T.C.

GEBZE YÜKSEK TEKNOLOJİ ENSTİTÜSÜ

MÜHENDİSLİK VE FEN BİLİMLERİ

ENSTİTÜSÜ

QUANSER I/O KARTI İÇİN MATLAB ALTINDA REAL-TIME LINUX-XENOMAI TARGET SÜRÜCÜ YAZILIMI GELİŞTİRİLMESİ VE UYGULAMASI

EGEMEN CUMHUR KALELİ

YÜKSEK LİSANS TEZİ

BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

GEBZE

2011

YÜKSEK LİSANS TEZİ JÜRİ ONAY SAYFASI

G.Y.T.E. Mühendislik ve Fen Bilimleri Enstitüsü Yönetim Kurulu'nun 04.07.2011 tarih ve 2011/21 sayılı kararıyla oluşturulan jüri tarafından 08.09.2011 tarihinde tez savunma sınavı yapılan Egemen Kaleli'nin tez çalışması Bilgisayar Mühendisliği Anabilim Dalında YÜKSEK LİSANS tezi olarak kabul edilmiştir.

JÜRİ

ÜYE
(TEZ DANIŞMANI) : Doç.Dr. Erkan ZERGEROĞLU

ÜYE : Yrd.Doç.Dr.Abdülkadir BALIKÇI

ÜYE : Yrd.Doç.Dr. F.Erdoğan SEVİLGEN

ONAY

G.Y.T.E. Mühendislik ve Fen Bilimleri Enstitüsü Yönetim Kurulu'nun ...../...../20... tarih ve ...../..... sayılı kararı.

İMZA/MÜHÜR

ÖZET

TEZİN BAŞLIĞI :**QUANSER I/O KART İÇİN MATLAB ALTINDA REAL-TIME LINUX-XENOMAI TARGET SÜRÜCÜ YAZILIMI GELİŞTİRİLMESİ VE UYGULAMASI**

YAZAR ADI : **EGEMEN CUMHUR KALELİ**

Bu çalışmada "RTXenoLab" olarak isimlendirilen gerçek zamanlı donanımlı simülasyon ve gerçekleme yazılımı tanıtılacaktır. Bu yazılım Linux/Xenomai çift çekirdek tabanlı işletim sistemi altında çalışmaktadır.

Tezdeki amaç, Linux/Xenomai işletim sistemi altında gerçek zamanlı çalışabilir donanım döngülü bir simülasyon yazılımı geliştirmek ve gerçeklemektir. Bu simülasyon yazılımı; Quanser Q8 veri edinim ve kontrol donanımının gerçek zamanlı sürücü yazılımından, Matlab/Simulink ortamında donanım döngülü gerçek zamanlı çalışabilir Simulink modellerinin oluşturulabilmesine olanak sağlayan Simulink kütüphanesinden, Simulink modellerinden gerçek zamanlı çalıştırılabilir kod üretilmesini sağlayan Linux/Xenomai gerçek zamanlı hedef platformu araçlarından ve bu gerçek zamanlı kodu çalıştırıp Simulink model parametrelerinin çalışma zamanı içerisinde değiştirilmesini ve model sinyallerinin gerçek zamanlılığa yakın biçimde izlenmesini sağlayan grafiksel ara yüz modüllerinden oluşur.

Windows işletim sisteminin güncelleme maliyetleri ve Matlab ile yeni Windows versiyonlarının uyuşma problemleri mevcut sistemlerin günümüz teknolojisine uygun hale getirilmesini ve gelecek teknolojik gelişmelere adapte edilmesini zorlaştırmaktadır. Bu çalışma ile Linux işletim sistemi tabanlı sürücü yazılımı ve bu yazılımın Matlab/Simulink'le entegrasyonu için gerekli yazılım araçları gerçeklenerek daha önce değinilen sorunlar ortadan kaldırılmıştır.

SUMMARY

TITLE OF THE THESIS : **IMPLEMENTATION AND APPLICATION OF MATLAB BASED LINUX-XENOMAI REAL-TIME DRIVER FOR QUANSER I/O CARD**

AUTHOR : **EGEMEN CUMHUR KALELİ**

This work introduces the design and implementation of "RTXenolab", a real-time hardware in the loop (H.I.L) simulation/implementation platform on Linux/Xenomai dual kernel based operating system.

The main aim of this thesis is to develop and implement a Linux/Xenomai operating system based HIL real-time simulation software which consists of a hard real-time driver software for Quanser Q8 data acquistion and control hardware, a dedicated Matlab/Simulink library for development of HIL real-time based Simulink models, a Linux/Xenomai real-time target platform that implements tools for generating real-time executable code from Simulink models, and a GUI software which runs this real-time based code and provides tools for manipulating Simulink model parameters and monitoring Simulink model signals in real time .

TEŞEKKÜR

Tez çalışmam süresince benden ilgi ve alakalarını hiç eksik etmeyen, her şart altında beni destekleyen aileme, her ihtiyaç duyduğumda kendilerini yanımda bulduğum akademisyen arkadaşlarıma ve hocalarıma teşekkürü bir borç bilirim.

Hakkını hiçbir şekilde ödeyemeyeceğimi bildiğim, yüksek lisans eğitimimi ve bu tez çalışmasını yaparken sürekli desteğini gördüğüm, bilgilerinden ve tecrübelerinden her zaman faydalı şeyler öğrendiğim sayın hocam Doç. Dr. Erkan Zergeroğlu'na beni bu günlere getirdiği için teşekkürü bir borç bilirim.

Saygılarımla.

Eylül 2011 Egemen Cumhur KALELİ

İÇİNDEKİLER DİZİNİ

KISALTMALAR VE SİMGELER DİZİNİ

Simgeler	Açıklama
RTXLB	RTXenoLab
HIL	Hardware In The Loop
MATLAB	Matrix Laboratory
Adeos	Adaptive Domain Environment For Operating Systems
RTDM	Real-Time Driver Model
POSIX	Portable Operating System Interface For Linux
PCI	Peripheral Component Interconnect
MHz	Mega Hertz
TLC	Target Language Compiler
Dc	Direct Current
GHz	Giga Hertz
KHz	Kilo Hertz
RTW	Real-Time WorkShop
GUI	Graphical User Interface
FLTK	Fast Light Tool Kit
CAPI	C Application Program Interface
GYTE	Gebze Yüksek teknoloji Enstitüsü
KRL	Kontrol ve Robotik Laboratuvarı

ŞEKİLLER DİZİNİ

1. GİRİŞ

1.1 Konu ve Önemi

Donanımlı simülasyon (ing: Hardware In the Loop, H.I.L), mekanik, elektronik, ve kontrol sistemlerinin dinamik davranışlarının anlaşılmasına ve fiziksel cihaz ya da makine modellerinin oluşturulmasında yardımcı olan yöntemlerden belki de en önemlilerinden biridir [1]. Çalışmada gerçeklenen RTXeno-Lab yazılımı öncesinde GYTE Kontrol Uygulamaları ve Robotik Labaratuvarında kullanılan donanımlı simülasyon sistemi, Windows RTX işletim sistemi, Matlab-Simulink görsel yazılım geliştirme platformu, Quanser Q8 donanımlı simülasyon amaçlı kontrol donanımı ile uç birimleri, gerçek zamanlı sürücü yazılımı ve bütün bu yazılım ve donanımların üzerinde koşturulduğu bir bilgisayardan oluşmaktaydı. Sahip olunan sistemi teknolojik açıdan yeniletip güncellemek Quanser Q8 kontrol donanımının Windows tabanlı sürücüsünün ve Windows işletim sisteminin periyodik olarak güncelleme zorunluluğunu beraberinde getirmektedir. Ancak bu güncellemelerin birbiri ile senkron olmaması, kısaca işletim sistemi güncellemeleri ile sürücü güncellemeleri arasında gecikmelerin yaşanması, her güncelleme sonrası üretilen kodların denenme sürelerinin belirsizlikleri ve bunlara ek olarak her güncellemenin bir maliyetinin olması yapılan deneysel çalışmaların zorluklarla ilerlemesine sebebiyet vermekteydi. Bu zorunluluğu ve Windows işletim sistemine bağımlılığı ortadan kaldırmak için RTXenoLab yazılım paketi geliştirildi.

Temel olarak RTXenoLab kullanıcının Simulink blok diyagramını standart bir bilgisayar üzerinde gerçekleyip tam gerçek zamanlı çalıştırmasına olanak verir. RTXenoLab; Linux / Xenomai ve Simulink/Real-Time WorkShop kütüphaneleri kullanılarak meydana getirilen kaynak kodlardan, Quanser Q8 I/O kartı için gerçeklenen sürücü kütüphanesinden, kod derleme dosyasından model kodu üretim dosyasından, sürücü kodu üretim dosyaları ve kullanıcı grafik ara yüzü kod dosyalarından meydana gelir. Kullanıcı grafik ara yüzü Simulink model blok

parametrelerinin çalışma zamanı içerisinde ayarlanmalarına ve model sinyallerinin izlenmesine olanak verir.

Bu çalışmada kullanılan yazılım paketleri için kısa tanıtım bilgileri aşağıda sunulmuştur.

1.2 MATLAB

MATLAB, kullanıcının kolaylıkla hesaba ve görselliğe dayalı deneylerinin birbirine entegre edilebildiği bir yazılım ortamıdır. MATLAB'in asıl avantajı problemlerin ve çözümlerin benzer matematiksel notasyonlar ile ifade edebilmesidir. Birçok araç kutusu ve farklı yazılım paketlerine sahip olan MATLAB bu özelliği sayesinde hesaplama, algoritma geliştirme, modelleme, simülasyon, veri analizi, görselleştirme, mühendislik grafikleri ve uygulama geliştirme gibi pek çok alanda kullanılabilmektedir[2].

1.3 Simulink

Simulink; MATLAB ortamında dinamik sistemlerin analizinde, modelleme ve simülasyon işlemlerinde kullanılan bir yazılım paketidir. Ayrık ve sürekli zamanda modellenen sistemler Simulink tarafından desteklenmektedir. Simulink içerisindeki grafiksel ara yüz blok diyagram modellerini meydana getirmek için kullanılır. Çeşitli blok kümesi kütüphaneleri önceden yapılandırılmış bloklar ve bağlayıcılar sağlar. Bu elemanlar basit çek ve bırak işlemleriyle modele dahil edilirler. Kütüphanelerde bulunan farklı tipteki sinyal sağlayıcıları ve sinyal istemcileri kullanıcının farklı girişleri uygulamasını ve çıkışları analiz etmesine olanak sağlar. Simulink'in bir diğer önemli özelliği modellerin hiyerarşik yapıda olmasıdır. Kullanıcı en üst seviyede sistem modelini görür ve isterse alt seviyelerdeki bloklara detayları görmek amacıyla ulaşabilir. Model tanımlandıktan sonra, kullanıcı seçtiği uygun zamanlama metotlarıyla sistem cevabının simülasyonunu gerçekleştirir. Simulink aynı zamanda çalışma zamanında parametre ayarlama ve simülasyon sonuçlarını izleme araçları

sağlamaktadır. Bunlara ek olarak, simülasyon sonuçları MATLAB çalışma alanına aktarılabilmektedir[2].

1.4 Real-Time WorkShop

Real-Time WorkShop; Simulink içinde bulunan otomatik C kodu üretim jeneratörüdür. Simulink modellerinden direkt olarak gerçek zamanlı çalışabilen kod üretir[2]. Bu kodun üretilmesi Target Language Compiler denen bir yorumlayıcı sayesinde gerçekleşir.

Matlab, Simulink ve Real-Time WorkShop yazılım paketleri ile ilgili daha detaylı bilgiler Matlab ağ sayfasından elde edilebilir [6].

Bu tezin geri kalan kısmı şu şekilde planlanmıştır:

Bölüm 2'de Linux/Xenomai çift çekirdekli işletimi sistemine gerçek zamanlı çalışma özelliğini kazandıran Xenomai çekirdeği ve Xenomai yazılım ortamı anlatılmıştır.

Bölüm 3'te RTXenoLab bileşenlerinden birisi olan Quanser Q8 gerçek zamanlı sürücü yazılımının gerçeklenmesi anlatılmıştır.

Bölüm 4'te RTXenoLab Simulink Kütüphanesi ve RTXenoLab grafiksel ara yüzü anlatılmıştır.

Bölüm 5'te RTXenoLab ile yapılmış deneysel uygulama anlatılmıştır.

Bölüm 6'da RTXenoLab ve Simulink'in karşılaştırıldığı simülasyon uygulaması anlatılmıştır.

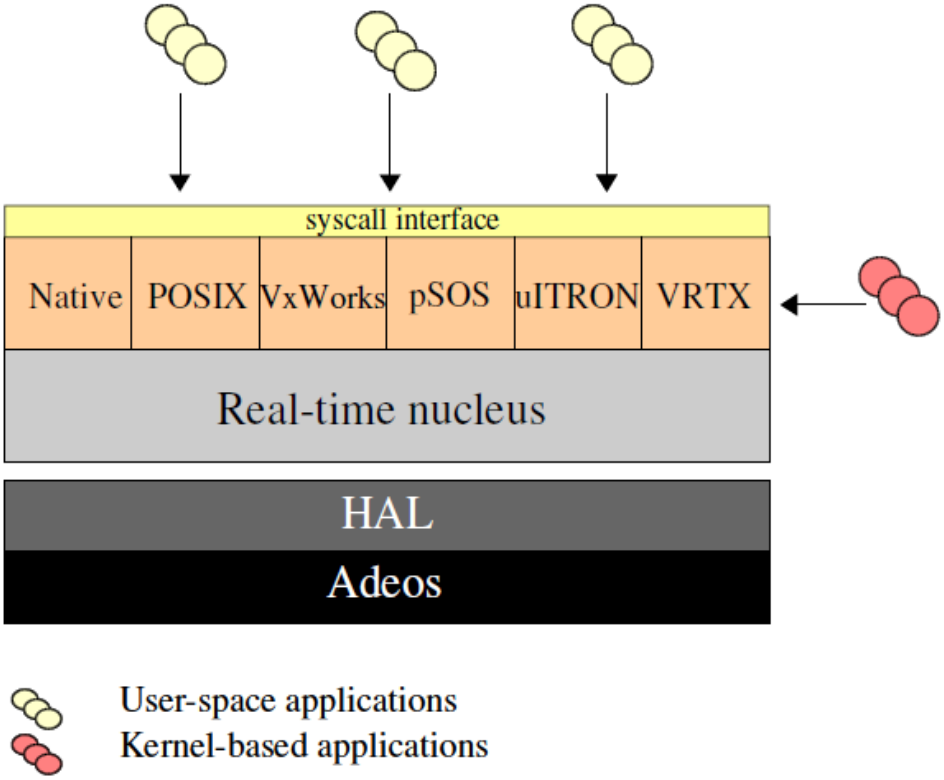
2. XENOMAİ

Xenomai, gerçek zamanlı çalışmaya olanak veren işletim sistemi çekirdeği ve uygulama geliştirme servislerinden oluşan bir yazılım platformudur. Kesme isteği yönlendirme katmanını kullanır (Adeos). Bu yaklaşımda bir kesme isteği oluştuğunda nano çekirdek Linux prosesleri ile gerçek zamanlı prosesleri birbirinden birbirinden soyutlar ve isteği bu iki koldan birine iletir. Eğer bu isteği alana gerçek zamanlı proses ise kesme kotarıcısı hemen çalışmaya başlar, diğer durumda istek bir kuyruğa alınır. Gerçek zamanlı bir proses çalışıyorsa prosesin bitmesi beklenir, bekleyen böyle bir proses yok ise kesme isteği Linux proseslere iletilir. Linux her ne zaman kesmelerin devre dışı bırakılması isteğinde bulunsa Linux / Xenomai'nin üzerine kurulu olduğu nano çekirdek Linux'e kesmelerin devre dışı bırakıldığını inandırır ancak kesme isteklerini gerçek zamanlı Xenomai tabanında çalışan proseslere aktarır. Linux ile ilgili kesme istekleri bir kuyrukta toplanır ve Linux kesmeleri devreye almak istediğinde bu kuyruktan kendisine iletilir[3].

Xenomai tasarımcıya gerçek zamanlı işletim sistemlerine özgün yazılım araçlarını sunar. Çoğu geleneksel yapıdaki gömülü işletim sistemlerinde bulunan proses akış kolları (threads) takvimlemesi ve senkronizasyon servislerini barındırır. Bu servislerle gerek çekirdek gerekse kullanıcı uzayında gerçek zamanlı uygulamala programları geliştirilir.

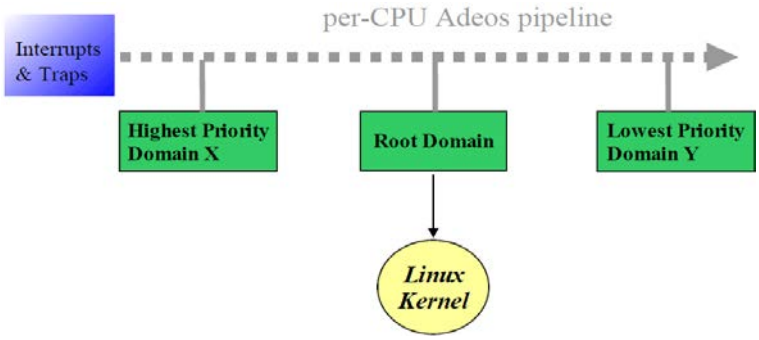
2.1 Xenomai'nin Mimari Yapısı

Xenomai proseslerinin tam gerçek zamanlı (hard real time) olarak çalışabilmesi için gerçek zamanlı uygulama arayüzü yardımcı çekirdeği kullanılır[4]. Bu çekirdek sayesinde gerçek zamanlı prosesler Linux altında çalışabilir. Linux prosesleri gerçek zamanlı proseslerden daha düşük öncelik seviyelerine sahiptir. Bu yardımcı çekirdek (co-kernel) donanım kesmelerini, yazılım kesmelerini (exceptions/traps/) ve hata kesmelerini (faults) Adapte Olabilen İşletim Sistemi Platformu (Adaptive Domain Environment for Operating Systems) katmanını kullanarak Linux çekirdeğiyle paylaşır.



