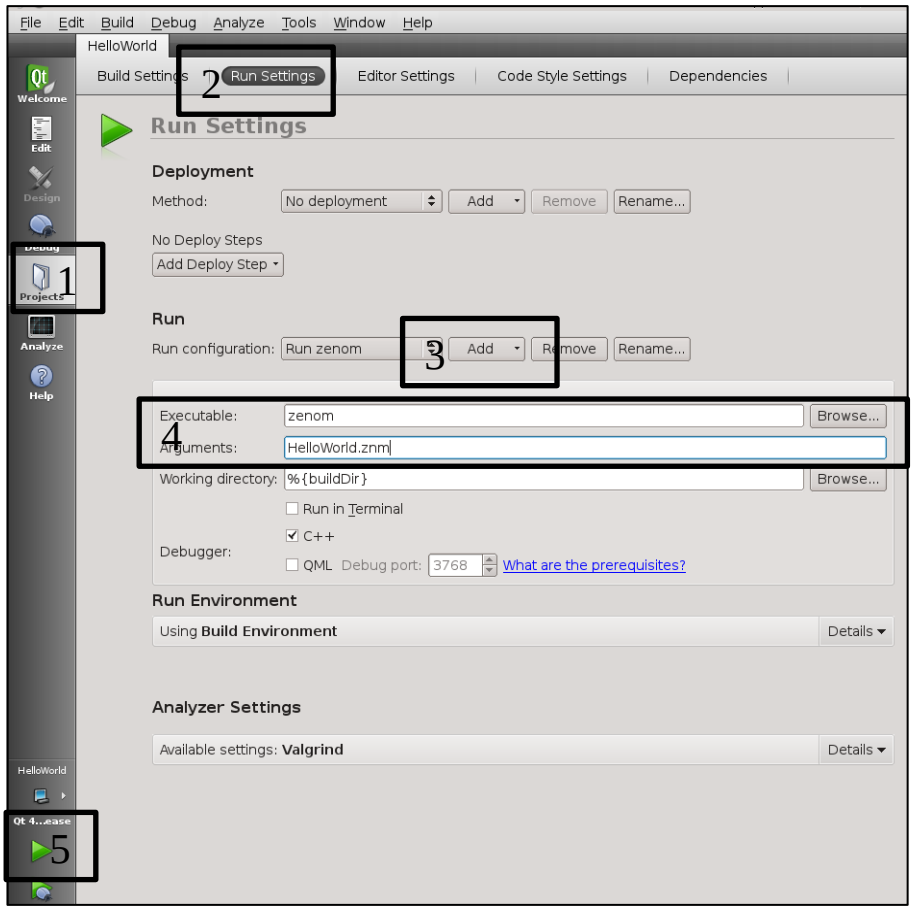
Bu kısımda QtCreator kullanarak Zenom benzetim ortamı için nasıl proje oluşturulacağı anlatılmıştır. QtCreator için "*http://qt-project.org/doc/qtcreator-2.6/creator-project-wizards.html*" adresinde anlatıldığı şekilde Zenom'a özel proje oluşturma sihirbazı yapılmıştır. Bu sihirbaz, Zenom sisteme yüklendiği zaman otomatik olarak yüklenir. Bu sihirbaz ile Zenom projesi oluşturmak için aşağıdaki adımları izleyin:

- Adım 1: Konsol penceresi açılır ve *sudo qtcreator* komutu yazılır.
- Adım 2: QtCreator programı açılır. *File* menüsünden *New File or Project…* seçeneği seçilir. Şekil C1.1'de gösterilen pencere açılır.



Şekil C1.1: QtCreator Proje Oluşturma Penceresi

- Adım 3: Açılan pencereden *Custom Projects* başlığından *Zenom* seçilir ve *Choose…* düğmesine basılır.
- Adım 4: *Name* alanına uygun proje ismi girilir. Bu alana girilen isim aynı zamanda oluşturulacak kontrol sınıfının da ismi olacaktır. *Next* düğmesine basılır. Ayrıca açılan pencereleri *Next* ile geçebilirsiniz.
- Adım 5: Kontrol uygulaması geliştirmek için Zenom şablon sınıfını içeren *main.cpp* oluşturulur. QtCreator ile kontrol uygulamasını Zenom'da otomatik açmak için QtCreator çalıştırma ayarları düzenlemek gerekir. Sol tarafta yer alan *Projects* menüsüne tıklanır ve *Run Settings* sekmesi seçilir.
- Adım 6: Şekil C1.2'de gösterilen *Run Settings* penceresinde *Run configuration* bölümünde *Add* düğmesine tıklanarak *Custom Executable* seçilir. *Executable* alanına *zenom* yazılır. *Arguments* alanına 4. adımda girilen proje ismine .znm ekleyerek yazılır. Örneğin *HelloWorld* olan proje ismi için *HelloWorld.znm* yazılır.



Şekil C1.2: QtCreator Run Settings Penceresi

- Adım 7: QtCreator *Run* düğmesine basıldığında Zenom benzetim ortamı kontrol uygulaması yüklenmiş olarak açılır.