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

ZENOM BENZETİM ORTAMI İLE ARDUINO KONTROLÜ

HÜSNÜ KARAKÜCÜK
YÜKSEK LİSANS TEZİ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

GEBZE
2014

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

ZENOM BENZETİM ORTAMI İLE ARDUINO KONTROLÜ

HÜSNÜ KARAKÜCÜK
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

ARDUINO CONTROL WITH ZENOM SIMULATION ENVIRONMENT

HÜSNÜ KARAKÜCÜK

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 23.06.2014 tarih ve 2014 / 37 sayılı kararıyla oluşturulan jüri tarafından 18 / 09 / 2014 tarihinde tez savunma sınavı yapılan Hüsnü KARAKÜCÜK' ün tez çalışması Bilgisayar Mühendisliği Anabilim Dalında YÜKSEK LİSANS tezi olarak kabul edilmiştir.

JÜRİ

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

ÜYE : Doç. Dr. Yusuf Sinan AKGUL

ÜYE : Doç. Dr. İlyas KANDEMİR

ONAY

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

İMZA/MÜHÜR

ÖZET

Gerçek zamanlı benzetimler robot ve otomasyon sistemlerinin geliştirilmesinde ve test edilmesinde kullanılmaktadır. Gerçek zamanlı benzetimleri çalıştırmak için gerekli olan gerçek zamanlı sistemler, benzetim geliştirme maliyetlerini artırmaktadır. Bu durumun önüne geçebilmek için Xenomai yaması ile gerçek zamanlı özellikler kazanmış Linux işletim sistemi üzerinde çalışan Zenom benzetim ortamı geliştirilmiştir. Zenom benzetim ortamı, gelişmiş arayüzlere, 2 boyutlu grafik ekranlarına, benzetim sonuçlarını 3 boyutlu görselleyen sahne arayüzüne ve benzetim sonuçlarına müdahale etmeyi sağlayan kontrol ekranlarına sahiptir. Bu çalışma kapsamında, Zenom benzetim ortamının yetenekleri ve benzetim geliştirme işlemleri açıklanmıştır. Benzetim ortamının fiziksel sistemler ile veri alışveriş yeteneklerini göstermek için Arduino fiziksel platformu ile birlikte çalışmasını sağlayacak bileşenler gerçeklenmiştir. Örnek bir uygulama ile Zenom benzetim ortamı ile Arduino fiziksel platformunun birlikte çalışmasına yer verilmiştir.

Anahtar Kelimeler: Gerçek Zamanlı Benzetim, Xenomai, Zenom, Arduino.

SUMMARY

Real-time simulations are utilized in the development of robotic and automation systems. However, real-time systems which are required to operate real-time simulations increase development costs. In order to overcome this problem, a Zenom simulation environment which has acquired real time features by using Xenomai patch and which runs on Linux operation system has been developed. The Zenom simulation environment has advanced interfaces, 2-dimensional graphical screens, a scene interface that visualizes simulation results in 3-dimensions and control screens that enable to manipulate simulation results. In this project, capabilities of the simulation environment and simulation development processes expressed. To demonstrate Zenom's capabilities regarding data exchange with physical systems, components which will facilitate collaboration with the Arduino physical platform have been realized and the collaboration of Zenom application.

Key Words: Real-time simulation, Xenomai, Zenom, Arduino.

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,

Çalışmaları birlikte yaptığımız, gerektiğinde gece gündüz demeden çalışmanın tamamlanması için emek veren ve bütün çalışma boyunca yanımda olan, bilgi ve tecrübelerini benimle paylaşan değerli arkadaşım Cüneyt AY'a,

Ve göstermiş olduğu desteklerinden dolayı sevgili eşim Gökçe KARAKÜCÜK'e en içten teşekkürlerimi sunarım.

İÇİNDEKİLER

SİMGELER ve KISALTMALAR DİZİNİ

Simgeler ve Kısaltmalar		**Açıklamalar**
API	:	Application Programming Interface
GUI	:	Graphical User Interface
POSIX	:	Portable Operating System Interface for Unix
VRML	:	Virtual Reality Modeling Language

ŞEKİLLER DİZİNİ

1. GİRİŞ

Günümüzde bilgisayar destekli tasarım ve benzetim ürün ve sistemlerin geliştirilmesi ve test edilmesinde önemli bir yer tutmaktadır. Tasarım ve test sürelerinin kısaltılması ve test maliyetlerinin düşürülmesinde bilgisayar yazılımları ile yapılan benzetimlerin büyük etkisi vardır. Ayrıca bilgisayar ortamında yapılan benzetimler ile gerçek hayatta oluşturulamayacak koşullar yaratılarak normal koşullarda test edilemeyecek senaryolar test edilebilmektedir. Robotik gibi kontrol algoritmaları üzerinde çalışan sistemlerin, gerçek hayata geçirilmeden önce benzetim ortamları ile test edilmesi, gerçek sistemde oluşabilecek hataların önlenmesine büyük katkı sağlar [Khanna et al., 2013].