Şekil 2.1: Xenomai'nin Mimari Yapısı (Life with Adeos 2005)

Şekil 2.1'de Xenomai'nin mimari yapısı görülmektedir. Adeos bir olay boru hattıdır. Adeos'un amacı hafıza, işlemci, giriş/çıkış çevre birimleri gibi donanım kaynaklarını işletim sistemleri arasında paylaştırmak için esnek bir platform sağlamaktır. Sistem olaylarını (donanım kesmesi, yazılım kesmesi, sistem çağrıları) zamansal ve öncelikli yapıda boru hattında bulunan işletim sistemlerine ulaştırır.



Şekil 2.2: İşletim Sistemleri Adeos Boru Hattı (Life with Adeos 2005)

Şekil 2.2'de Adeos boru hattında borunun başında bulunan işletim sistemi (domain) en yüksek önceliğe sahiptir. Bu yapıda Xenomai çekirdeği en yüksek önceliğe sahip işletim sistemi olarak boruya yerleşir. Kök ya da ana işletim sistemi Linux çekirdeğidir. Diğer tüm işletim sistemi çekirdekleri çekirdek modüllerinin yüklenebilmesi için Linux'e ihtiyaç duyarlar. Adeos yapısının kendisi de Linux'e derleme esnasında yerleştirildiğinden Linux'e bağımlıdır.

2.2 Xenomai Uygulama Geliştirme Kütüphaneleri

Xenomai, gerçek zamanlı uygulama ve donanım sürücüleri yazılımların geliştirmek amacıyla üç temel kütüphaneyle çalışma imkanı sunar: Native Xenomai Kütüphanesi, Gerçek Zamanlı Sürücü Kütüphanesi (RTDM), POSIX (Portable Operating System Interface For Linux). Bundan sonraki üç bölümde bu kütüphanelerden kısaca bahsedilecektir.

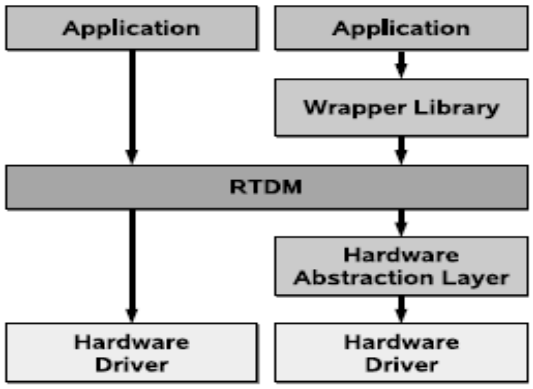
2.2.1 Native Xenomai Kütüphanesi

Native API, Xenomai'nin gerçek zamanlı uygulamalar ve prosesler arası ya da proses içi kod akış kolları (thread) arasında haberleşmenin sağlanabilmesi gerekli fonksiyonlar ve veri yapılarını kullanıcının hizmetine sunar. Bunlar: semafor, kuyruk (queue), paylaşımlı hafıza ya da tek başına kullanım için hafıza yığın yapıları, boru (pipe), gerçek zamanlı thread oluşturma servisleri, alarm ve signal servisleri, olay, zamanlayıcı ve kesme yönetim servisleridir.

2.2.2 Gerçek Zamanlı Sürücü kütüphanesi (RTDM)

Xenomai gerçek zamanlı sürücü kütüphanesi (RTDM), bilgisayar donanımlarının gerçek zamanlı prosesler tarafından gerçek zamanlılık ve determinizm prensibine uygun bir şekilde kullanımını sağlayan veri yapıları ve fonksiyonları içerir. RTDM iki tip donanım sürücüsü gerçeklemeyi desteklemektedir. Bunlar : Protokol donanımları ve isimli donanımlardır. Protokol donanımları, tüm mesaj tabanlı donanımları içerir. Ağ haberleşmesi için kullanılan donanımlar bunlara örnek olarak gösterilebilir. Bu tip donanımlar protokol ailesi ve

soket tipiyle birlikte sisteme kayıt edilir. POSIX soket modeliyle işlem gören bu donanımlar "socket()" fonksiyonuyla sistemde erişilebilir bir referansa sahip olurlar ve "close()" komutuyla silinirler. İsimli donanımlar ise metin isimleriyle kayıt edilirler."open()" ve "close()" fonksiyonlarıyla sistemde ulaşılıp silinebilirler. Donanımla veri alış verişi ise protokol donanımlarında sendmsg(), recvmsg(); isimli donanımlarda ise read(), write() fonksiyonları gerçekleştirir. Bu fonksiyonların RTDM versiyonları rtdm_device veri yapısı içerisinde sistem çağrısı kotarıcısı olarak bir çekirdek modülü içerisinde kayıt edilir. Çekirdek modülü içerisindeki başlangıç ayarlarını yapan fonksiyon içerisinde bu veri yapısı rtdm_dev_register(…) fonksiyonu vasıtasıyla işletim sistemi veri yapılarına kayıt edilir. Çekirdek modülü sistemden silinirken aynı veri yapısı kayıttan düşülerek gereğinde sürücü yazılımı sistemden çıkartılır. RTDM kullanıcı uzayındaki proseslerin çekirdek modül şeklinde gerçeklenen sürücü modülü içerisindeki fonksiyonları kullanıcı uzayı kütüphane fonksiyonlarıyla sistem çağrıları meydana getirme yoluyla kullanır. Kullanıcı uzayından çekirdek uzayı içerisindeki sürücü katmanına erişim sürücüsü yazılan donanımın ismi ile sağlanır (isimli donanımlar).



Şekil 2.3: RTDM ve İlgili Katmanlar (Xenomai Ağ Sitesi)

2.2.3 POSIX Kütüphanesi

POSIX kütüphanesi gerçek zamanlı olmayan prosesler içerisinde thread yaratmaya yarayan ve thread/prosesler arası haberleşmelerin sağlanabilmesi için gerekli fonksiyon ve veri yapılarını içerir. Bu kütüphanenin Xenomai ile birlikte

kullanılması ile gerçek zamanlı uygulamalar ile gerçekleştirmek mümkündür. Gerçek zamanlı olmayan bir uygulamaya gerçek zamanlılık özelliğini sonradan kazandırmak da mümkün olabileceğinden tasarım süreci kısalmaktadır. Xenomai Native kütüphanesi içerisindeki `rt_task_shadow(…)` fonksiyonu standart bir POSIX thread içerisinde çağrılacak olursa bu thread gerçek zamanlılık özelliği kazanır. Dikkat edilmesi gereken husus derleme zamanında POSIX bayraklarının ayarlanmasıdır.

3 QUANSER Q8 KONTROL DONANIMI İÇİN GERÇEK ZAMANLI SÜRÜCÜ YAZILIMI

Bu bölümde Q8 kartı tanıtılacak, sürücü yazılımının iki ana parçası olan çekirdek ve kullanıcı modülleri anlatılacaktır. Bu modüller Xenomai RTDM kütüphanesi kullanılarak gerçeklenmiş ve giriş bölümünde bahsedilen RTXenoLab yazılım paketini oluşturan ana parçalardan bir tanesini oluşturmuştur.

3.1 Quanser Q8 Kontrol Donanımı

Quanser Q8 kontrol donanımı, Quanser firması tarafından üretilmiş donanımlı simülasyon sistemlerinde kullanılan üzerinde PCI slot bulunan standart bir bilgisayar ana kartına takılabilir özellikte çevresel arabirimdir. PCI veri yoluna takılan PCI kartı ve uç biriminden oluşur.

Şekil 3.1: Quanser Q8 Kontrol Donanımı Uç Birimi

Şekil 3.1'de Quanser Q8 kontrol donanımı uç birimi gösterilmektedir. Q8; sekizer adet programlanabilir dijitalden analog işarete çevirici arabirimi, analogdan dijital işarete çevirici arabirimi, enkoder arabirimi taşır. Bunun yanında otuz iki bitten oluşan programlanabilir dijital giriş/çıkış birimine, çift kanal darbe genişlik modülasyonlu sinyal jeneratörüne, bekçi köpeği zamanlayıcı çıkışına ve standart zamanlayıcı çıkışına sahiptir. Ayrıca zamanlayıcı ve analogdan dijitale çevrim ara yüzü kesme istekleri üretir. Bilgisayar donanımı ile 33Mhz PCI veri yolu aracılığıyla haberleşir. Motor kontrol, mekatronik sistemler, robotik, güç elektroniği, kontrol

sistemleri, veri ve sinyal okuma gibi pek çok uygulama alanında ve hızlı prototipleme, donanımlı simülasyonun gerekli olduğu pek çok alanda kullanılabilen güçlü bir çevresel arabirimdir.

3.2 Sürücü Yazılımı Çekirdek Modülü

Linux işletim sisteminde çekirdek modülü olarak yazılan kullanıcı tanımlı sistem çağrısı kotarıcılarının bulunduğu dosyalar çekirdeğin bir parçası olarak Linux'e eklenebilir. Xenomai tabanlı çekirdek modülleri, Linux işletim sisteminin boru hattında ana işletim sistemi olmasından faydalanarak çekirdeğe işletim sisteminin bir parçası olarak eklenebilirler.

Quanser Q8 gerçek zamanlı sürücü fonksiyonları Linux / Xenomai çekirdek modülünde sistem çağrıları kotarıcıları olarak tanımlanarak sisteme kaydedildi ve gerçeklendi. Xenomai/RTDM kullanıcı uzayı gerçek zamanlı sürücü kütüphanesi fonksiyonları kullanıcı uzayı prosesleri içerisinde çağrılacak olduğunda, çekirdek modülü içerisinde kaydedilmiş ve gerçeklenmiş sürücü fonksiyonları sistem çağrısı kotarıcıları olarak görev yapmak üzere çağrılırlar.

RTDM, `ioctl_rt(…)` fonksiyonunu sistem çağrısı kotarıcısı olarak gerçeklendiğini gördüğünde bu sürücü modülünün değişik fonksiyonlara sahip bir donanım için gerçeklendiğini düşünür. Bu fonksiyonel yapı donanım sıfırlanması, kesme devreye alma/kapatma gibi işlevsel tanımlara sahip olabilir. Ancak `ioctl(…)` fonksiyonları yapı olarak switch/case kontrol döngüsünü kullanır. Genel olarak bu kontrol döngüsü C derleyicileri tarafından çok boyutlu ve statik özellikteki fonksiyon göstericileri dizisi şekline dönüştürülerek kod optimizasyonu yapılmış olur. Bu optimizasyonu derleyicinin insiyatifine bırakmak yerine tasarımcının yapmasının daha kararlı bir sistem oluşturulacağı düşünülerek switch/case kontrol döngüsü kullanmak yerine çok boyutlu ve statik özellikte fonksiyon göstericileri dizisi gerçeklenmiştir. Dizi içerisinde bulunan bu fonksiyon göstericileri Quanser Q8 çevresel birimlerine veri aktarmaya ve onlardan veri almaya yarayan fonksiyonların sistem belleğindeki adreslerini tutar. Sadece okuma ve yazma işlemleri yapılarak Q8

kontrol kartına ulaşılabilir. Bu amaçla tek bir sistem çağrısı kotarıcısı çekirdek modül içerisinde gerçeklenip ilgili `rtdm_device` veri yapısı içerisinde modülde kayıt altına alındı. Kullanıcı uzayından yapılan sürücü fonksiyonu çağrısı bu sistem kotarıcısına ulaşılacak çevresel arabirimin kanal numarasını analog -dijital dönüştürücü, dijital- analog dönüştürücü, enkoder birimleri için (0-…-7) aralığındaki değerler ve dijital giriş çıkış için (0-…-32) aralığındaki değerler fonksiyon göstericilerini tutan dizilere indis değeri olarak tanımlanır. Böyle bir metot izlemek hem switch/case kontrol döngüsünü kullanma durumunu ortadan kaldırır hem de Quanser Q8 üretici firmasının sağladığı veri yapılarını kullanma imkanını yaratır. Üretici firmanın sağladığı veri yapılarını kullanmak sürücü modülünün okunabilirliğini arttırmış hem de sistem çağrısı kotarıcılarının gerçeklenmesini kolaylaştırmıştır.

Kullanıcı modülünden gelen verilerin çekirdek modüle aktarılması ve çekirdek modülündeki değerlerin, Quanser Q8 kontrol donanımı yazmaçlarındaki değerlerin kullanıcı modülüne aktarılması RTDM'in sağladığı servisler sayesinde sağlanmaktadır. Bu servisler kullanıcı bilgilerinin saklandığı bir veri yapısını, okunacak ya da yazılacak veri yapısını tutan değişken referansını, bu referanstaki bilginin aktarılacağı değişkeni ve değişken büyüklüğünü parametre olarak alırlar. Çekirdek modülü içerisinde Q8 kontrol donanımının konfigürasyon ayarlarını değiştirmek ya da yeniden konfigürasyonu baştan yapmak için kullanılmak üzere bir veri yapısı tanımlandı. Bu veri yapısıyla tüm Q8 çevre birimlerinin kontrol ve giriş /çıkış yazmaçları temsil edildi ve yukarıda bahsedilen veri yapısını tutan parametre sistem çağrısı kotarıcıları içerisinde çekirdek ve kullanıcı modülleri arasında tampon vazifesi gören bu veri yapısına atandı. Böylelikle Q8 yazmaçlarının manipülasyonu ve okunması sağlanmış oldu.

PCI veri yoluna bağlanmış bir donanımın taban adres atanması sürücü yazılımları içerisindeki `_probe` fonksiyonlarıyla sağlanır. Q8 çekirdek modülünde, Q8 kontrol donanımına taban adresinin atanması, donanım adres uzayı için Linux çekirdeğinde hafıza tahsis edilip bu hafıza için bilgisayar giriş/ çıkış adres uzayında referans meydana getirilmesi, Q8 donanım kesmeleri için IRQ numarasının okunması amacıyla bir "probe" fonksiyonu gerçeklendi. Bu fonksiyon vasıtasıyla

çekirdek modülünün Linux / Xenomai çekirdeğine sokulmasından itibaren ana kart PCI slotlarında Q8 donanımın takılı olup olmadığı kontrol edilir. Böyle bir donanımın takılı olduğu anlaşıldığında Q8 çevresel ara birimlerinin başlangıç ayarları yapılır ve yukarıda değinilen işlemler gerçekleştirilir. Kartın bir Q8 donanımı olduğu çekirdek modülü içerisinde belirtilen üreticiyi tanımlayan bir numaranın Q8'in ilgili konfigürasyon yazmacındaki numarayla karşılaştırılmasıyla anlaşılır.

3.3 Sürücü Yazılımı Kullanıcı Modülü

Sürücü yazılımı kullanıcı modülü kullanıcı uzayı proseslerinin Q8 çekirdek modülü içerisindeki sistem çağrısı kotarıcıları aracılığıyla donanıma ulaşmasını sağlayan fonksiyonlardan oluşmaktadır. Bu fonksiyonlar Bölüm 3.1 Quanser Kontrol Donanımı'nda belirtilen çevre arabirimlerinin kullanıcılar tarafından erişimini ve manipülasyonunu sağlar. Kullanıcı modülü gerçeklenirken çekirdek modülüne erişimi sağlayan RTDM kullanıcı yüzeyi servisleri kullanılmıştır. Bu servislerden `rt_dev_open(…)` fonksiyonu çekirdek modülünde belirtilen donanım ismi vasıtasıyla donanıma erişim kapısını açar. Sırasıyla okuma ve yazma fonksiyonları olan `rt_dev_read(…)` ve `rt_dev_write(…)` fonksiyonları çekirdek modülünde belirtilen tampon veri yapısını parametre olarak kullanarak çekirdek modülle veri alış verişi gerçekleştirirler. Bu veriler ulaşılcak arabirimin kanal numarası, sinyal yazmaçlarına atanacak olan voltaj değerleri ve arabirim konfigürasyon ayarları bilgilerini içerir. Kullanıcı uzayı prosesi sonlandırılırken ya da Q8 donanımı ile iletişim kesilmek istendiğinde `rt_dev_close(…)` RTDM servisini kullanan fonksiyon kullanılır.

4 RTXenoLab

Bu bölümde Bölüm 3'te anlatılan Q8 veri edinim ve kontrol donanım sürücüsünün parçalarından birini oluşturduğu RTXenoLab donanımlı simülasyon (H.I.L) programı anlatılacaktır.

Q8 kontrol donamının ve diğer üçüncü parti donanımlarının genel bir donanımlı simülasyon programı altında birleştirilmesi amacıyla RTXenoLab donanımlı simülasyon programı geliştirilmiştir. Linux açık kodlu ve son zamanlarda populerite kazanmış bir işletim sistemi olmasına karşın gerçek zamanlı bir işletim sistemi değildir. Genel olarak Linux'e gerçek zamanlılık özelliğinin kazandırılmasında iki yaklaşım bulunmaktadır. Bu yaklaşımlardan ilki gerçek zamanlı proses yada proses içi işlemleri yönetmek için ikinci bir işletim sistemi çekirdeği kullanmaktır; Xenomai/ADEOS, RTLinux bu yaklaşımın birer çıktılarıdır. Diğer yaklaşım ise Linux çekirdeğinin kendisini düşük gecikme süreleri ve öncelik kazandırma gibi gerçek zamanlılık özellikleri bakımından güçlendirmektir [5].

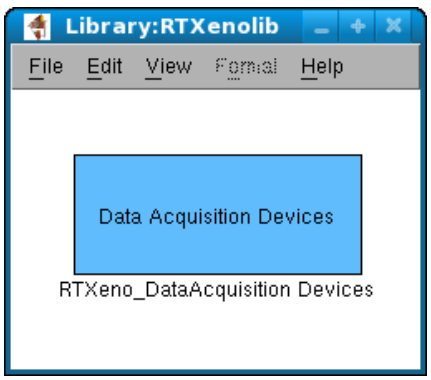
RTXeno-Lab yazılımının Linux altında gerçek zamanlı olarak çalışabilmesi Xenomai'nin gerçek zamanlı işletim sistemi çekirdeği sayesinde mümkün olmaktadır. Xenomai, çekirdeğin yanı sıra gerçek zamanlı uygulamaları geliştirmeye yarayan fonksiyon ve veri yapıları kütüphanelerini barındırır. MATLAB-Simulink-Real-Time WorkShop üçüncü parti paket yazılım kütüphanelerini kullanarak Simulink tabanlı gerçek zamanlı donanımlı simülasyon uygulamalarının gerçeklenmesini destekleyen RTXeno-Lab, kendine has grafik arayüzünü kullanarak Simulink modeli blok parametrelerinin kullanıcı tarafından çalışma zamanı içerisinde ayarlanabilmesine olanak sağladığı gibi model blok sinyallerinin izlenebilmesini sağlayan grafik pencerelerini kullanıcının hizmetine sunar.

4.1 RTXenoLab Devices Simulink Kütüphanesi

RTXenoLab, barındırdığı Simulink kütüphanesi sayesinde MATLAB/Simulink ortamında model tabanlı, gerçek zamanlı uygulamaların gerçeklenmesine olanak sağlar. RTXenoLab kütüphanesi, bir ana kütüphane ve bu kütüphane dahilinde bulunan veri edinim aygıtları alt kütüphanesinden oluşmaktadır. Kütüphane sayıları yapılacak uygulamaların çeşidine göre arttırılabilir. Kütüphanede Q8 kontrol donanımının ara birimlerini temsil eden gerçek zamanlı kullanıcı modülü fonksiyonlarını kullanan Simulink blokları bulunmaktadır. Bu bloklar aracılığıyla sürücü bloklarının kanal ve voltaj değerleri ayarlanır, konfigürasyon ayarları ve başlangıç ayarları yapılır.

4.1.1 RTXenoLab Ana Simulink Kütüphanesi

Ana kütüphane veri edinim aygıtları kütüphanesini barındırır. Kütüphane farklı işlevdeki Simulink bloklarını gruplandırmak için kullanılır.



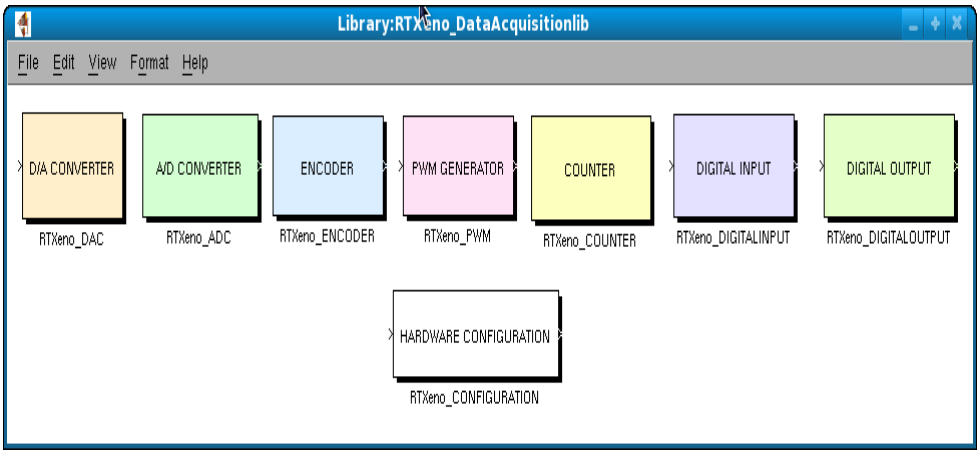
Şekil 4.1:RTXenoLab Ana Kütüphanesi

4.1.1.1 RTXenoLab Veri Edinim Aygıtları Simulink Kütüphanesi

Veri edinim aygıtları kütüphanesi, Linux/Xenomai tabanlı gerçek zamanlı yazılımı bulunan, PCI, USB, Ethernet ya da ISA veri yoluna sahip ve enkoder, sayısal/analog dönüştürücü, analog/sayısal dönüştürücü, sayıcı/zamanlayıcı, sayısal giriş/çıkış, darbe genişlik modülasyonu jeneratörü modüllerine sahip her hangi bir

veri edinim kartını destekleyebilecek şekilde tasarlanmıştır. Kullanıcı, kütüphanenin donanım konfigürasyonu blokunu kullanarak Simulink modelinde kullanmak istediği veri edinim aygıtını seçebilir. Örneğin Q8 kontrol donanımı kütüphanesini kullanmak isteyen kullanıcı bu kütüphaneye tıklağında ortaya çıkan Şekil 4.2'de gösterilen "HARDWARE CONFIGURATION" bloğu vasıtasıyla "Q8" seçeneğini işaretleyerek RTXeno-Lab Veri Edinim Aygıtları Kütüphanesini Q8 kontrol donanımı sürücü kütüphanesi olarak kullanabilir.

Kütüphanenin Simulink blokları ile seçilen veri edinim donanımının başlangıç ayarları yapılır. Farklı bloklar için farklı konfigürasyon seçeneklerinin oluşturulabilmesi için Simulink blok maskeleme yöntemi olarak adlandırılan yöntemle bloklar tıklandığında görünür hale gelen blok parametreleri ayarlama pencereleri ortaya çıkar. Sürücü konfigürasyonları sadece modelin ilk meydana getirilme zamanında ya da tekrar derleneceği zaman ayarlanır.



Şekil 4.2: RTXeno-Lab Veri Edinim Aygıtları Kütüphanesi

4.1.2 Kütüphanenin Gerçeklenmesi

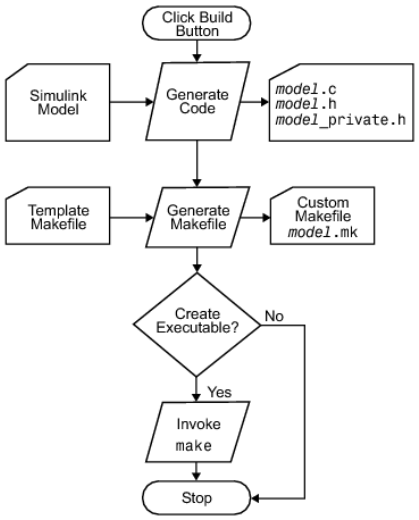
Kütüphanenin her bir bloğunun gerçeklenmesi için iki farklı adet kaynak kodun gerçeklenmesi gerekir. Bu dosyalar `<blockname>.c` ve `<blockname>.tlc` dosyalarıdır. C kaynak kodu bloğun kaç adet giriş ve çıkışa sahip olduklarını, giriş ve çıkışların boyutlarını belirleyen ve `<blockname. tlc>` dosyasına kontrol

donanımıyla ilgili parametrelerin aktarılmasını sağlayan fonksiyonları barındırır. C kaynak kodunun derleme esnasında herhangi bir işlevi bulunmaz. Hazırlanan C kaynak kodunun Simulink ortamı tarafından tanınması basit bir işlemle MATLAB EXECUTABLE haline dönüştürülmesiyle gerçekleştirilir.

Hedef dili derleyici dosyası (`<blockname.tlc>`) ise model blok diyagramının Real-Time Workshop Hedef Dili Derleyicisi (ing:Target Language Compiler) tarafından C koduna çevrilmesi için referans olarak kullanılır. Q8 ve gerçeklenecek diğer sürücü yazılımlarının kullanıcı modülü fonksiyonları bu dosya içerisinde çağrılır. C kaynak kodundan bu dosyaya geçirilen parametreler donanım sürücü fonksiyon parametrelerine argüman olarak aktarılır.

4.2 Simulink Modeli Blok Diyagramından Çalıştırılabilir Kod Üretilmesi

Simulink/Real-Time WorkShop programı Simulink modeli blok diyagramından C kodu üretir.



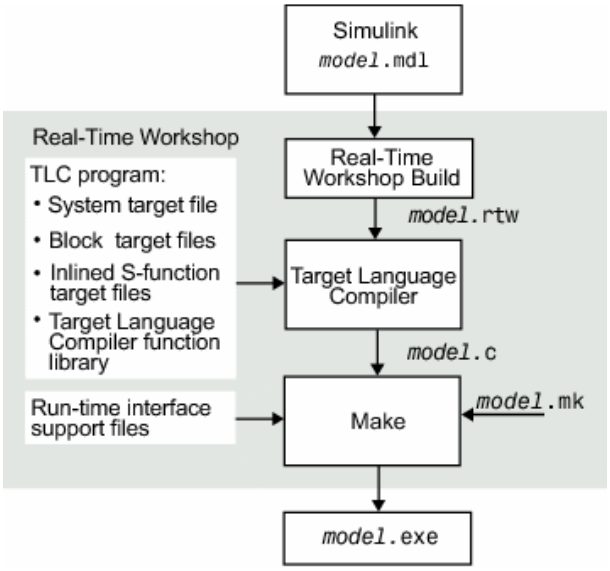
Şekil 4.3: Simulink Blok Diyagramından Çalıştırılabilir Kod Üretilmesi

(Matlab Yardım Dökümanı)

Şekil 4.3'te RTXeno-Lab ile çalıştırılcak kodun üretilme safhaları gösterilmiştir. İlk aşamada Simulink modeli blok diyagramından C programlama dilinde bir takım kaynak dosya TLC (Target Language Compiler) ile `*.tlc` ve `*.rtw` model dosyaları kullanılarak üretilir. Ardından gerçek zamanlı programın çalıştırılacağı hedef donanım/işletim sistemi ortamına uygun derleme dosyası üretilir. Bu derleme dosyasının üretimi için Simulink/Real-Time WorkShop'un sağladığı şablon derleme dosyaları kullanılır. Bu şablon derleme dosyası RTXeno-Lab/Xenomai ortamı için düzenlenir. Ardından derleme yapılarak çalıştırılabilir kod üretilir.

4.2.1 Simulink Blok Diyagramından C Kodu Üretilmesi

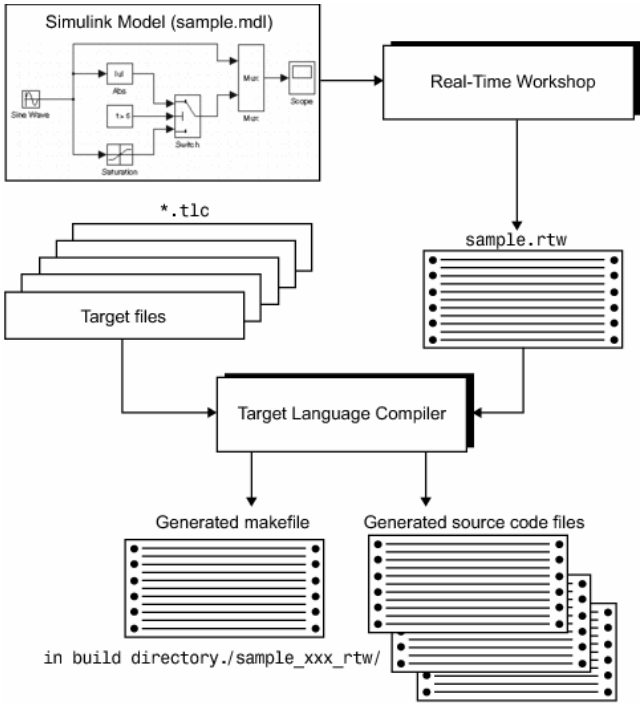
Çalıştırılabilir kod üretilmesi için öncelikle model diyagramdan C kodu üretilmesi gerekir. Bu C kodu üretilirken hedefe özel bir takım ayarların yapılması gerekmektedir. RTXeno-Lab; Simulink/Real-Time Workshop ortamına RTXenoLab çalışması kapsamında "Linux/Xenomai Real-Time Target" isimlendirilen hedef ortama göre üretilen C kodunu kullanır. Böyle özelleştirilmiş bir C kodu için tez kapsamında hazırlanan `linuxenomai.tlc,` genel model için sistem dosyası ve model bloklarını gerçekleyen `*.tlc` dosyaları kullanılır. Bu dosyalar TLC tarafından yorumlanarak hedefe özel C kodu üretilmiş olur. TLC tarafından modelin yorumlanabilmesi için öncelikle model özelliklerinin yazılı bulunduğu model.rtw dosyası Real-Time Workshop tarafından meydana getirilir. Bu dosya TLC' ye giriş olarak girer ve bir takım C kaynak dosyaları olarak çıkar.



Şekil 4.4: TLC İle C Kodu Üretilmesi

(Matlab Yardım Dökümanı)

Çalışma zamanında model sinyallerini izlemeye ve model blok parametrelerini ayarlamayı sağlayan kaynak kodu dosyaları ve hedefe özel çalıştırılabilir koda giriş noktası olan, içerisinde `main(…)` fonksiyonu barındıran kaynak kodlar Çalışma Zamanı Ara Yüzünü Destekleme Dosyaları (ing: Run-time interface support files) olarak isimlendirilir. Bu kodların bulunduğu dosyalar TLC tarafından üretilen model ve hedef bazlı kodlarla derlenerek gerçek zamanlı çalıştırılabilir dosya üretilmiş olur. Çalışma zamanı ara yüzünü destekleme kod dosyaları, main(…) içeren yürütülebilir dosyanın yanı sıra model parametrelerine erişimi sağlayan dosyalar harici modda Simulink modellerini çalıştırmaya ve çalışmada geliştirilen RTXenoLab gibi programların model blok parametrelerine ve sinyallerine erişimi sağlayan modüllerdir. RTXenoLab bu tip modüllerden "CAPI" modüllerini kullanır. Şekil 4.5'te TLC ile kod üretiminin aşamaları ayrıntılı olarak gösterilmiştir.



Şekil 4.5: Kod Üretim Hiyerarşisi

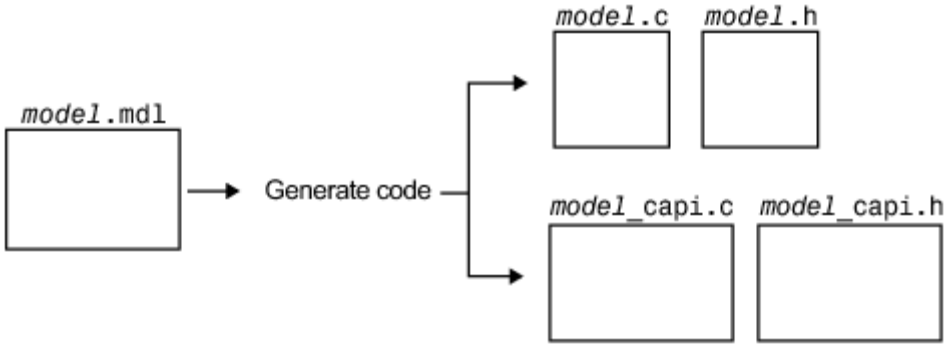
(Matlab Yardım Dökümanı)

4.3 Simulink Modeli Blok Parametreleri ve Sinyallerine Erişim

RTXeno-Lab Simulink model parametreleri ve sinyalleriyle kullanıcı arasında etkileşimi sağlamak için grafiksel ara yüz kullanır. Bu ara yüzün parametre ve sinyallere erişimi ise Real-Time WorkShop/CAPI kullanılarak sağlanır. RTXenoLab CAPI vasıtasıyla, model kodu çalışırken; model sinyallerini kayıt altına alma, sinyalleri izleme, parametreleri ayarlama özelliklerine kavuşmuştur.

Model parametre ve sinyallerinin sayısal değerleri ve konfigürasyon özellikleri, veri yapıları içerisinde tanımlanır. Bu veri yapıları parametre ve sinyal dizileri içerisindeki elemanlar olarak gerçeklenir. Parametre/Sinyal ismi, index numarası, sayısal değeri, veri boyutu, veri yönelimi, blok ismi/dizini, model ismi değişkenleri veri yapılarının üyeleri olarak tanımlanmıştır. Model konfigürasyon ayarlarından CAPI özelliği devreye alınırsa Real-Time Workshop `<modelismi>_capi.h` ve

`<modelismi>_capi.c` isminde iki tane ek dosya daha üretir. Veri yapıları dizileri bu dosyalarda gerçeklenmiştir. Şekil 4.6'da CAPI seçeneği konfigürasyon penceresinde aktif duruma getirildiğinde üretilen kaynak kodlar görülmektedir.



Şekil 4.6: CAPI Kaynak Kodları

(Matlab Yardım Dökümanı)

RTXeno-Lab içerisinde kullanılan CAPI veri yapıları dizileri aşağıdaki gibidir:

rtBlockSignals: Model içerisindeki tüm blokların çıkışlarından oluşan sinyallerin bilgilerinin bulunduğu dizidir. Dizideki her elemanın tipi `rtwCAPI_Signals` yapısı tipindedir. Yapının üyeleri; sinyalin ismini, adresini, data tipi indeksini, boyutu, sabit nokta(fixed_point) bilgisini sağlarlar.