Kontrol algoritmalarının testini doğru şekilde yapabilmek için algoritmanın çalıştırılacağı sistemin frekansında hesaplama yapmak gereklidir. Ayrıca benzetim gerektiğinde sistem ile haberleşebilmeli ve sistemden gelen verileri hesaplamalarda kullanabilmelidir. Bu benzetimleri çalıştırabilmek için gerçek zamanlı çalışabilecek ve donanım ya da sistem ile iletişim kurabilecek benzetim ortamına ihtiyaç duyulur.

Gerçek zamanlı benzetimlerin çalıştırılması için gerçek zamanlı sistemler kullanılır. Bu sistemler, karmaşık kullanıcı etkileşimine izin vermeyen, sadece sistem için özel geliştirilmiş uygulamaları çalıştırabilen, bazen özel donanımlara ihtiyaç duyan sistemlerdir. QNX [Web 1, 2014] ve VxWorks [Behnam et al., 2008] gibi gerçek zamanlı işletim sistemleri bulunmaktadır. Ayrıca, günümüzde yaygın olarak kullanılan açık kaynak kodlu Linux işletim sistemi, üzerine kurulan yamalar ile gerçek zamanlı uygulama çalıştırabilir hale getirilebilmektedir. Bu sayede standart bir bilgisayar gerçek zamanlı uygulamaların çalıştırılabileceği bir sisteme dönüştürülebilir.

Linux işletim sistemini gerçek zamanlı uygulama çalıştırabilir hale getirmek uygulanabilecek birden fazla yama vardır. Bunlardan yaygın olarak kullanılanların Xenomai [Kohen, 1999] ve RTAI yamasıdır [Mantegazza, 2007]. Xenomai dünyaca kabul görmüş Posix API arayüzü 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. Xenomai kolay kurulumu ve

standart uygulamaların hızlı şekilde Xenomai ile uyumlu hale getirilebilmesi ile diğer yamalardan ayrılır [Barbalace et al., 2007].

Linux işletim sistemine uygulanan yamalar ile gerçek zamanlı uygulamaların çalıştırmasını sağlayarak, gerçek zamanlı benzetim koşturmak, ek gerçek zamanlı sistem ve bu sistem için özelleşmiş uygulama geliştirmek için gereken maliyetleri büyük ölçüde azaltacaktır.

Benzetimin koşturulacağı sistemin daha karmaşık işlemlere izin vermesi, görsel arayüzleri daha gelişmiş, daha fazla fonksiyona sahip, geliştirilmesi için PC uygulama geliştirme bilgisinin yeterli olacağı benzetim ortamlarının geliştirilmesinin önünü açmaktadır.

Biz de Linux işletim sisteminin bu özelliğinden faydalanarak, gerçek zamanlı benzetim çalıştırabilen bir benzetim ortamı geliştirdik. Geliştirdiğimiz benzetim ortamı Xenomai yamalı Linux üzerinde çalışan, açık kaynak kodlu, gelişmiş arayüz yeteneklerine sahip bir benzetim ortamıdır. Geliştirdiğimiz benzetim ortamını Zenom olarak adlandırıldı ve tasarımı yeni özelliklerin kolayca geliştirilebileceği bir yapıda olmasına önem verilerek yapıldı.

Zenom benzetim ortamı geliştirilmeden önce mevcut benzetim ortamları incelenerek, kullanımı kolay ve temel programlama bilgisi ile benzetim geliştirilebilecek bir benzetim ortamı geliştirilmesine önem verildi. Mevcut benzetim ortamlarından QMotor [Web 2, 2001], RT-LAB [Web 3, 2014], RTAI-Lab [Web 4, 2014], VisSim [Web 5, 2014] ve dSPACE [Web 6, 2014] ürünleri incelenmiştir. İncelenen benzetim ortamları geliştirdiğimiz benzetim ortamı için referans olarak kullanılmıştır.

"QMotor", yalnızca QNX işletim sistemi üzerinde çalışmaktadır ve C++ benzetim geliştirilmesine imkân vermektedir. QNX işletim sisteminin yapısından dolayı gerçek zamanlı çalışmaktadır ve ücretsizdir. 2002 yılında geliştirilmesi tamamlanmıştır [Loffler et al., 2002] ve bu nedenle güncel benzetim ortamlardan görsel olarak geridedir. 2 boyutlu grafik arayüzüne sahiptir.

dSPACE benzetim otomotiv ve uzay araştırmalarında kullanılan gelişmiş bir benzetim ortamıdır [Gumeniuk et al., 2012]. Gelişmiş arayüzlere, 2 boyutlu ve 3 boyutlu görselleme ekranlarına sahiptir. Gerçek zamanlı çalışması için dSPACE firması tarafından sağlanan donanımlara ihtiyaç duyar. MATLAB [Web 7, 2014] üzerinde geliştirilen benzetimlerin dSPACE üzerinde çalışmasını sağlayan bileşenler

içerir. Benzetim geliştirmek için Simulink [Web 8, 2014] bilgisine ihtiyaç vardır. Ücretli bir uygulamadır.