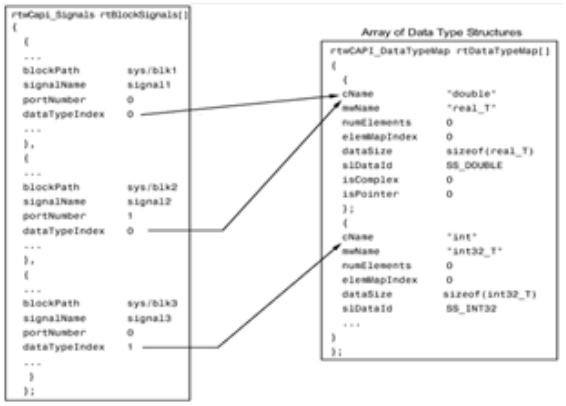
rtBlockParameters: Blok ismi ve parametre isimleriyle beraber model içerisindeki tüm parametrelerin bilgilerinin bulunduğu dizidir. Dizideki her elemenın tipi `rtwCAPI_BlockParameters` yapısı tipindedir. Yapının üyeleri; parametre ismini, blok dizinini, adresini ve data tipi indeksini , boyut ve sabit nokta bilgisini sağlarlar.

rtDataAddrMap: `rtBlockSignals` ve `rtBlockParameters` içinde gözüken sinyal ve parametrelerin taban adreslerini tutan dizidir. Bu dizideki her bir elemanın tipi void tipine bir göstericidir (void*).

rtDataTypeMap: Modeldeki değişik data tipleri hakkında bilgileri tutan veri yapılarından oluşan dizidir. Dizinin her bir elemanının tipi `rtwCAPI_DataTypeMap` yapısı tipindedir. Bu yapının üyeleri; data tipi ismi, data tipinin büyüklüğü, datanın kompleks sayı olup olmadığı bilgilerini tutar.

rtDimensionMap: Modeldeki değişik boyut tipleri hakkında bilgileri tutan veri yapılarından oluşan dizidir. Bu dizideki her bir elemanın tipi, `rtwCAPI_DimensionMap` yapısı tipindedir. Bu yapının her bir üyesi; verinin boyut sayısı, datanın skaler, vektör ya da matris olup olmadığı, verinin satır ve sütun sayısı bilgilerini tutar.

Şekil 4.7'de `rtBlockSignals` dizisindeki veri yapılarındaki üye değişkenlerle `rtDataTypeMap` dizisindeki veri yapıları arasındaki ilişki gösterilmiştir. Data tipi indeksi '0' olan bir sinyalin veri tipi verilerini öğrenmek için `rtDataTypeMap` dizisindeki '0' indeksli veri yapısının elemanlarına bakmak gerekir. Bu örnekte gösterilen şekilde tüm parametre ve sinyallerin veri tipi, adresi, veri boyutu ilgili dizinin ilgili indeksli elemanına ulaşılarak öğrenilebilir. CAPI bu dizilerdeki sinyal ve parametre verilerine ulaşmak için bir takım makrolar tanımlamıştır. RTXenoLab bu makroları kullanarak model hakkındaki gerekli bilgileri paylaşımlı hafızada saklar.



Şekil 4.7: CAPI Yapı Dizileri Elemanları Arasındaki İlişki

(Matlab Yardım Dökümanı)

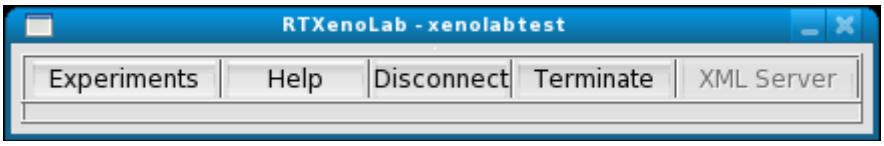
RTXenoLab kullanıcı arayüzünde çalışan prosesler hedef donanım/işletim sistemin ortamında bulunan model sinyal ve parametrelerine bu paylaşımlı hafıza yığını kullanarak erişirler.

4.4 RTXenoLab Kullanıcı Ara Yüzü (İstemci Program)

Kullanıcı ara yüzü kullanıcının fiziksel bir donanımın dahil olduğu gerçek zamanlı bir simülasyonu başlatıp yönetebilmesi, Simulink model parametrelerini ve sinyallerini manipüle edebilmesi için gerekli ara yüz araçlarını kullanıcın hizmetine sunar. Arayüz kullanıcıdan gelen isteklerin hedef ortamdaki sunucu programa iletilmesinden, sunucu programdan bu isteklere karşılık gönderilen cevapların değerlendirilmesinden ve grafik çıkışlarının güncellenmesinden sorumludur. Kullanıcı ara yüzü FLTK C++ kütüphanesi kullanılarak gerçeklenmiştir.

4.4.1 Kontrol Paneli

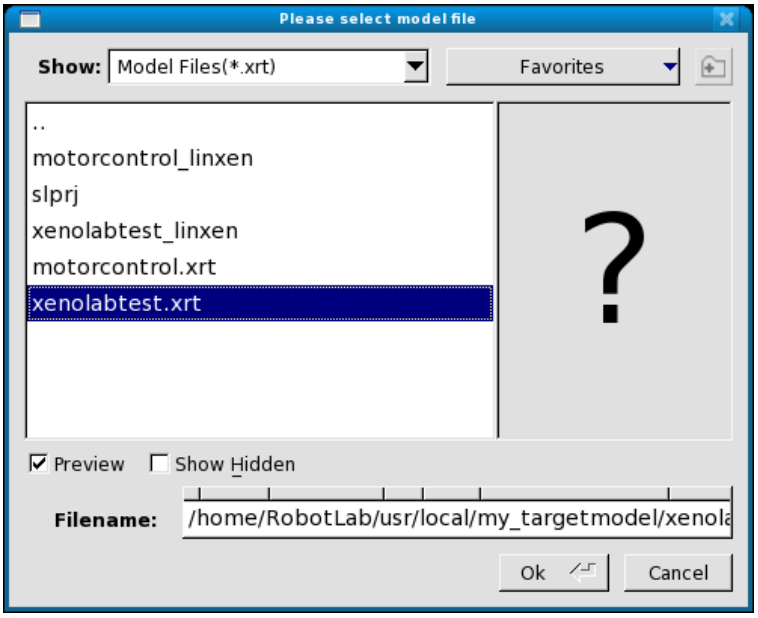
Kontrol paneli, `*.xrt` uzantılı çalıştırılabilir Linux/Xenomai-Simulink model kodunun bir dosya seçici vasıtasıyla bulunup çalıştırılabilmesini ve GUI ile koda erşim sağlanabilmesini sağlar. Şekil 4.8'de kontrol paneli görünmektedir.



Şekil 4.8: Kontrol Paneli

Kontrol paneli beş temel alt menü seçeneğinden meydana gelmektedir.

- **Experiments:** `*.xrt` uzantılı Linux/Xenomai-Simulink ortamında çalışabilen özellikteki kodu dosya sisteminde bulup çalıştırılabilmesini sağlar. Şekil 4.9'da dosya seçim penceresi görülmektedir.



Şekil 4.9: Dosya Seçim Penceresi

- **Terminate:** `*.xrt` uzantılı programı sonlandırır ve kontrol paneli dışındaki tüm pencerelerin kapanmasını sağlar.

- **Disconnect:** `*.xrt` uzantılı program ile GUI arasındaki veri alış-verişini keser.

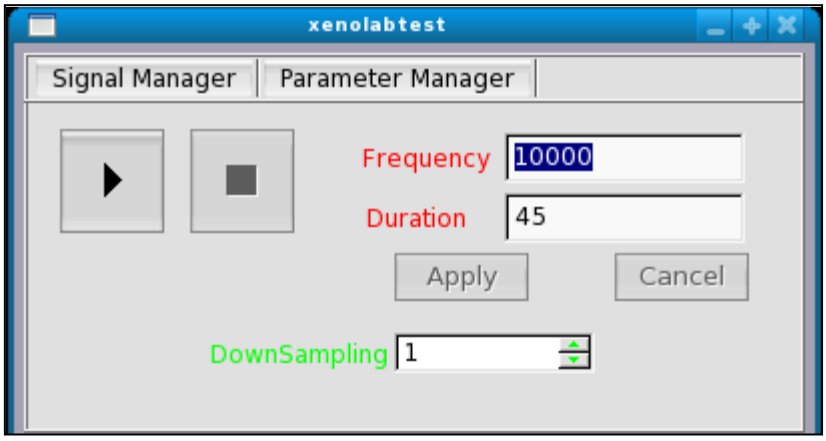
- **Help:** Programın çalıştırılabilmesi için gerekli temel bilgileri kullanıcıya gösteren pencereleri oluşturması için planlanan bu öğe aktif değildir.

XML Server: Model programın çalışması boyunca (<`modelname`>.xrt) XML formatında kaydedilen verinin internetteki hostlara ulaşmasını sağlayan sunucuyu çalıştırmak üzere planlanan bu öğe henüz aktif değildir.

4.4.2 Model Yönetim Penceresi

Model yönetim penceresi simülasyon süresinin , gerçek zamanlı iş parçacıklarının (thread) çalışma peryotlarının, sinyal kayıt frekansının ayarlanmasını sağlar. Bunun yanında `*.xrt` uzantılı Simulink model programının sinyallerinin

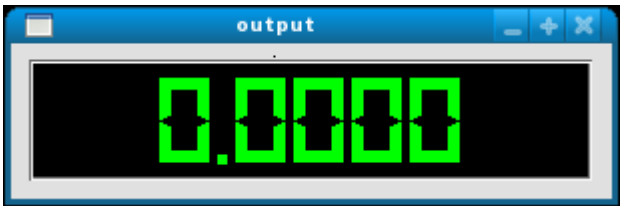
izlenmesini ve model programının parametrelerinin çalışma zamanında ayarlanabilmesini sağlayan pencereleri oluşturan menüye sahiptir.



Şekil 4.10: Model Yönetim Penceresi

Model yönetim penceresi “Signal Manager” ve “Parameter Manager” menülerini barındırır:

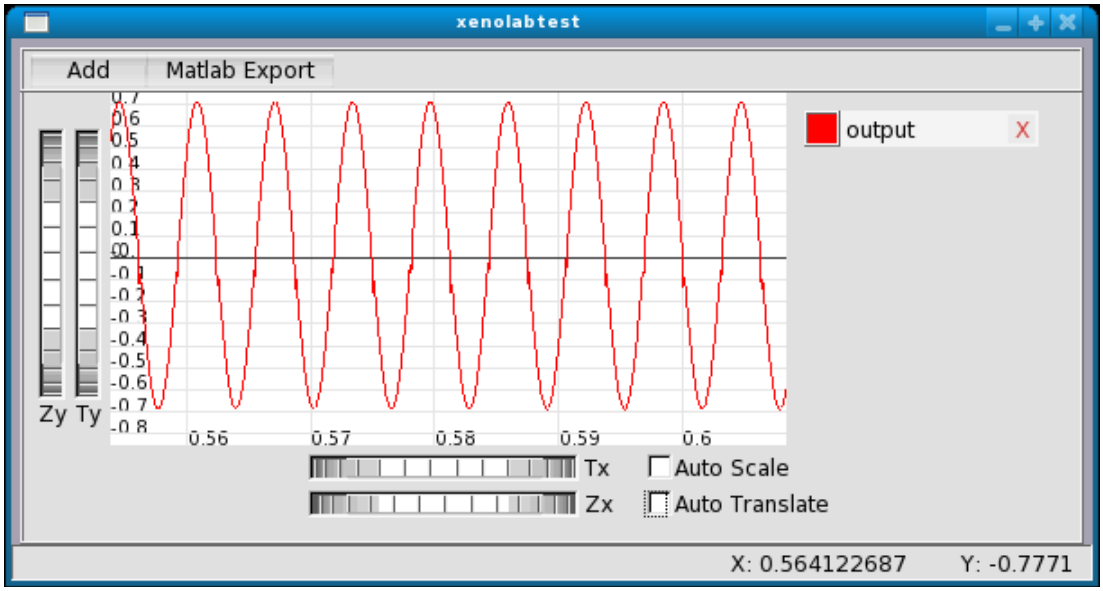
Signal Manager: Simulink model sinyallerinin 7 segment gösterge pencereleriyle gösterilmesini ve sinyallerin zamana göre değişiminin çizilmesini sağlar. Şekil 4.11, 7 segment gösterge pencere örneğini göstermektedir.



Şekil 4.11: 7 Segment Gösterge

Pencerenin başlığı olan “output” kullanıcı tarafından Simulink modelinin derleme sırasında sinyale verdiği isimdir. Kullanıcı istediği herhangi bir sinyali 7 segment göstergeye bağlayabilir.

Sinyal yönetim menüsü (Signal Manager) sinyal gösterimi için 7 segment gösterim penceresi dışında, model sinyal değerlerinin zamana göre değişiminin çizimini sağlayan "scope" pencerelerinin oluşturulmasına imkan sağlar. Şekil 4.12'de örnek bir "scope" penceresi görülmektedir.

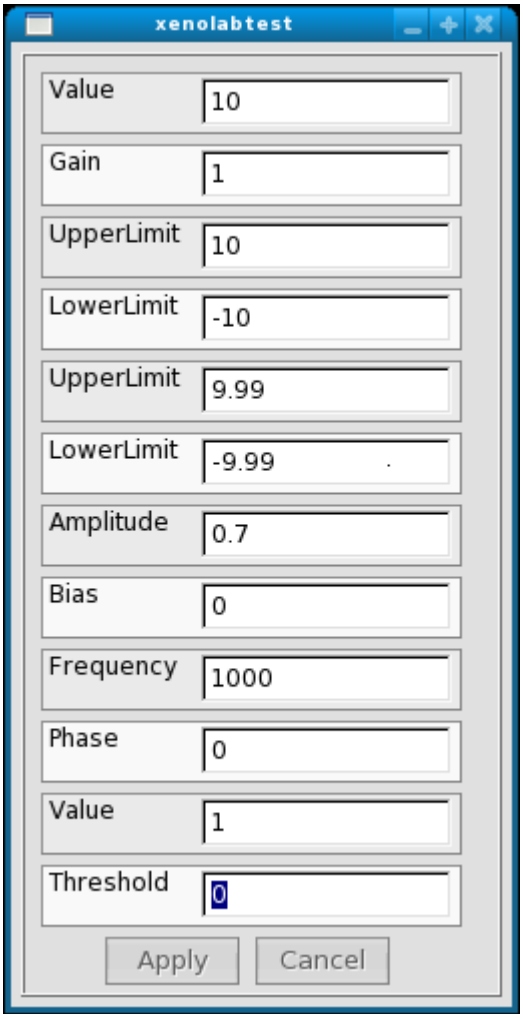


Şekil 4.12: Örnek Scope Penceresi

Scope penceresi üzerindeki GUI kontrolleriyle sinyalin değişik zaman aralıklarındaki değişiminin net olarak görülebilmesini sağlayan zaman kayması (time shifting), sinyal yakınlaştırma (zoom), sinyal boyutunun grafiksel değişimi ve zamana göre otomatik ölçeklendirilmesi yapılabilmektedir. "Add" düğmesi ile yeni bir sinyal değişkeni pencereye bağlanabilmekte ve "Matlab Export" düğmesiyle sinyal değişkeninin zamana göre değişimini veren veriler <*.m> uzantılı Matlab dosyası şeklinde kaydedilebilmektedir.

Parameter Manager : Parametre yönetim penceresi ile model blok parametreleri çalışma zamanı içerisinde ayarlanabilmektedir. Şekil 4.13'te örnek bir parametre penceresi gösterilmektedir. Şekilde görülen etiket isimleri bir Simulink model programındaki sinyallere aittir. "Apply" düğmesine basıldığında parametre değerlerindeki değişiklikler hedef ortamda çalışan sunucu programa

aktarılır.”Cancel” düğmesi ile parametre penceresindeki “apply” edilmemiş değerler silinir.



Şekil 4.13: Parametre Penceresi

4.5 Hedef Ortamda Çalışan Sunucu Program

Sunucu program GUI kontrol paneli tarafından çalıştırılan <`model name.xrt`> formatındaki derlenmiş, çalıştırılabilir programdır. Sunucu programın görevi, GUI ‘den gelen istekleri dinleyip bu isteklere karşılık gelen fonksiyonları icra etmek, Simulink model programını gerçek zamanlı olarak çalıştırmaktır. Sunucu program iki iş parçacığı icra eder. İlk iş parçacığı sunucu programla istemci (GUI) arasında senkronizasyonu sağlarken, ikinci iş parçacığı GUI’den gelen çalışma

frekansı ve süresi kıstaslarına göre Simulink model fonksiyonlarını icra eder. Program ilk çalıştırıldığında Simulink model parametreleri ve sinyalleri için paylaşımlı hafıza ayırır ve modelin başlangıç ayarlarını yapar.

Simulink modeliyle ilgili bilgiler MATLAB/Simulink/Real-Time WorkShop araclığıyla `RT_MODEL` türündeki veriyapısında saklanır. Bu bilgiler model parametreleri ve sinyallerle ilgili bir takım bilgiler ile modelin genel çalışması ile alakalı simülasyon süresi, çalışma frekansı ile ilgili verilerden oluşur. Sunucu program parametre ile ilgili olan bilgileri `Xrt_TargetParamInfo` veri yapısı içerisinde saklar.

```
typedef struct rtTargetParamInfo
        {
            char_T modelName[MAX_NAME_SIZE],
            char_T blockName[MAX_NAME_SIZE],
            char_T paramName[MAX_NAME_SIZE],
            int numRows,
            int numColumns,
            int dataType,
            boolean_T isComplex,
            rtwCAPI_Orientation orientation,
            double value[MAX_PARAM_SIZE]
       }Xrt_TargetParamInfo;
```

Sinyallerle ilgili bilgiler ise `SignalBuffer` ve `SignalBufferList` veri yapılarında saklanır. Bu yapılardan ilki her bir sinyal için bir adet oluşturulurken ikincisi bu ilk yapıdan kaç adet oluşturulduğuyla ilgili bilgileri tutar ve tek bir adet oluşturulur.

```
typedef struct signalBuffer_tag
    {

        char_T sigName,
        uint_T sigIdx,
        real_T slope,
        real_T bias,
        uint8_T slDataType,
        uint8_T dataType,
        boolean_T isComplex,
        rtwCAPI_Orientation orientation,
        uint_T numRows,
        uint_T updatedIndex
    }SignalBuffer;


typedef struct SignalBufferInfo_Tag{
        uint_T numSigBufs,
       uint_T numDataPoints,
        uint_T absNumSigBufs


    }SignalBufferList;
```

Sunucu program Xenomai'nin paylaşımlı hafıza oluşturma fonksiyonlarını kullanarak, yukarıda bahsedilen herbir veri yapısı için modeldeki sinyal yada parametre sayısı oranındaki hacimde paylaşımlı hafıza alanı oluşturur. Sinyal değerleri için bu oluşturulan hafızaya ek olarak sinyallerin reel kısımlarının kayıt altına alınmasını sağlayan eden `sigRes;` simülasyonun anlık değerini, simülasyonun bitiş zamanını, parametre sayısını, simülasyonun çalışma frekansı değerlerini tutan `controlHeap,` model ismini barındıran `nameHeap` ve simülasyon zamanını tutan `sigTime` paylaşımlı hafıza bölgelerini oluşturur. `sigRes ve sigTime` hafıza bölgelerinin hacmi sinyal değişkeninin veri türü, sinyal oryantasyonundaki eleman sayısı (skaler,vektör matris), bir sinyal için ayrılan kayıt sayısı ile doğru orantılıdır.

Sunucu program paylaşımlı hafızanın yanı sıra, istemci programla iletişimi sağlayan iki adet Xenomai tabanlı kuyruk yapısı ve bir adet Xenomai tabanlı olay (event flag group) yapısı barındırır.

Xenomai ile ilgli başlangıç ayarlarıyla birlikte Simulink model bilgilerini tutan `RT_MODEL` türündeki veri yapısı `INITIALISE_SIZES(RT_MODEL* rtM)`, `INITIALISE_SAMPLE_TIMES(RT_MODEL* rtM)` fonksiyonları ile derleme zamanında kullanıcı tarafından belirlenen parametrelerle yapılandırılır ve `START(rtM)` fonksiyonuyla birlikte model başlatılır. Model bilgilerine istemci program tarafından erişimi mümkün kılacak olan `rtwCAPI_ModelMappingInfo*` türündeki veri yapısı da gene sunucu program içerisinde `rtM` değişkeni vasıtasıyla yapılandırılır.

Sunucu program başlangıç ayarlarını yaptıktan sonra çalıştırdığı iki gerçek zamanlı Xenomai task vasıtasıyla host (Matlab/Simulink motorunun çalıştığı ortam) ile iletişim /senkronizasyonu ve Simulink blok diyagram modelini çalıştıran fonksiyonları çağırır. `OUTPUT(rtM)` fonksiyonu modeli çalıştıran asıl işleri yapan fonksiyondur. Bu fonksiyon `xrt_BaseTask` thread içerisinde kullanıcı tarafından grafiksel arayüze yazılan çalışma freakansı sıklığında çağrılır. `Xeno_SignalUpdate(…)` fonksiyonu `OUTPUT(rtM)`fonksiyonundan hemen sonra çağrılır ve sinyal değerleri paylaşımlı hafızaya kayıt edilir. Daha sonra `UPDATE(rtM)` fonksiyonu çağrılır. Bu fonksiyon kritik öneme sahiptir. Her bir çalışma çevriminde model yapısı içerisinde tanımlanmış zamanlayıcıyı ve modelin ayrık durumlarını günceller.

Modelin gerçek zamanda çalışması için çalışma frekansıyla belirlenen periyot diliminde `xrt_BaseTask(…)` içerisindeki tüm işler tamamlanır.

`xrt_BaseTask(…)`, sunucu programda `xrt_Main(…)` fonksiyonunda başlangıç ayarları yapıldıktan sonra yaratılıp çalıştırılan `xrt_HostInterface(…)` tarafından yaratılarak çağrılır. `xrt_HostInterface(…)`, grafiksel ara yüz istemci programdan gelen bilgilere ve `xrt_BaseTask'ın` `targetState` olarak tanımlanan çalışma durumuna göre `xrt_BaseTask'ı` çalıştırır, durdurur, duraksatır ya da çalışma süresini ayarlar.

4.6 Sunucu ve İstemci Programların Senkronize Çalışma Mekanizması

Bu bölümde RTXenoLab'ın çalışma mekanizması anlatılcaktır. RTXenoLab, daha önceki bölümlerde anlatılan Matlab/Simulink kütüphanesi, istemci program fonksiyonu gören grafiksel ara yüz ve hedef ortamda çalışan sunucu fonksiyonu gören programdan meydana gelir.

RTXenoLab kullanıcının grafiksel ara yüzünü çalıştırmasıyla başlar. Grafiksel ara yüz `<modelname.xrt>` isimli çalıştırılabilir hedef programı koşturmaya başlar. `<modelname.xrt>` Bölüm 4.9'da anlatılan başlangıç ayarlarını yaptıktan sonra `xrt_HostInterface(…)` taskı çalıştırır. Bu task grafiksel ara yüzde meydana getirilmiş Xenomai kuyruk yapılarına bağlanmak için beklemeye başlar. Kullanıcı ara yüzdeki "Connect" butonuna bastığında `Target` sınıfından bir obje yaratır. `Target` sınıfı, global olarak tanımlanmış Xenomai tabanlı paylaşımlı hafıza yığınlarını kullanarak hedef ortama (`<modelname.xrt>`) bağlanmak için gerekli fonksiyonlara, bu fonksiyonların döndürdüğü paylaşımlı hafıza referanslarını ihtiva eden veri gösterici üyelerine ve ara yüzün sonlandığını hedef programa haber veren kuyruk yapısını barındırır. `Target` objesi yaratıldıktan sonra bu objenin göstericisi `ModelManagerWindow` sınıfından bir objenin "constructor" fonksiyonuna parametre olarak geçirilir. `ModelManagerWindow` objesi grafik ara yüzdeki model yönetim penceresini meydana getiren objedir. Bu obje paylaşımlı hafıza bölgesine `Target` objesinin veri üyelerini kullanarak ulaşır. `ModelManagerWindow` paylaşımlı hafızadan model ismini öğrenir ve bu ismi kullanarak model parametre ve sinyallerinin bulunduğu paylaşımlı hafızaya bağlanır.

`<modelname.xrt>` hedef programı model isminin sağlıklı bir şekilde host programa ulaştırıldığını gelen kuyruk mesajıyla anladıktan sonra ara yüzden gelecek istekleri ve olayları dinlemeye başlar. Bu istek ve olaylar: `START_TARGET`, `STOP_TARGET`, `PAUSE_TARGET`, `UPDATE_DURATION`, `HOST_TERMINATED`.

- **START_TARGET:** Kullanıcı ara yüzdeki "START" butonuna basıldığında host/istemci/arayüz program Xenomai gerçek zaman tabanlı `MessageHandler` task/thread/iş parçacığını çalıştırmaya başlar. Görevi hedef/model koddan gelecek `xrt_BaseTask(…)`/Simulink-Xenomai simülasyon süresinin tamamlandığını işaret eden `TIME_ELAPSED` olayını beklemek ve model sinyal değerlerini 7 segment göstergeye yansıtmaktır (`TIME_ELAPSED` olayı gerçekleştiğinde `ModelManagerWindow` objesi `MessageHandler` iş parçacığını sonlandırır ve hedef programa "STOP" mesajı gönderir). `ModelManagerWindow` objesi `START_TARGET` mesajını hedefe gönderir. Hedef/`<modelname.xrt>/` `xrt_HostInterface(…)` task bu mesajı alınca eğer `targetState=STOP` ise `xrt_BaseTask` yaratılır ve ilgili fonksiyon koşmaya başlar. Eğer `targetState=PAUSE` ise `xrt_HostInterface` Xenomai tabanlı `rt_task_resume(…)` fonksiyonunu çağırarak `xrt_BaseTask'ın` kaldığı yerden çalışmasını sağlar.

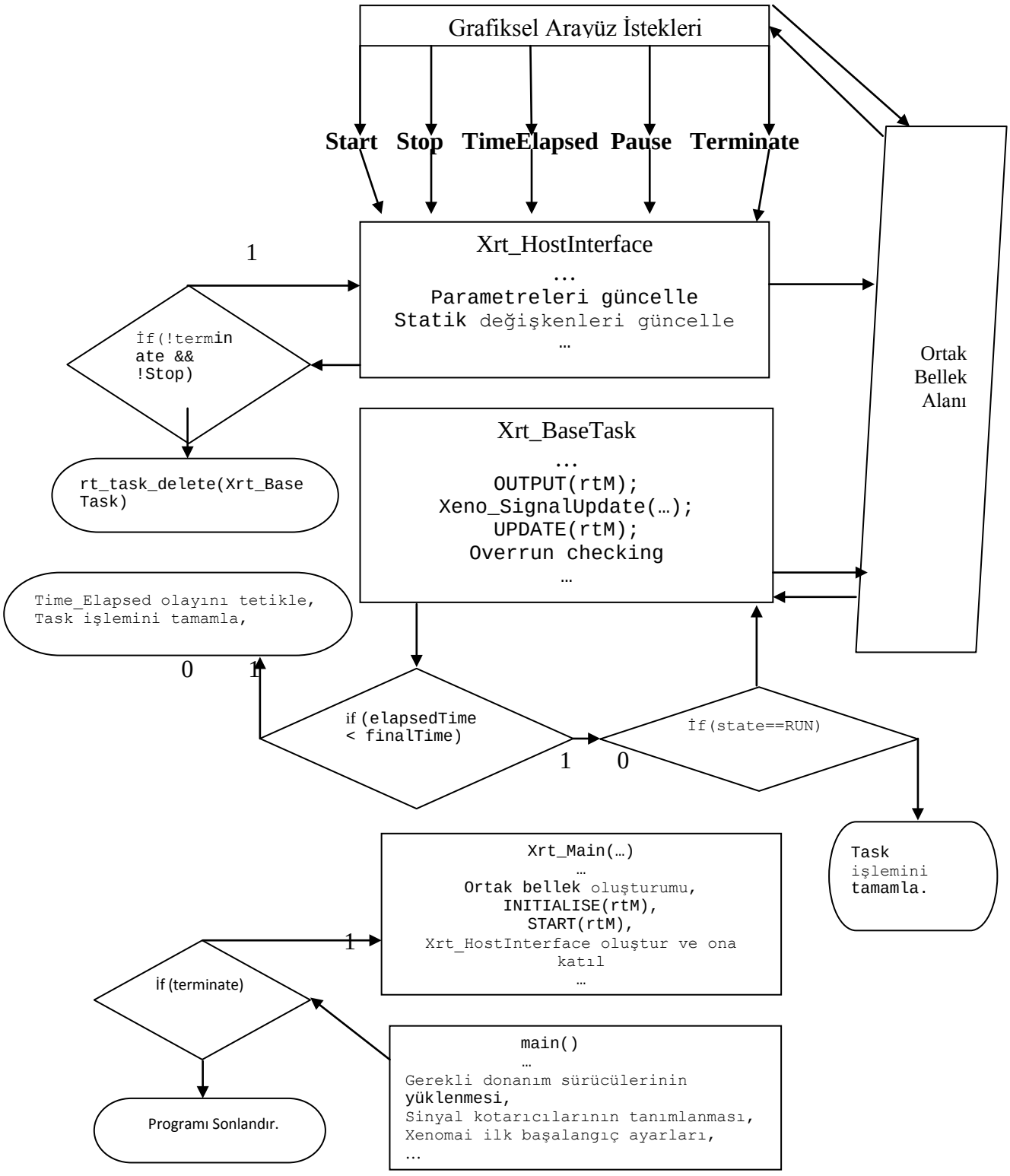
- **STOP_TARGET:** Bu istek kullanıcı ara yüzdeki "STOP" butonuna bastığında ya da `TIME_ELAPSED` olayı vuku bulduğunda gerçekleşir. `ModelManagerWindow` objesi `MessageHandler` iş parçacığını sonlandırır ve hedef programa "STOP" mesajı gönderir. `xrt_HostInterface, xrt_BaseTask` iş parçacığını siler. `xrt_HostInterface` kendini sonlandırır, kontrol `xrt_Main(…)` fonksiyonuna aktarılır. Eğer "Terminate" tuşuna basılmamışsa `xrt_Main(…)`tekrar çağrılarak program paylaşımlı hafızaların oluşturulma işlemini yapmadan baştan çalışmaya başlar.

- **PAUSE_TARGET:** Bu istek kullanıcının "PAUSE" butonuna bastığında gerçekleşir. `xrt_HostInterface` bu mesajı aldığında Xenomai tabanlı `rt_task_suspend(…)` fonksiyonunu çağırarak `xrt_BaseTask'ı` duraksatır.

- **`UPDATE_DURATION:`** Bu istek model yönetim penceresindeki simülasyon "duration" girişi değiştirilip "apply" düğmesine basıldığında gerçekleşir. `xrt_HostInterface` bu mesajı aldığında `controlHeap` paylaşımlı hafızasından simülasyon süresini alır, `rtmSetTFinal(rtM,duration)` makrosunu kullanarak model bilgilerini günceller.

- **`HOST_TERMINATED:`** Ara yüzdeki "Terminate" butonuna basıldığında `Target` objesi sistemden silinirken `~Target()` fonksiyonu ara yüzün kapatılacağı mesajını hedef koda kuyruk aracılığıyla gönderir. `xrt_HostInterface` önce `xrt_BaseTask` iş parçacığını daha sonra ise kendini sonlandırır ve kontrol `xrt_Main(…)` fonksiyonuna aktarılır. `xrt_Main(…)` gerçek zamanlı bu iş parçacıklarının sonlanmasını `rt_task_join(…)` ile bekler ve ardından tüm program sonlanır.

Bu mekanizmanın işleyişi Şekil 4.14'te gösterilmektedir.

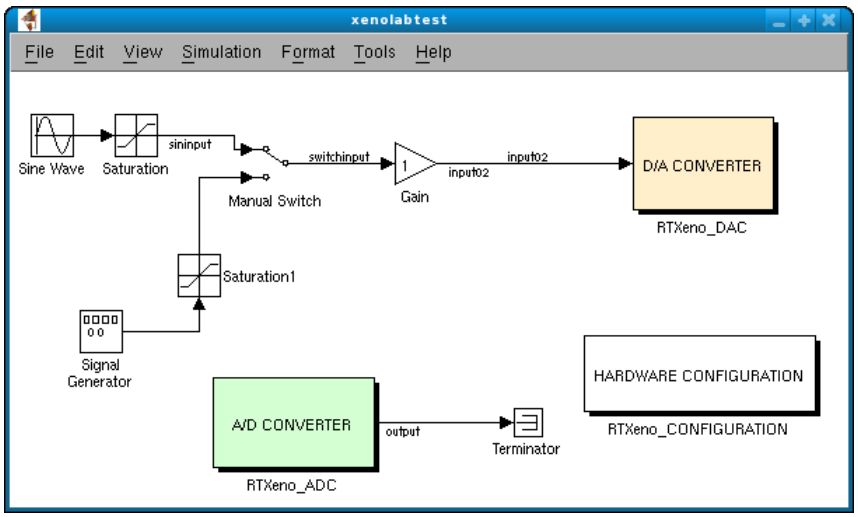


Şekil 4.14: RTXLB-Simulink Model Programı Sürekli Çalışma Akışı Detayı

4.7 RTXenoLab Simülasyon Programının Gecikme Süresinin Deneysel Bir Uygulamayla Gözlenmesi

Gerçek zamanlı uygulamalarda gecikme süresi çalışma/kontrol/task periyodunun süresini geçmemelidir. RTXenolab'ın çalışma performansını test etmek amacıyla Quanser Q8 kontrol donanımının sürücüsünü kullanan RTXenolab Simulink kütüphanesi kullanılarak Simulink'te bir model gerçeklenip çalıştırılmıştır. Bu bölümde bu uygulamanın sonuçları anlatılacaktır.

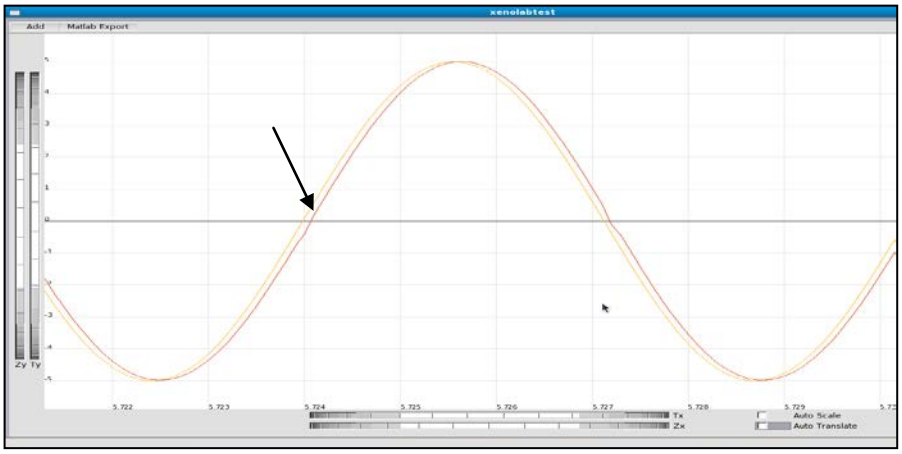
Şekil 3.1'de görünen Q8 terminalinin 0 numaralı dijital/analog çevirici çıkışı, 6 numaralı analog/dijital çevirici girişine takılmıştır. Simulink programında Şekil 4.15'te görülen Simulink "Linux/Xenomai Target" model oluşturulmuştur. Çalışma frekansı 10kHz'tir.