VisSim sadece Windows [Web 9, 2014] işletim sistemi üzerinde çalışan bir benzetim ortamıdır. Ücretlidir ve gerçek zamanlı çalışma desteği yoktur. Benzetimler model tabanlı tasarım arayüzü üzerinde geliştirilir. 3 boyutlu görselleme yeteneğine sahiptir. Benzetim geliştirmek için benzetim ortamının model tabanlı tasarımı hakkında bilgi sahibi olmak gereklidir.

RT-LAB benzetim ortamı OPAL-RT [Web 10, 2014] firması tarafından geliştirilmiş, ücretli bir benzetim ortamıdır. Gelişmiş arayüzlere sahiptir ve büyük ölçekli benzetimlerin geliştirilmesinde kullanılabilir [Dongping et al., 2007]. Simulink üzerinde geliştirilen benzetimlerin çalıştırılmasını sağlayan bileşenler içerir. C++ dilinde benzetim geliştirmek için gereken bileşenler ek bir paket olarak satılmaktadır. 3 boyutlu görselleme yeteneği için gerekli olan bileşenler de ek paket olarak satılmaktadır. Benzetim ortamının çalışması için OPAL-RT firması tarafından sağlanan donanımlara ihtiyaç vardır.

RTAI-Lab, Linux işletim sistemi için gerçek zamanlı çalışma yeteneği sunan RTAI yamasını geliştiren ekip tarafından geliştirilmiş bir benzetim ortamıdır [Dufour et al., 2008]. Model tabanlı tasarım arayüzü ile benzetim geliştirme yapılmaktadır. 3 boyutlu görselleme yeteneği yoktur, sadece 2 boyutlu görselleme yapabilmektedir. Ücretsiz bir uygulamadır ve RTAI yamalı Linux işletim sistemi üzerinde gerçek zamanlı olarak çalışmaktadır. Simulink ile geliştirilen benzetimlerin RTAI-Lab ile çalıştırılmasını sağlayan bileşenler içermektedir.

Referans alınan benzetim ortamlarının sunduğu imkânlar ve kullanım şekilleri değerlendirerek tasarlanan ve geliştirilen Zenom benzetim ortamının kolay geliştirilebilir yapısını kullanarak, günümüzde oldukça yaygın olarak kullanılan Arduino fiziksel programlama platformunu Zenom ile birlikte kullanabilme imkânı sağlayan eklentiler geliştirildi.

1.1. Tezin Amacı, Katkısı ve İçeriği

Gerçek zamanlı benzetimler günümüzde birçok alanda yaygın olarak kullanılmaktadır. Robotik, otomotiv endüstrisi gibi alanlarda gerçek zamanlı benzetimler gerçek hayatta oluşturulamayan ya da oluşturulma maliyeti çok yüksek

olan senaryolarının test edilmesine imkân sağlar. Gerçek zamanlı benzetimlerin ihtiyaç duyduğu gereksinimler nedeniyle özel sistemler kullanılması gerekmektedir.

Bu tez kapsamında geliştirilen Zenom benzetim ortamı sayesinde herhangi bir ek sistem maliyeti olmadan gerçek zamanlı benzetim çalıştırabilmektedir. Ayrıca benzetim geliştirmek için kullanılan dil ve araçlar, standart bilgisayar yazılımlarını geliştirmek için kullanılan dil ve araçlar ile aynıdır. Bu sayede temel programlama bilgisi ile benzetim gerçeklenebilir. Zenom benzetim ortamı ve Zenom üzerinde çalışacak benzetimin geliştirilmesi için C++ dili tercih edilmiştir. Zenom benzetim ortamı yaygın olarak kullanılan C++ programlama dilini tercih ederek geniş kitleler tarafından kullanılmayı amaçlamıştır [Web 11, 2014].

Gerçek zamanlı benzetimler genellikle bir donanım ya da sistem ile birlikte çalıştırılır. Bu sayede sistem ya da donanım test edilir [Sarker et al., 2006]. Zenom benzetim ortamının donanım ile birlikte çalışma yeteneklerini göstermek için Arduino fiziksel programlama platformu ile Zenom arasında haberleşmeyi sağlayan geliştirmeler gerçeklenmiştir. Gerçeklenen bileşenler "Zenom-Arduino" olarak adlandırılmıştır. "Zenom-Arduino" bileşenleri ile Zenom üzerine yeni eklentiler geliştirmenin kolaylığı gösterilmiştir. Ayrıca geliştirilen örneklerde, Arduino hesaplama gücünün yetersiz kaldığı durumlarda hesaplama işlemlerinin Zenom ile yapılmasını sağlanmıştır. Bu sayede sistemin ya da donanımın yetersiz kaldığı durumlarda Zenom ile bilgisayar kaynaklarının sistem ve donanıma ek hesaplama gücü katabileceği gösterilmiştir.

2. ZENOM

Zenom, gerçek zamanlı benzetim ve kontrol algoritmaların C++ dili ile geliştirilmesini sağlayan bir ortamdır. Zenom, benzetim sonuçlarının görsellenmesi ve koşum esnasında değişiklikler yapılmasına imkân veren "ZenomGUI" olarak adlandırılmış grafiksel bir arayüze sahiptir. "ZenomGUI", 2 boyutlu grafik gösterimi, 3 boyutlu sahne ve veri aktarma yeteneklerine de sahiptir. Zenom, çekirdeği Xenomai ile yamalanmış Linux işletim sistemi üzerinde hiçbir donanım kısıtı olmadan gerçek zamanlı olarak çalışabilmektedir.

Xenomai, Linux çekirdeğini gerçek zamanlı çalışabilir hale getiren, RTOS temeli üzerine kurulmuş, sıkı gerçek zaman ve esnek gerçek zamanlı servisleri destekleyen bir yamadır [Web 12, 2013]. Çalışma prensibi, öncelik verilmiş çekirdek yapısı oluşturur ve kendi çekirdeğini Linux çekirdeğinden daha öncelikli hale getirir. Bu sayede donanım ile ilk haberleşen yapı olur. Donanımdan gelen mesajları ve kesmeleri ilk olarak Xenomai işler. Gelen mesajlar ve kesmeler Xenomai ile ilgili ile işleme alır, eğer kendisi ile ilgili değilse Linux çekirdeğine yönlendirir. Bu sayede Linux çekirdeğinden bağımsız, donanım üzerinde daha öncelikli işlemler yapma hakkına sahip bir katman oluşur. Xenomai bu katmanı kullandırmak için "skin" olarak adlandırılan iki farklı kullanıcı programlara arayüzü sunar. Bu arayüzlerden ilki, Xenomai ile benzer işlevlere sahip VxWorks [Web 13, 2014] ve pSOS [Web 14, 2014] gibi yamaların arayüzlerine benzeyen ve sıkı gerçek zamanlı çalışan servisler sunan, "Native Skin" olarak adlandırılan arayüzdür. Diğer arayüz ise Linux programlamada standart olan POSIX arayüzü ile aynı olan esnek gerçek zamanlı çalışan servisler sunan, POSIX Skin olarak adlandırılan arayüzdür [Web 15, 2013]. Bu arayüzler sayesinde gerçek zamanlı uygulamaların Xenomai üzerinde çalıştırılması ve POSIX için yazılan uygulamaların esnek gerçek zamanlı hale getirilmesi oldukça kolaydır.

Zenom çatısı "ControlBase" ve "ZenomGUI" olmak üzere iki ana bileşen oluşur. ControlBase, Xenomai "Native Skin" kullanarak geliştirilmiş, benzetim ve kontrol algoritmalarının kullandığı beş temel işlevi çalıştıran sıkı gerçek zamanlı bir uygulamadır. Kullanıcı "ControlBase" içinde tanımlı beş temel işleve karşılık metotları gerçekler ve uygulama çalıştırıldığında bu metotları koşturarak sonuçları üretir. "ZenomGUI", esnek gerçek zamanlı Xenomai programlama arayüzünü

kullanarak “ControlBase” ile haberleşir ve üretilen sonuçları gelişmiş grafiksel arayüzleri ile görseller. Ayrıca “ZenomGUI”, “ControlBase” çalışırken benzetim parametrelerini değiştirme, sonuçlarda değişiklik yapma imkânı da sağlar.

Zenom benzetim ortamının temelinde iki farklı tip değişken vardır. Bu değişkenlerden ilki, çıkış değişkeni olarak ifade edilebilecek olan değişkenlerdir ve “LogVariable” olarak adlandırılır. “LogVariable” değişkenleri benzetim koşumu süresince güncel değerleri kullanıcıya aktarılan değişkenlerdir. Farklı bir ifade ile benzetimin ürettiği sonuç değerleridir. Diğer değişken tipi ise “ControlVariable” olarak adlandırır ve kullanıcının benzetime müdahale etmesine imkân verir. Koşum süresince kullanıcı bu değişkenlerin değerlerini değiştirerek koşum sonuçlarında değişiklikler sağlayabilir. Kullanıcı “ControlBase” gerçeklerken “LogVariable” ve “ControlVariable” tiplerini Zenom içindeki kayıt metotları ile benzetime kaydeder. “ControlBase” ve “ZenomGUI” uygulamaları “LogVariable” ve “ControlVariable” değişkenlerini kullanarak koşum sonuçlarını üretir ve görseller.