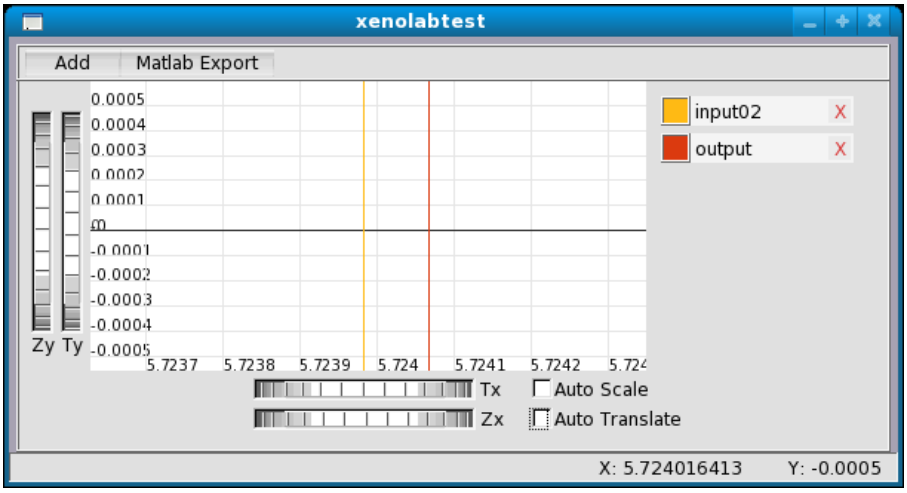
Şekil 4.15: Simulink Linux/Xenomai Target Test Modeli

Modelde her biri 1kHz. frekansında biri sinüs diğeri karedalga üreten iki sinyal üreteci bir manuel anahtar yardımıyla dijital/analog çevirici bloğuna giriş olarak verilmiş ve bu bloğun fiziksel çıkışı analog/dijital çeviricinin fiziksel girişine bağlanarak modelde görülen input02 ve output sinyalleri arasındaki faz farkı yani giriş/çıkış arasındaki gecikme gözlemlenmek istenmiştir. Bu iki sinyalin RTXenolab sinyal ekranındaki zamanla değişim grafiği girişin sinüs ve kare dalgalar olduğu durumlarda sırasıyla şekil 4.16, 4.17 ve şekil 4.18, 4.19'da görülmektedir. Şekillerde

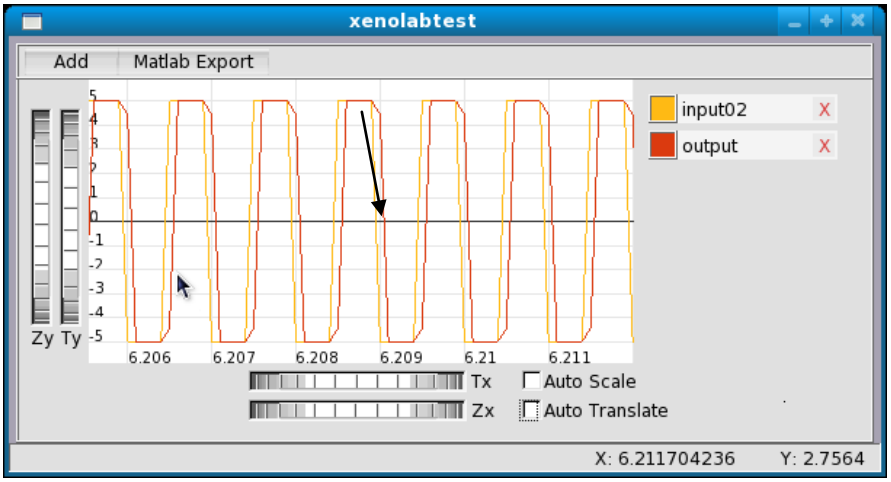
yatay eksen saniye cinsinden zaman eksenini ve dikey eksen işaretin büyüklük değerini temsil etmektedir. Sarı çizgi giriş sinyalini, kırmızı çizgi çıkış sinyalini göstermektedir. Okla gösterilen sinyal bölgesinde büyütülme yapılarak giriş çıkış arasındaki faz farkı daha net bir şekilde gösterilmiştir.



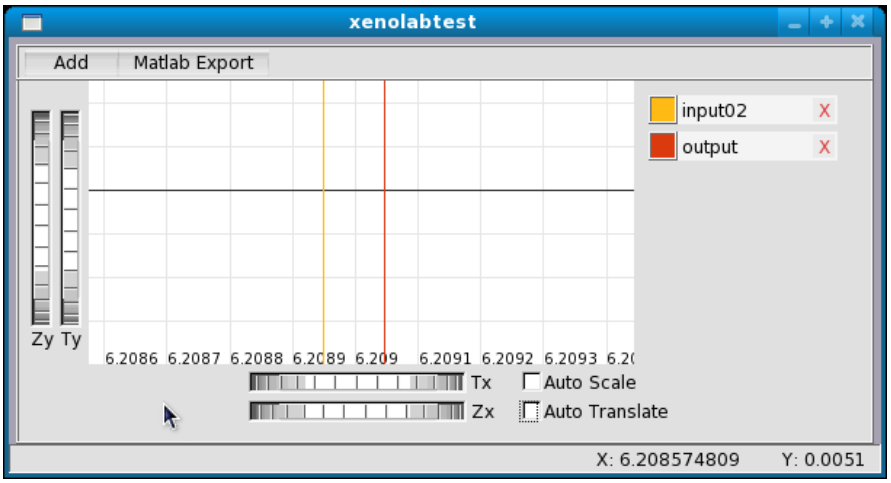
Şekil 4.16 : Test Modeldeki Sinüs Dalga Giriş ve Çıkış Sinyalleri



Şekil 4.17 : Şekil 4.16'daki Okla Gösterilen Grafik Bölgesi



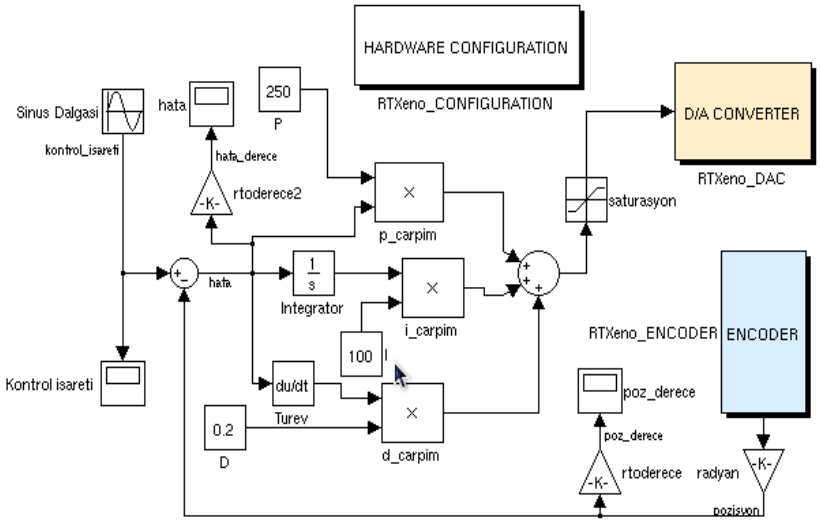
Şekil 4.18: Test Modeldeki Kare Dalga Giriş ve Çıkış Sinyalleri



Şekil 4.19: Şekil 4.18'deki Okla Gösterilen Grafik Bölgesi

5. DENEYSEL UYGULAMA

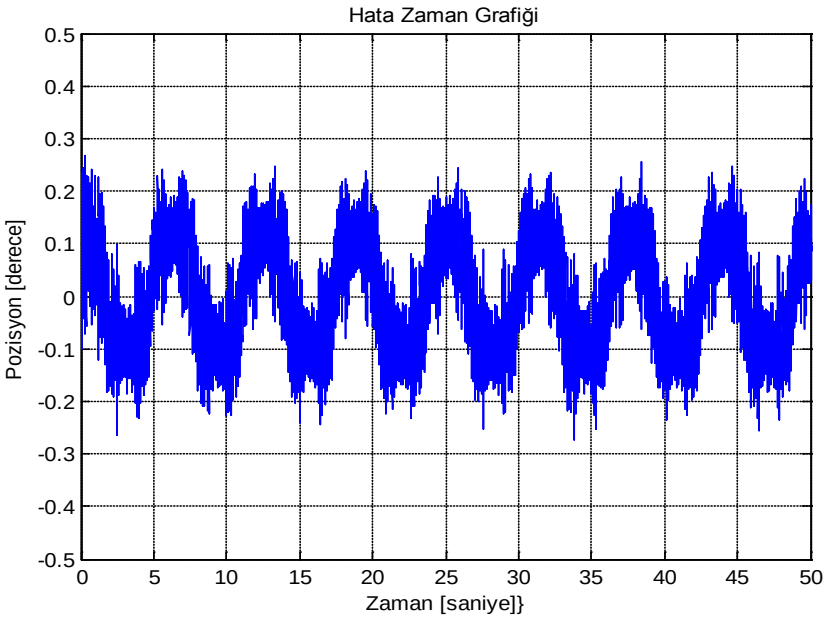
RTXeno-Lab'ın performansını test etmek amacıyla, bir fırçalı DC motorunun PID denetleyici kullanarak pozisyon izleme deneyi gerçekleştirildi. Deneye Şekil 5.1'de gösterilen Simulink blok diyagramı oluşturularak başlandı. Fiziksel deney düzeneği olarak da Şekil 5.2'de gösterilen deney düzeneğini kuruldu. Düzenek 12 V nominal voltaj değerine sahip enkoderli dc motor, Q8 kontrol donanımı ve uç birimi, Intel Quad işlemcili 2.5 GHz. bilgisayar, lineer sürücü ve anahtarlamalı güç kaynakları katından meydana gelmektedir[7].



Şekil 5.1: DC Motor Kontrolü Simulink Modeli

Şekil 5.2: Deney Düzeneği

Motor enkoderi Q8 kartının 5 numaralı enkoder girişine, motor besleme uçlarını lineer sürücünün konrol sinyali ve toprak uçlarına, lineer sürücünün kontrol girişine Q8 kartının 0 numaralı D/A çevirici çıkışına bağlandı. Simulink modelindeki HARDWARE CONFIGURATION bloğundan donanım olarak Q8'i seçilmiştir. D/A CONVERTER ve ENCODER bloklarından ilgili giriş çıkış kanalarını da seçtikten sonra Simulink Solver menüsünden örnekleme frekansı 20 KHz olarak ayarlandı ve RTW menüsünden sistem hedef dosyası seçtik (linuxenomai.tlc). Deney düzeneğimizde kullanıcıdan 20 KHz' lik denetim frekansı motor kontrol düzeneği için gereğinden fazla bir kontrol frekansı oluşturmaktadır. Ancak sunulan deneylerdeki temel amaç belirtilen motoru en iyi şekilde kontrol etmek dışında istenildiği durumlarda tasarlayıp üretilen yazılımın yüksek kontrol frekanslarına çıkabildiğini de göstermektir. Matlab komut penceresinden make_rtw komutu girildiğinde modelimiz için gerçek zamanlı platformda çalışabilecek kod üretip sonrasında da derlenmiş olur. Model kodunu konsoldan çalıştırıp modeldeki PID kontrolörün kazançalarını ayarlayarak motor şaftının pozisyon kontrolü yapılmıştır.. Şekil 5.3'de derece cinsinden pozisyon hatası verisi Matlab dosyası olarak kayıt edilerek çizdirilmiştir.



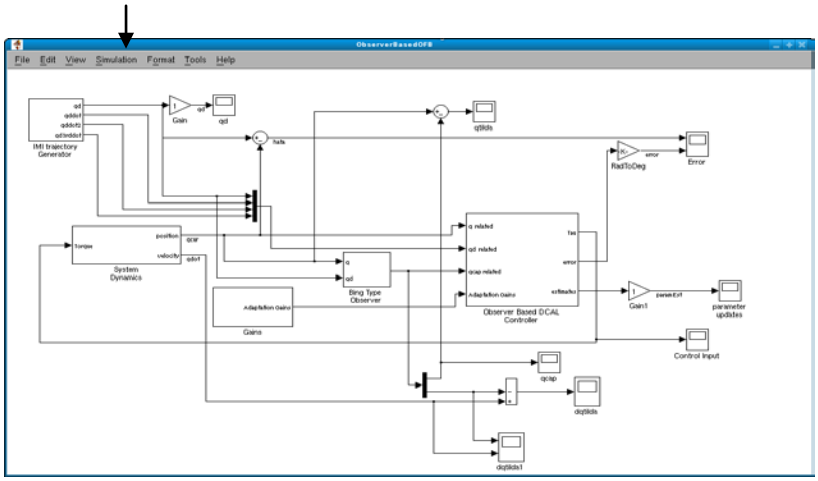
Şekil 5.3: Motor Şaftı Pozisyon Hatası Grafiği

6. SİMÜLATİF UYGULAMA

Bu bölümde, oluşturulan bir Simulink modelindeki sinyallerin çalışma zamanındaki değişimleri ile model parametrelerinin sinyal davranışları üzerindeki etkilerinin Matlab/Simulink grafiksel ara yüzü ile RtxenoLab grafiksel ara yüzündeki gösterimlerinin karşılaştırılması için yapılan simulatif bir uygulama ve ulaşılan sonuçlar anlatılacaktır. RtxenoLab platformunun daha iyi anlaşılması için simulasyonun çalışması için yapılan işlemler adım adım anlatılacaktır.

6.1 Uygulamada Kullanılan Simulink Modeli

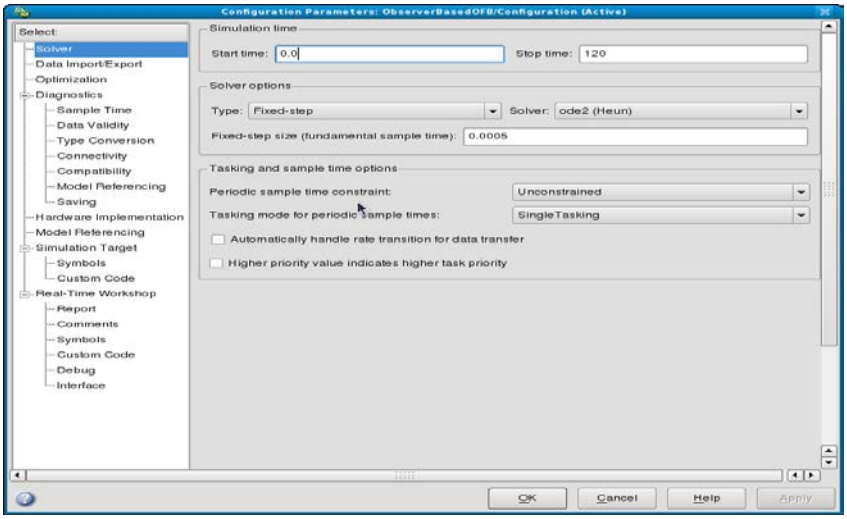
Uygulama için seçilen Simulink modeli iki linkli bir robot kolun Observer Based Output FeedBack kontrol algoritmasıyla kontrolünün simülasyonunun yapılmasını sağlamaktadır. Modelde iki linkli bir robot kolun dinamik modeli kullanılmıştır. Bu modelin uygulama için seçilmesinin nedenleri modelin görece karmaşık bir kontrol algoritması kullanması, birçok vektörel sinyal ve parametreye sahip olmasıdır. Şekil 6.1'de uygulamada kullanılan Simulink blok diyagramı görülmektedir.



Şekil 6.1: Observer Based Output FeedBack Simulink Modeli

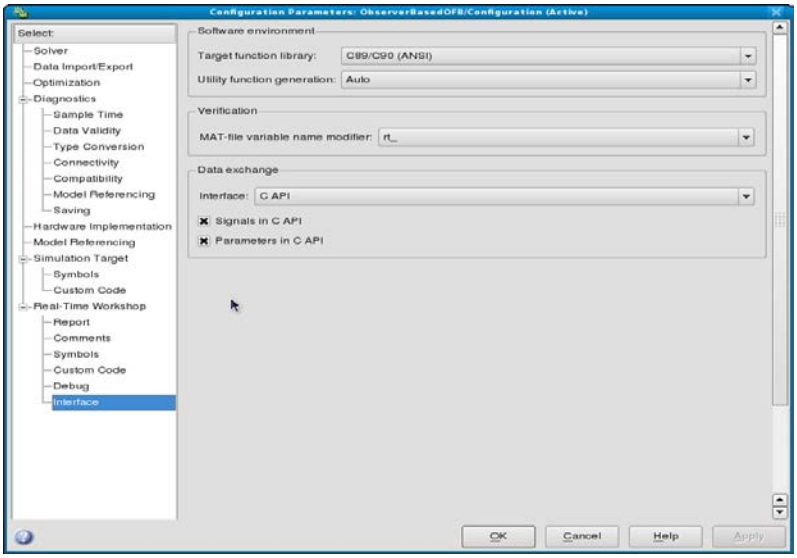
6.2 Simulink Modelinin RTXenoLab ile Çalıştırılması İçin Konfigürasyon Ayarlarının Yapılması

Model oluşturulduktan ya da kapatılıp tekrar açıldıktan sonra simülasyonun ne kadar süre ile çalışacağı, modelin çalışma/örnekleme frekansı gibi başlangıç ayarları yapılır. Bunun için şekil 6.1'de görülen model penceresinin üst tarafındaki menüdeki okla gösterilen "**Simulation**" menüsünden "**Configuration Parameters**" penceresi açılır. Bu pencereden şekil 6.2'de görülen **"Solver"** penceresi açılır**.**



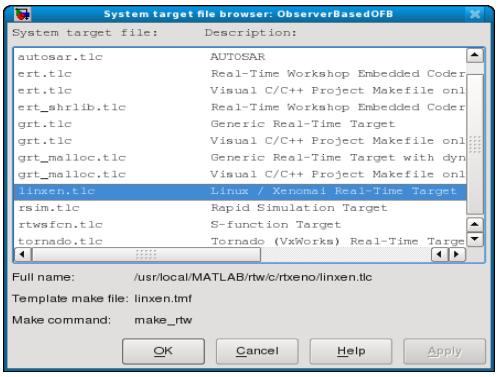
Şekil 6.2: Configuration Parameters Solver Penceresi

Bu pencereden simülasyon süresi 120 saniye ve simülasyon çalışma/örnekleme periyodu 0.0005 saniye(çalışma frekansı 2kHz.) olarak ayarlanmıştır. Bu ayarlamalar yapıldıktan sonra Şekil 6.3'te görülen "**Real-Time WorkShop/Interface**" menüsünden kullanılacak veri değişimi(data exchange) ara yüzü olarak Bölüm 4.3'te bahsedilen CAPI seçeneği seçilerek, "**Signals in CAPI**" ve "**Parameters in CAPI**" seçim kutuları işaretlenmiştir. Böylelikle Simulink model diyagramından RTW tarafından C kodu üretilirken sunucu ve istemci programlar arasındaki veri alış verişini sağlayan CAPI kaynak dosyaları da üretilecektir.