Zenom benzetim ortamı “Zenom-Fltk” ve “Zenom-Qt” olmak üzere iki farklı sürümü bulunmaktadır. Zenom-Fltk daha önce geliştirilmiştir ve hız odaklı Fltk [Web 16, 2012] ve OpenVrml [Web 17, 2013] kütüphaneler ile gerçeklenmiştir. “Zenom-Qt” sürümü ise, Qt ve OpenSceneGraph [Wang et al., 2010] kütüphanelerini ile gerçeklenmiştir. Qt çoklu platform desteği olan grafiksel kullanıcı arayüzü geliştirme çatısıdır. C++ programlama dili kullanarak Windows, Linux, Mac OS işletim sistemlerinde çalıştırılabilen uygulamaların geliştirilmesinde kullanılır [Web 18, 2014]. OpenSceneGraph kütüphanesi açık kaynak kodlu, yüksek performanslı 3 boyutlu grafiksel arayüz geliştirme kütüphanesidir [Web 19, 2013]. OpenSceneGraph benzetim ve modelleme projelerinde 3 boyutlu grafiksel arayüz geliştirmelerinde kullanılmaktadır [Yuan et al., 2007]. “Zenom-Qt” sürümünde, “Zenom-Fltk” sürümünde karşılaşılan hataların giderilmiş, arayüzler daha modern ve sorunsuz hale getirilmiştir. Ayrıca, “Zenom-Qt” sürümünde projenin tasarımı sadeleştirilmiş ve açık kaynak kodun anlaşılabilirliği artırılmıştır.

Zenom benzetim ortamında geliştirme işlemleri hızlandırmak için algoritma geliştirmek için gerekli olacak temel matematik işlemlerini içeren “ZenomMath” kütüphanesi bulunmaktadır. Bu kütüphane ile türev ve integral alma gibi işlemler metot çağrımları ile yapılabilmektedir.

2.1. Zenom Bileşenleri

Zenom benzetim ortamı “ControlBase” ve “ZenomGUI” olmak üzere iki temel uygulamadan oluşur. Ayrıca proje oluşturmaya amacıyla kullanılan “Zenom Project” uygulaması yeni proje oluşturulmasını sağlar. Zenom benzetim ortamının kullanıcıya görsellenecek ve kullanıcıdan alınacak değerler için ZenomVariable adında değişken tanımlaması bulunur. “ControlBase” ve “ZenomGUI” bu değişkenler üzerinde çalışmaktadır.

2.1.1. ZenomVariable

Zenom benzetim ortamında kullanıcı tarafından giriş ve çıkış değişkenleri olarak kullanılan değişkenlere genel olarak “ZenomVariable” adı verilmiştir. Çıkış değişkeni olarak kullanılan değişkenlere “LogVariable”, giriş değişkenlerine ise “ControlVariable” adı verilmiştir. Kullanıcı “ControlBase” gerçeklerken gerekli metot çağrımlarını yaparak “ZenomVariable” değişkenlerini tanımlar. Tanımlanan bu değişkenler bir yapılandırma dosyasına kayıt edilir. “ControlBase” ve “ZenomGUI” kaydedilen yapılandırma dosyasını okuyarak iki uygulamanın ortak ZenomVariable değişkenleri üzerinde çalışması sağlanır.

2.1.1.1. LogVariable

Zenom benzetim ortamında kullanıcı tarafından çıkış değişkenleri olarak tanımlanan değişkenlerdir. Kullanıcı “ControlBase” gerçeklerken “void registerLogVariable(double*, int, string)” imzalı metodu çağırarak değişkenini “LogVariable” olarak kaydeder. “LogVariable” değişkenleri sınırsız sayıda olabilir. Benzetim koşumu esnasında kullanıcının belirlediği frekansa göre değerleri güncellenir ve kullanıcıya görsellenir.

2.1.1.2. ControlVariable

Zenom benzetim ortamında kullanıcı tarafından giriş değişkenleri olarak tanımlanan değişkenlerdir. Kullanıcı “ControlBase” gerçeklerken “void

registerControlVariable(double*, int, string)" imzalı metodu çağırarak değişkenini "ControlVariable" olarak kaydeder. "ControlVariable" değişkenleri sınırsız sayıda olabilir. Benzetim koşumu esnasında "ZenomGUI" kullanılarak "ControlVariable" değerleri güncellenebilir "ControlVariable" değeri güncellendiği anda "ControlBase" uygulamasına güncel değer iletilir ve koşum sonuçlarına müdahale edilebilir.

2.1.2. Zenom Project

Zenom Project uygulaması, Zenom üzerinde çalıştırılacak benzetim geliştirileceği projeyi oluşturan uygulamadır. Uygulama proje ismi parametresi ile birlikte çalıştırılır ve çıktı olarak girilen uygulama isminde bir klasör oluşturur. Oluşan klasör içinde kullanıcının gerçeklemesi gereken beş metodun bulunduğu ".cpp" uzantılı bir dosya ve projeyi derlemek için kullanacak "makefile" dosyası bulunur.

2.1.3. ControlBase

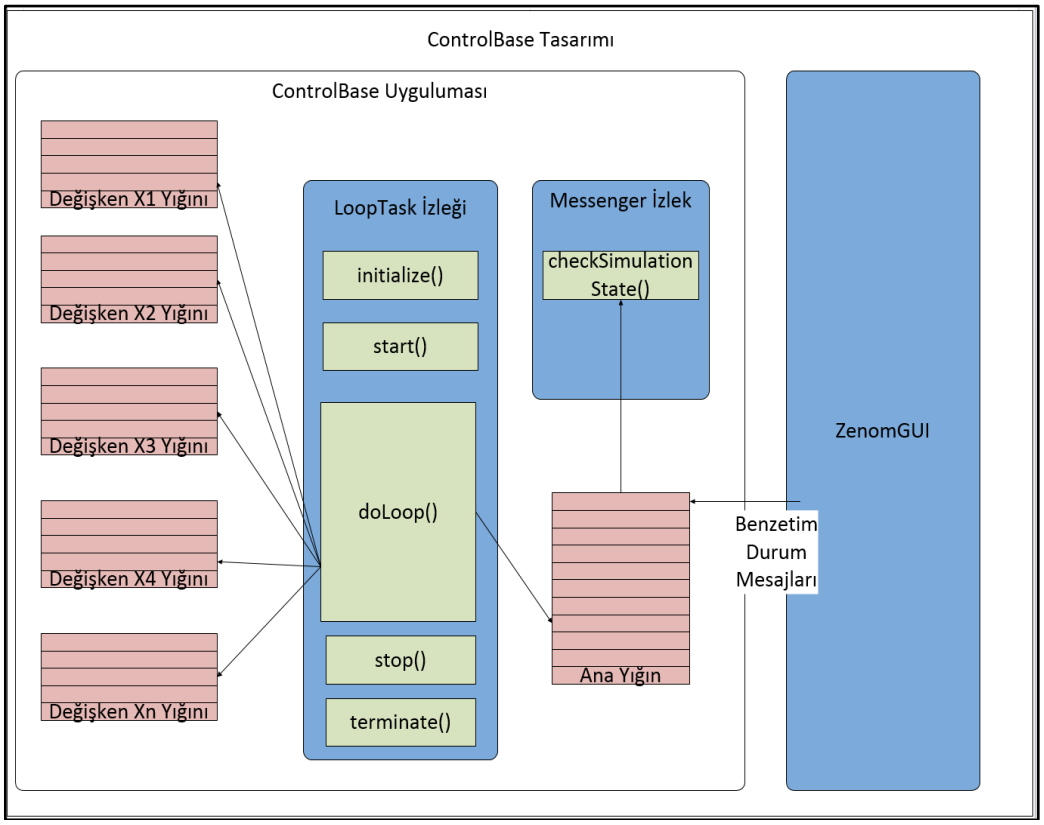
"ControlBase" uygulaması Zenom benzetim ortamının benzetimi çalıştıran uygulamasıdır. Xenomai "Native Skin" ile geliştirilmiş ve sıkı gerçek zamanlı çalışan bir uygulamadır. Zenom için benzetim geliştirirken "ControlBase" sınıfından türetilmiş bir sınıf içindeki beş metot gerçeklenir ve derlenir. Derleme sonucunda "ControlBase" uygulaması oluşur.

"ControlBase" uygulamasının tasarımı "Zenom-Fltk" ve "Zenom-Qt" sürümlerinde farklılaşmıştır. Ancak kullanıcının gerçeklediği kısımda herhangi bir değişiklik yapılmamıştır. Bu sayede kullanıcı "Zenom-Fltk" için geliştirdiği projeyi "Zenom-Qt" ile derleyerek yeni sürüm ile kullanabilir.

"ControlBase" uygulamasının yeni sürümünde tasarım sadeleştirilmiştir. "Zenom-Fltk" sürümünde üç izlek kullanan "ControlBase" yeni sürümde iki izlek kullanarak çalışmaktadır. Uygulamada çalışan izleklerden birinci benzetim koşumunu sağlayan, kullanıcının gerçeklediği metotları çağıran izlektir. Diğer izlek ise "ControlBase" ile "ZenomGUI" arasındaki iletişimi sağlayan izlektir.