Şekil 6.3: Configuration Parameters/RTW/Interface Penceresi

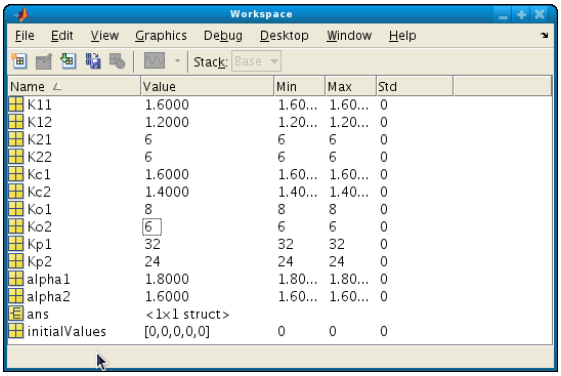
Model blok diyagramından üretilen C kodlarından Linux/Xenomai ortamında gerçek zamanlı çalışabilir kod üretmek için RTW'ye bunu başarabilmesi için hangi sistem dosyalarının kullanılması gereketiğinin "söylenmesi" gerekir. Bu amaçla hazırlanan ve Bölüm 4.2.1'de bahsedilen `linuxenomai.tlc (linxen.tlc)` sistem dosyası Şekil 6.4'te gösterilen dosya seçim penceresi aracılığıyla seçilir. `linxen.tlc` C kodu üretim parametrelerinin yanı sıra, oluşturulan kodun derlenmesi için hangi şablon derleme dosyasının (ing: template makefile) kullanılacağı bilgisini içerir. Şablon derleme dosyası olarak tez kapsamında hazırlanan `linxen.tmf` kullanılır.



Şekil 6.4 Configuration Parameters/RTW/System Target File Browser

6.3 Model Blok Parametrelerine Başlangıç Değerlerinin Atanması ,Kod Üretim ve Derleme İşlemi

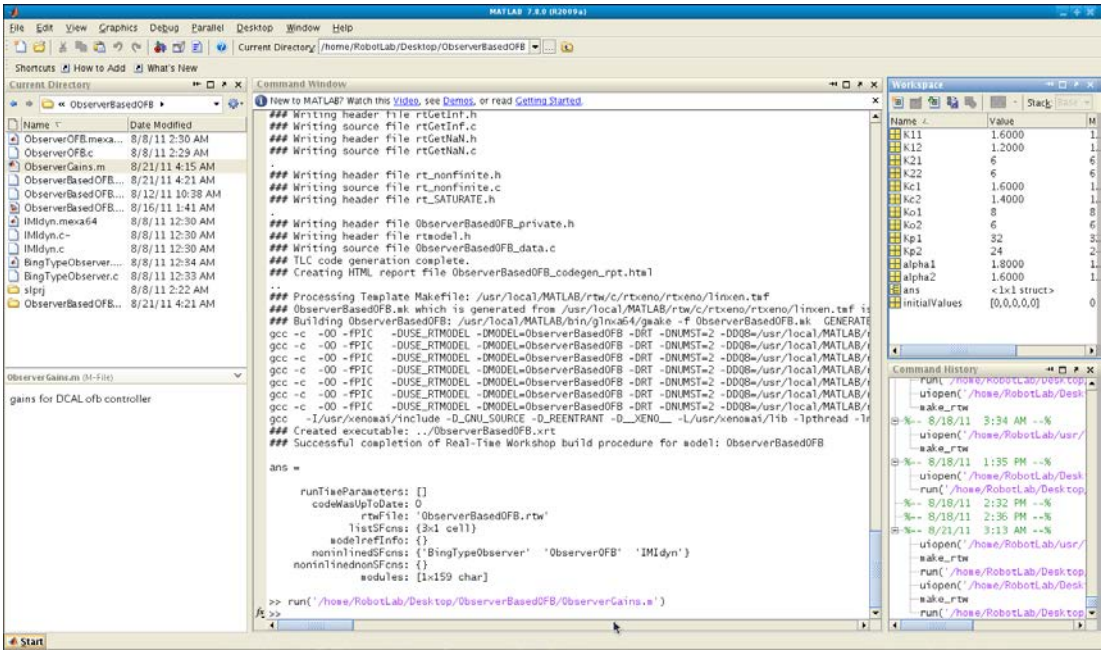
Model parametrelerine başlangıç değerlerinin atanması Matlab çalışma alanına Şekil 6.5'te listesi görülen parametre ve değerlerinin içinde bulunduğu `modelgains.m` dosyasının yüklenmesiyle gerçekleşir.



Şekil 6.5 Model Blok Parametrelerinin Başlangıç Değerleri

Model blok diyagramından C kodu üretip, Linux/Xenomai hedef ortamında gerçek zamanlı çalışabilir kod üretmek için Matlab komut satırında `make_rtw` komutu çağrıldığında C kodu üretim ve ardından derleme, link işlemleriyle `ObserverBasedOFB.xrt` isimli çalıştırılabilir kod üretilmiş olur. Şekil 6.6'da model kodunun derlenmesinden sonra Matlab ekran görüntüsü görülmektedir.

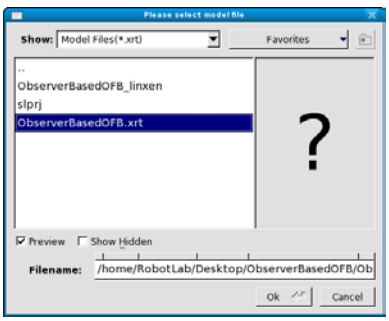
Bu aşamadan sonra `ObserverBasedOFB.xrt` gerçek zamanlı olarak Linux/Xenomai ortamında çalışabilen dosya RTXenoLab-GUI araclığıyla çalıştırılabilir. Bundan sonraki alt bölümde bu dosyanın RTXenoLab ile çalıştırılması anlatılacaktır.



Şekil 6.6: Model Kodunun Derlenmesi Sonrası Matlab Ekranı

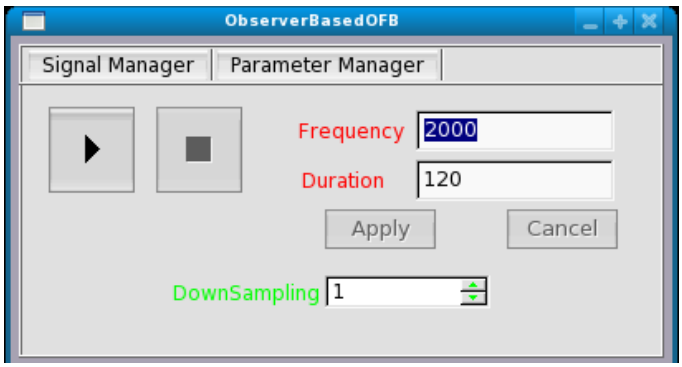
6.4 Model Kodunun RTXenoLab ile Çalıştırılması

Konsol ya da kısayol aracılığıyla RTXenoLab programı çalıştırıldıktan sonra Şekil 4.8'de görülen RTXenoLab kontrol paneli bilgisayar ekranında görülür. Kontrol panelinde "Experiments" menüsünden "New" seçeneği ile Şekil 4.9'da görülen dosya seçim penceresi oluşturulur. Bu dosya seçim penceresinden Simulink model diyagramının bulunduğu ve derlendiği dizinden Şekil 6.7'te görüldüğü üzere `ObserverBasedOFB.xrt` dosyası seçilerek çalıştırılır.



Şekil 6.7: Model Dosyasının Çalıştırılması

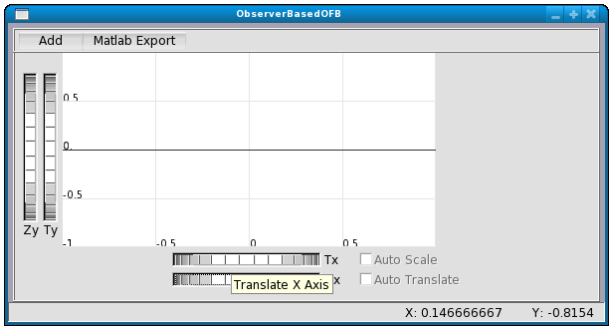
Model kodu çalıştırıldıktan sonra RTXenoLab kontrol panelindeki “Connect” seçeneği aktif hale gelir. Bu aşamada sunucu program, istemci programda yaratılmış olan ve sunucu ile istemci arasında veri alışverişini sağlayan Xenomai veri yapılarına bağlanmış durumdadır.



Şekil 6.8: ObserverBasedOFB Simulink-RTXLB Model Yönetim Penceresi

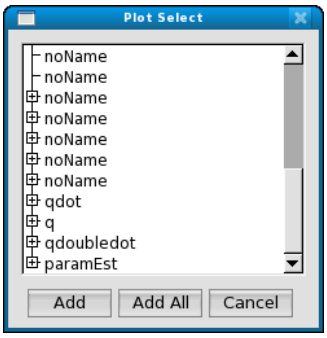
Kontrol panelindeki “Connect” seçeneğine basıldığında Şekil 6.8’de görülen model yönetim penceresi karşımıza çıkar. Burda “Frequency” ve “Duration” Şekil 6.2’de gösterilen, sırasıyla “Fixed step size (fundamental sample time)” ve “ Stop Time” konfigürasyon parametrelerine karşılık gelir. Bu parametreler gerçek zamanlı simülasyonun çalışma frekansı ve çalışma süresidir.

Model yönetim penceresinde görülen “Signal Manager” menüsü yardımıyla Şekil 6.9’da görülen “ ObserverBasedOFB Scope” penceresi karşımıza gelir.



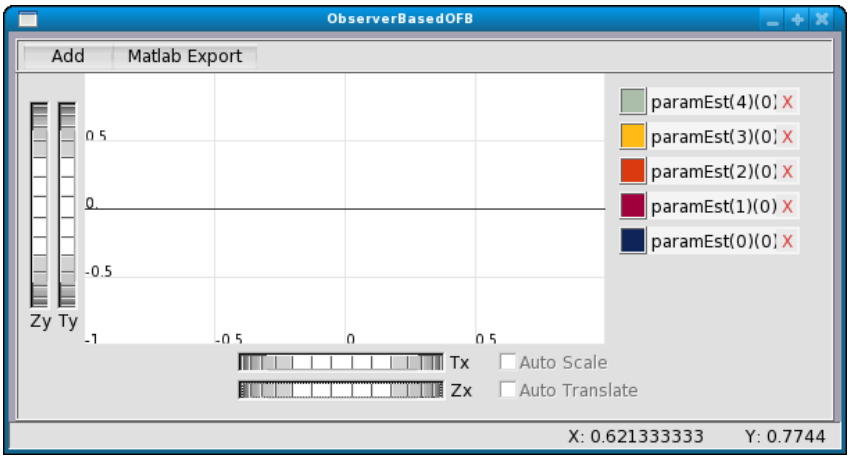
Şekil 6.9: ObserverBasedOFB Scope

Pencere üzerindeki “Add” butonuna basıldığında model sinyallerinin listelendiği şekil 6.10’da gösterilen pencere karşımıza çıkar.



Şekil 6.10: Model Sinyalleri Listesi

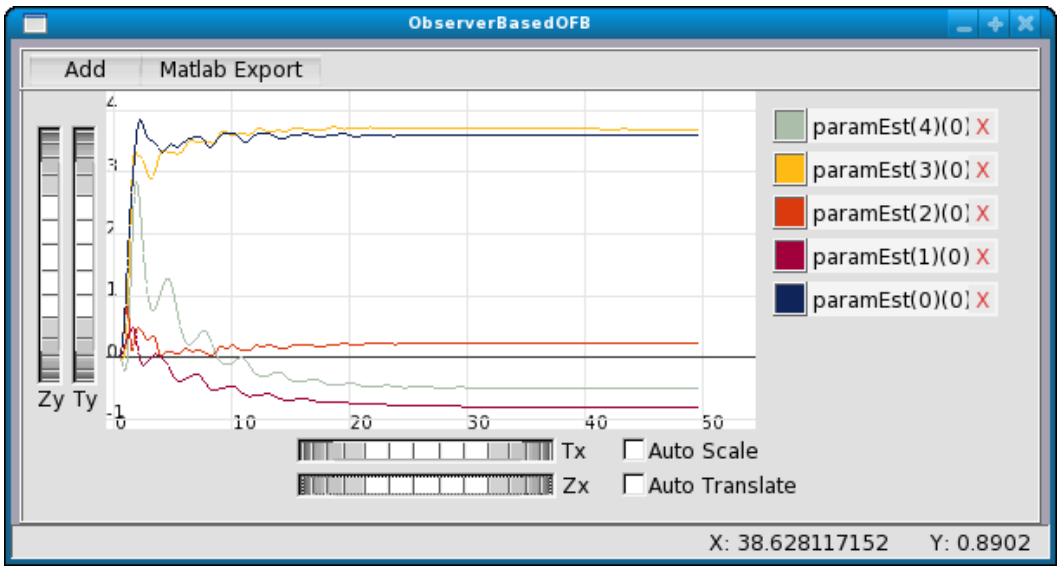
Model sinyalleri listesinde isim verilmeyen model sinyalleri “noName” olarak gösterilir ve bu sinyallere çalışma zamanında kayıt için kullanılan hafıza alanı tahsis edilmez.



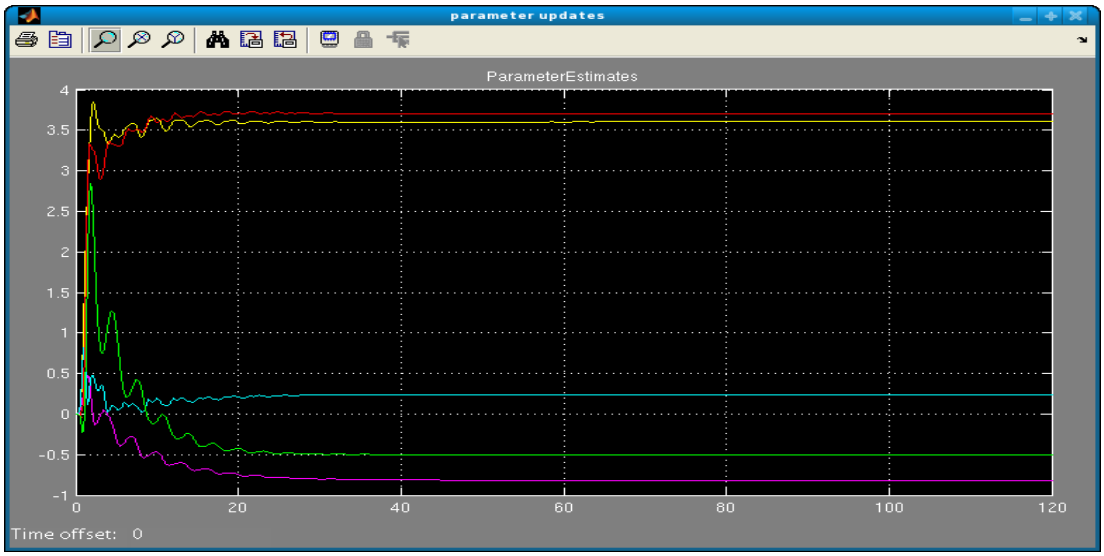
Şekil 6.11: paramEst Sinyalinin Scope Penceresine Eklenmesi

Listede, “paramEst” isimli sinyal seçilip liste penceresindeki “Add” butonuna basıldığında Şekil 6.11’de görüldüğü gibi ,sinyal Scope penceresine eklenir. Sinyal bir vektördür ve CAPI tarafından sütun matrisi olarak tanımlanmıştır. RTXenoLab bu nedenle sinyal üyelerini `<sinyal_ismi>(satır indeksi)(sütun indeksi)` formatında isimlendirir.

Model yönetim penceresindeki “START” butonuna basıldığında simülasyon çalışmaya başlar.