"ControlBase" uygulaması benzetim süresince kullanılacak hafıza elemanlarının yönetiminden sorumludur. Kullanıcının tanımladığı "LogVariable"

sayısı, “ControlVariable” sayısı, koşum süresi, koşum frekansı ve kayıt frekansına göre gereken hafıza elemanlarının boyutları ile sayısını belirler ve rezerve eder. Koşum süresince iki temel hafıza kullanır. Bunlardan “Main Heap” olarak adlandırılan hafıza, koşum esnasında zaman, “ControlVariable” ve “LogVariable” değişkenlerinin en güncel halini tutan hafızadır. Her değerin en güncel hali bu hafızada tutulur. Ayrıca “ZenomGUI” ile yapılan haberleşmedeki mesajlarda bu hafızaya yazılır. Boyutu koşum başlamadan önce belirlenir ve koşum boyunca değişmez. Diğer hafıza elemanı ise “Variable Heap” olarak adlandırılır ve her “LogVariable” değişkeni için bir tane oluşturulur. Koşum süresince kayıt frekansına göre “LogVariable” güncel değeri zaman etiketi ile birlikte her “LogVariable” için oluşturulan “Variable Heap” içine yazılır. Bu hafıza tipinin boyutu koşum süresince artmaktadır. Her yeni değer hafızaya eklenir. Zamana bağlı değer değişimi görsellemek için bu hafıza kullanılır. Zenom benzetim ortamı bileşeni “ControlBase” uygulamasının tasarımı Şekil 2.1 ile gösterilmiştir.



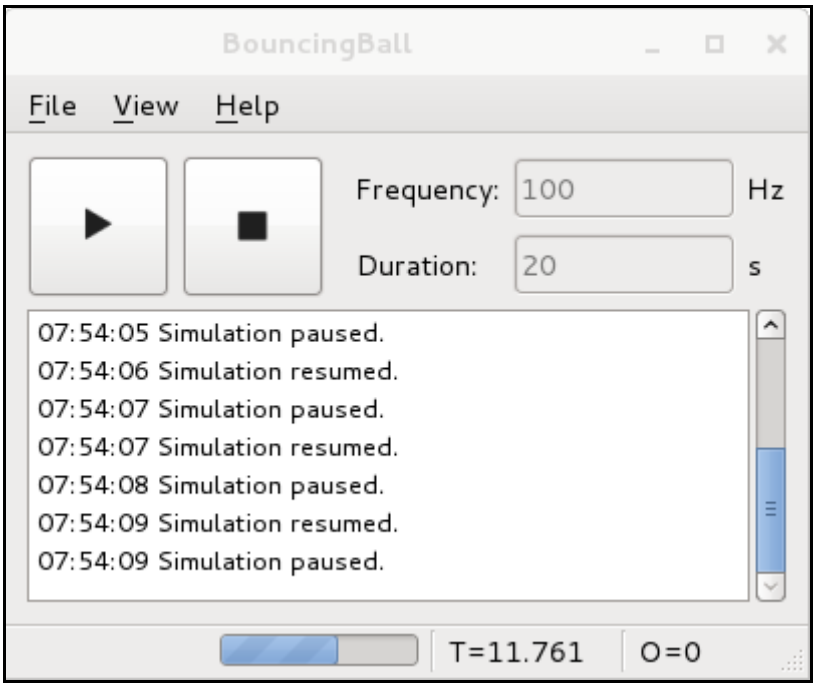
Şekil 2.1: Zenom “ControlBase” tasarım şeması.

2.1.4. ZenomGUI

"ZenomGUI" , "ControlBase" uygulamasının yüklenmesi/kapatılması ve benzetim koşumunun kontrol edilmesi, "ControlBase" tarafından üretilen sonuçların görsellenmesi ve koşum esnasında "ControlBase" parametrelerinin değiştirilmesi gibi işlevlere sahip olan kullanıcı arayüzü uygulamasıdır. "ZenomGUI", "ControlBase" sonuçlarını anlık olarak göstermekle birlikte, 2 boyutlu grafikler ile zamana göre değişimin takip edilmesi ve 3 boyutlu sahne ile sonuçların model üzerinde gösterilmesi gibi yeteneklere de sahiptir.

2.1.4.1. Main Window Penceresi

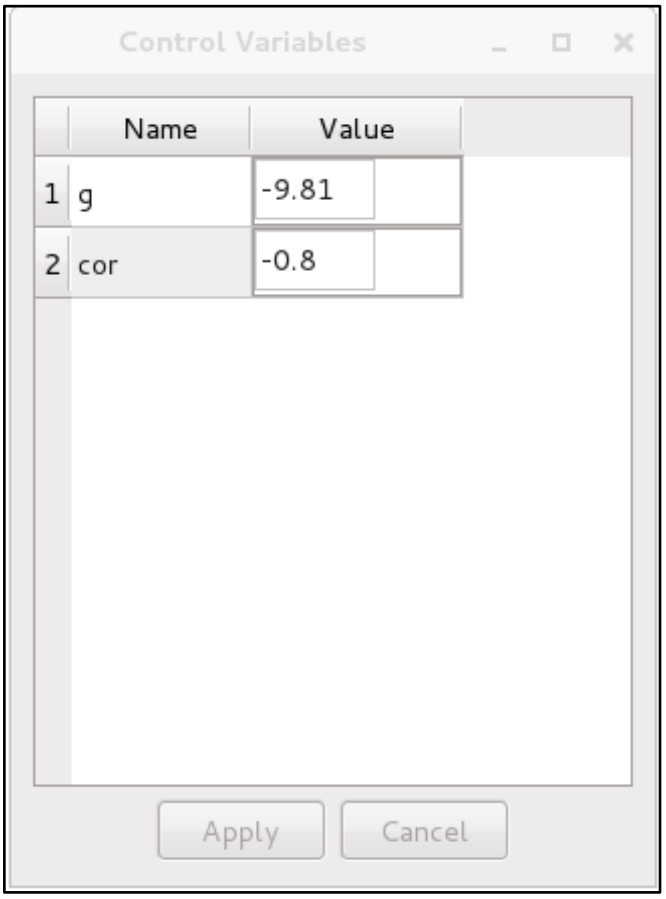
"ZenomGUI" uygulamasında "ControlBase" projesini yüklemek ve koşumu kontrol etmek için kullanılan arayüzdür. Koşum süresi ve frekansı bu arayüz ile belirlenir. Koşum başlatma, durdurma ve duraklatma gibi işlevlere sahiptir. "MainWindow" içindeki mesaj ekranında, koşum süresince oluşan olaylar listelenir. "MainWindow" penceresinin ekran görüntüsü Şekil 2.2 ile gösterilmiştir.