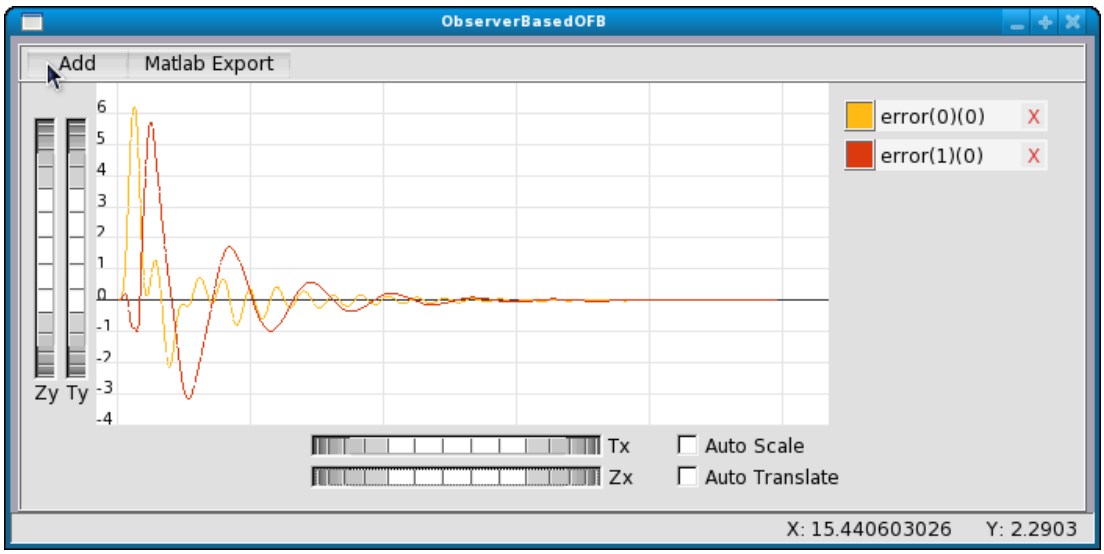


Şekil 6.12: paramEst Sinyalinin RTXLB Scope Penceresindeki Zamana Göre Değişim Grafiği

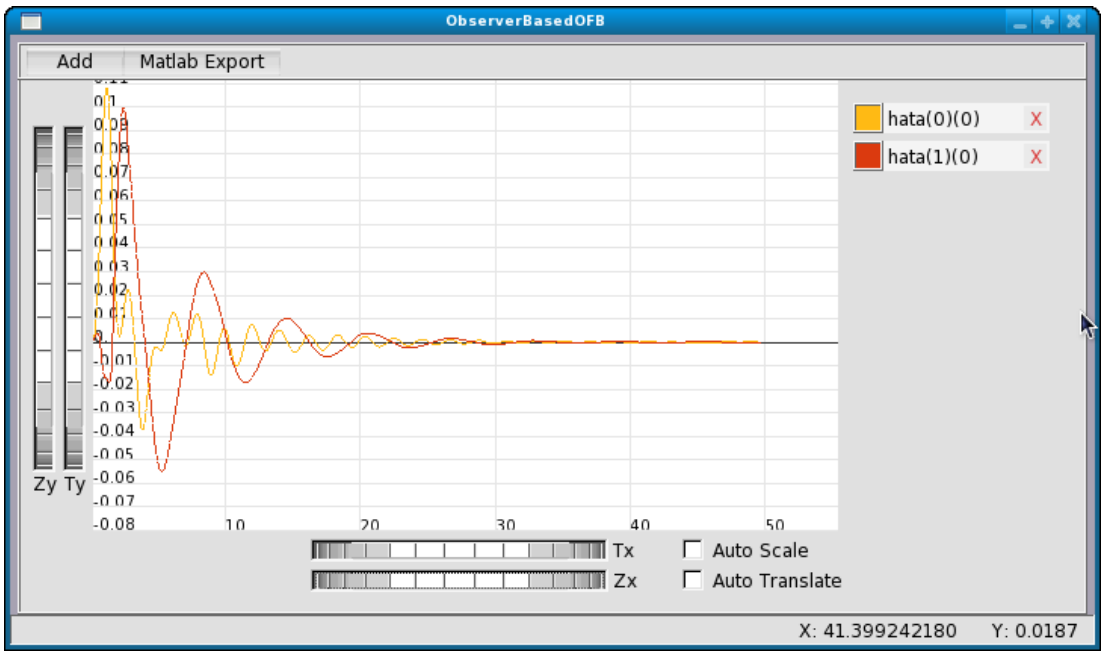


Şekil 6.13: paramEst Sinyalinin Simulink Scope Penceresindeki Zamana Göre Değişim Grafiği

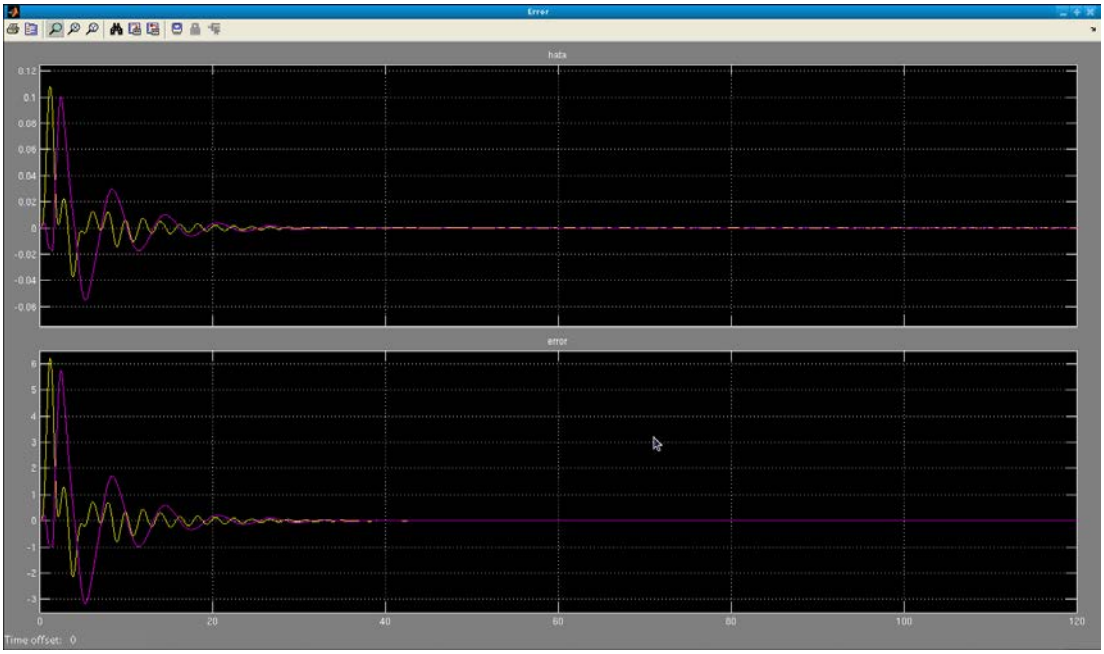
paramEst sinyalinin simülasyonun çalışma zamanı içindeki zamana göre değişim grafiklerinin RTXenoLab ve Simulink ortamları içindeki görünümü Şekil 6.12 ve Şekil 6.13'de görülmektedir. Şekil 6.14, 6.15 ve 6.16'da "error" ve "hata" model sinyalleri görülmektedir.



Şekil 6.14: "error" Sinyalinin RTXLB Scope Penceresindeki Grafiği



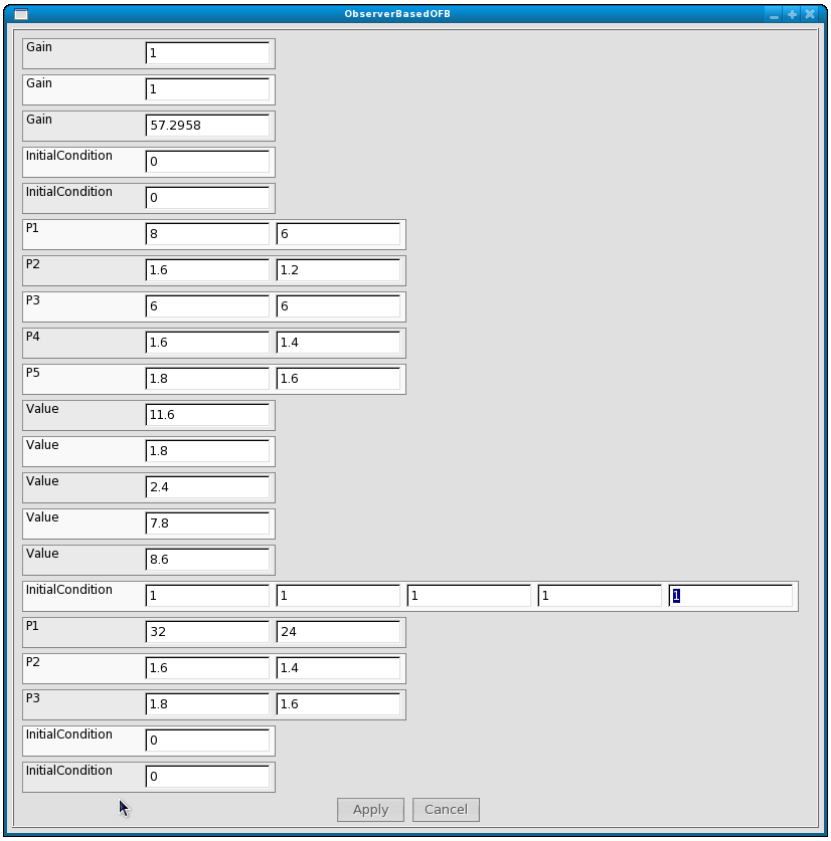
Şekil 6.15: "hata" Sinyalinin RTXLB Scope Penceresindeki Grafiği



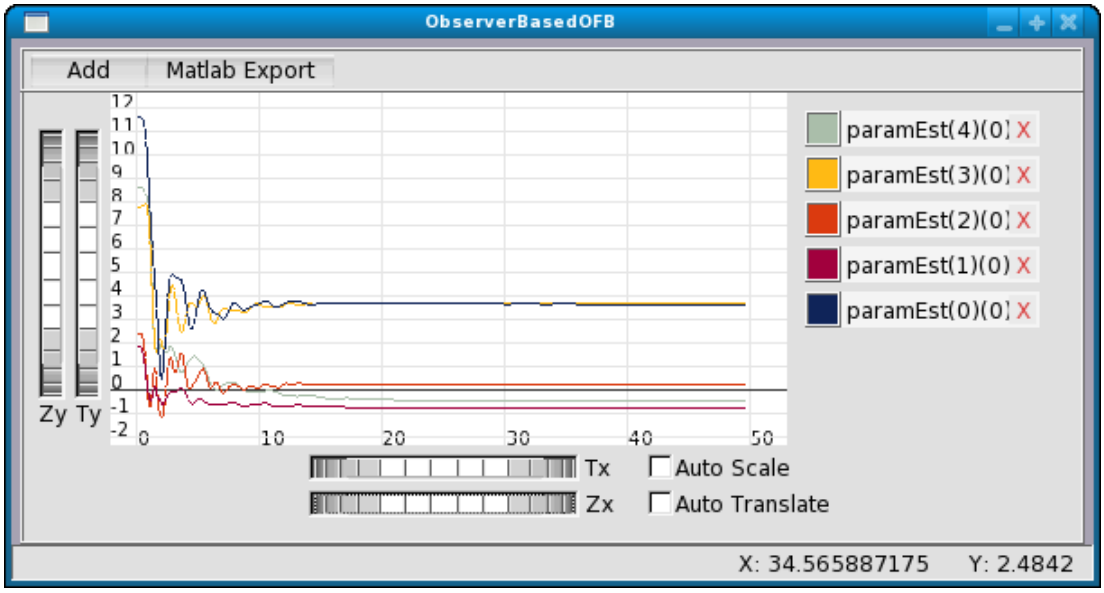
Şekil 6.16: “hata” ve “error” Sinyallerinin Simulink Scope Penceresindeki Grafikleri

Modelin blok parametrelerinin değişimlerinin sinyaller üzerindeki etkisi göstermek amacıyla “paramEst” başlangıç sinyalinin başlangıç değerleri RTXenoLab model yönetimi penceresindeki “Parameter Manager” menüsü üzerinden açılan “Parameter Properties” penceresi ile [0,0,0,0,0] değerinden [1,1,1,1,1] değerine ayarlanmıştır. Şekil 6.17’ de ObserverBasedOFB parametre penceresi görülmektedir.

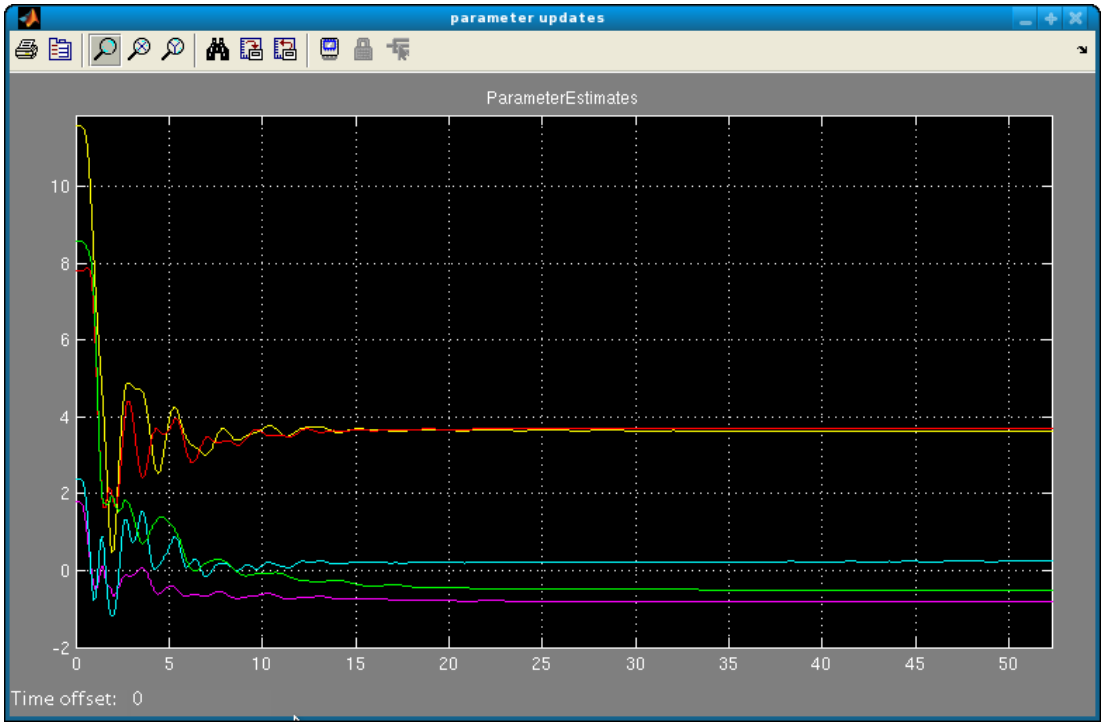
Yapılan bu parametre değişikliklerinin “paramEst” sinyali üzerinde yaptığı etki Şekil 6.18 ve Şekil 6.19’da gösterilmektedir. Bu şekiller Şekil 6.12 ve Şekil 6.13 ile karşılaştırılırsa parametre değişiminin sinyal üzerindeki etkileri gözlemlenebilir.



Şekil 6.17: RTXLB ObserverBasedOFB Modeli Parametre Penceresi



Şekil 6.18: Parametre Değişiminden Sonra RTXLB “paramEst” Sinyal Grafiği



Şekil 6.19: Parametre Değişiminden Sonra Simulink “paramEst” Sinyal Grafiği

7. SONUÇLAR

Bu tez kapsamında geliştirilen RTXenoLab, Simulink blok diyagramlarının gerçek zamanlı simülasyonunu yapan ve istendiğinde sahip olduğu Simulink donanım sürücüsü kütüphanesiyle donanım döngüsünü de simülasyona katan gerçek zamanlı bir simülasyon programıdır. RTXenoLab, şu anda sadece Quanser Q8 veri edinim ve kontrol donanımını desteklemekle beraber Linux/Xenomai tabanlı sürücüleri bulunan herhangi bir gömülü sisteminin ya da PC tabanlı bir donanımın Simulink/RTW sürücü blokları gerçeklenerek RTXenoLab Simulink kütüphanesine katılabilir[7].

RTXenoLab, yüksek çalışma frekanslarında gerçek zamanlı donanım döngülü simülasyon yapılmasına olanak sağlar. GYTE Kontrol ve Robotik Labaratuvarında (K.R.L) bulunan haptik sistemin Windows işletim sistemi tabanlı Simulink destekli donanım döngülü Simülasyon sistemi maksimum 1kHz'lik gerçek zamanlı çalışma frekansına sahiptir. Yine GYTE K.R.L içinde bulunan Windows/Simulink destekli donanım döngülü simülasyon sistemi dSpace yapılan deneylere göre 35kHz'lik gerçek zamanlı çalışma frekansını desteklemektedir. RTXenoLab ile 2,5GHz. Quad işlemcili bir bilgisayarda 180kHz'lik gerçek zamanlı çalışma frekanslarına çıkılmıştır.

RTXenoLab taşınabilir bir platformdur. Çapraz derleyiciyle derlendiğinde Linux/Xenomai sistemini barındıran herhangi bir donanımda çalışabilir. Çalıştırılabilir kod büyüklüğü yaklaşık 4 MB'tır. RTXenoLab, Simulink skaler model sinyallerinin her biri için hafızada 0.8 MB yer ayırır. Düşük hacimde hafıza kullandığından çoğu gömülü sistemde gerçek zamanlı olarak kullanılabilir.

RTXenoLab açık kaynak kodlu bir yazılımdır. Windows işletim sistemine ve Windows tabanlı, belli maliyetleri bulunan sürücü yazılımlarına gereksinimi ortadan kaldırır.

RTXenoLab geliştirilmeye açık bir platformdur. RTXenoLab'a gerçek zamanlı ağ haberleşmesini mümkün kılan Simulink blokları eklenerek farklı donanımlarda çalışan modellerin aynı anda bir arada koordineli çalışması sağlanırsa dağıtık sistemler oluştulabilir.

KAYNAKLAR

[1] C. Zhang,V. Vijapurapu,A. Srivastava,N. Schulz,J. Bastos,R. Wierckx,"Hardware-in-the-Loop Simulation of Distance Relay Using RTDS",SCSC:Proceedings of the 2007 Summer Computer Simulation Conference,July 2007.

[2] Zhiao Yao, Nicolae P. Costescu, Siddharth P.Nagarkatti , and Darren M. Dawson, "Real-Time Linux Target: A MATLAB-Based Graphical Control Environment", 2000.

[3] P. Regnier,G. Lima,L Barreto,"Evaluation of Interrupt Handling Timeliness in Real-Time Linux Operating Systems",SIGOPS Operating Systems Review,Volume 42 Issue6,October 2008.

[4] İ. Engin Sarıtaş, E. Zergeroğlu, "Kontrol / Kumanda Uygulama ve Geliştirmelerinde Kullanılmak Üzere Gerçek Zamanlı Bir Benzetim Platformu", Otomatik Kontrol Türk Mili Komitesi Ulusal Toplantısı, 2009.

[5] Y. Tian, G. Ren, Q. Wu, "Implementation of Real-time Network Extension on Embedded Linux", International Conference on Communication Software and Networks, 2009

[6] www.mathworks.com

[7] Egemen Kaleli, Erkan Zergeroğlu, "RTXeno-Lab:Gerçek Zamanlı Kontrol Uygulaması Geliştirme Yazılımı", Otomatik Kontrol Türk Mili Komitesi Ulusal Toplantısı, 2011.

ÖZGEÇMİŞ

Egemen Cumhur KALELİ İstanbul'da doğdu. Liseyi şimdiki adı olan Hasan Polatkan Anadolu Lisesi'nde okudu. Yıldız Teknik Üniversitesi Elektrik Mühendisliği bölümünden mezun olduktan sonra askerlik görevini tamamladı. Kısa bir süre özel bir şirkette çalıştı. Çalışma alanları gerçek zamanlı uygulama geliştirme ve robotiktir.