Şekil 2.2: "ZenomGUI" – "MainWindow" penceresi görüntüsü.

2.1.4.2. Control Variables Penceresi

Kullanıcı benzetimini gerçeklerken kaydettiği “ControlVariable” tipindeki değişkenler bu pencerede gösterilir. Bu pencerede her “ControlVariable” değişkeni için bir satır bulunur ve “Value” başlığı altındaki alandan değişken değeri değiştirilebilir. Bu ekranda yapılan değişiklikler “ControlBase” uygulamasına iletilir. “ControlVariable” tipindeki değişkenlerin sadece anlık değerleri sistemde tutulmaktadır. Bu pencerede değişkenlerin en güncel değerleri görüntülenir. “Control Variables” penceresinin ekran görüntüsü Şekil 2.3 ile gösterilmiştir.

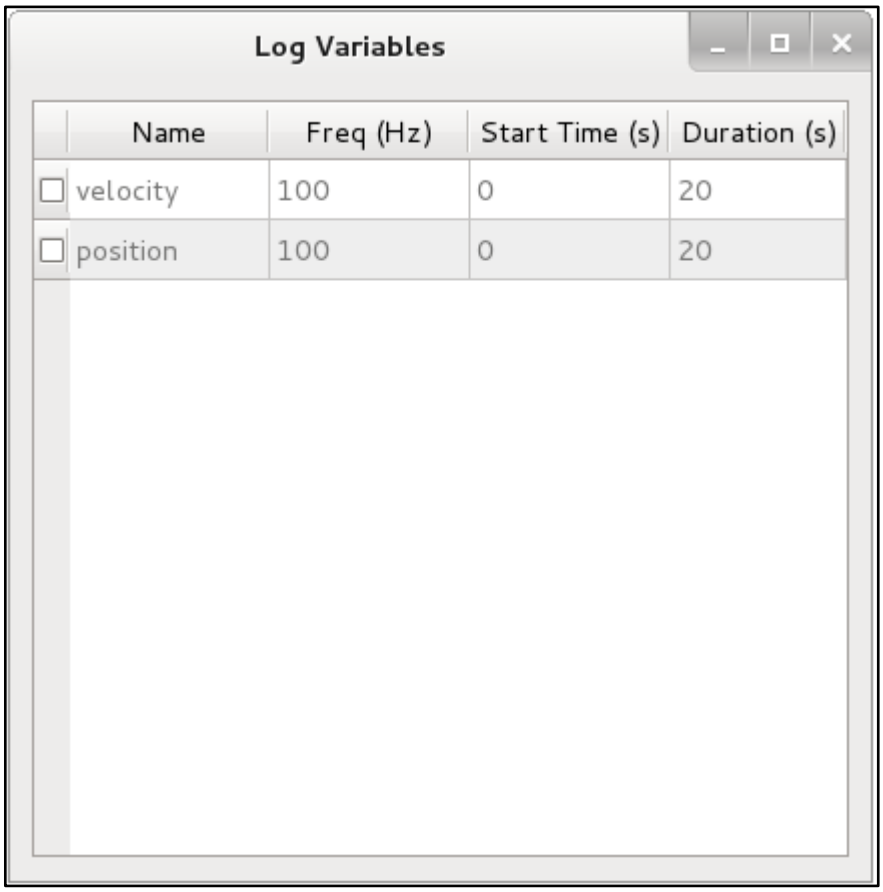


Şekil 2.3: “ZenomGUI” – “Control Variables” penceresi görüntüsü.

2.1.4.3. Log Variables Penceresi

Kullanıcı benzetimini gerçeklerken kaydettiği “LogVariable” tipindeki değişkenler bu pencerede gösterilir. Bu pencerede her “ControlVariable” değişkeni

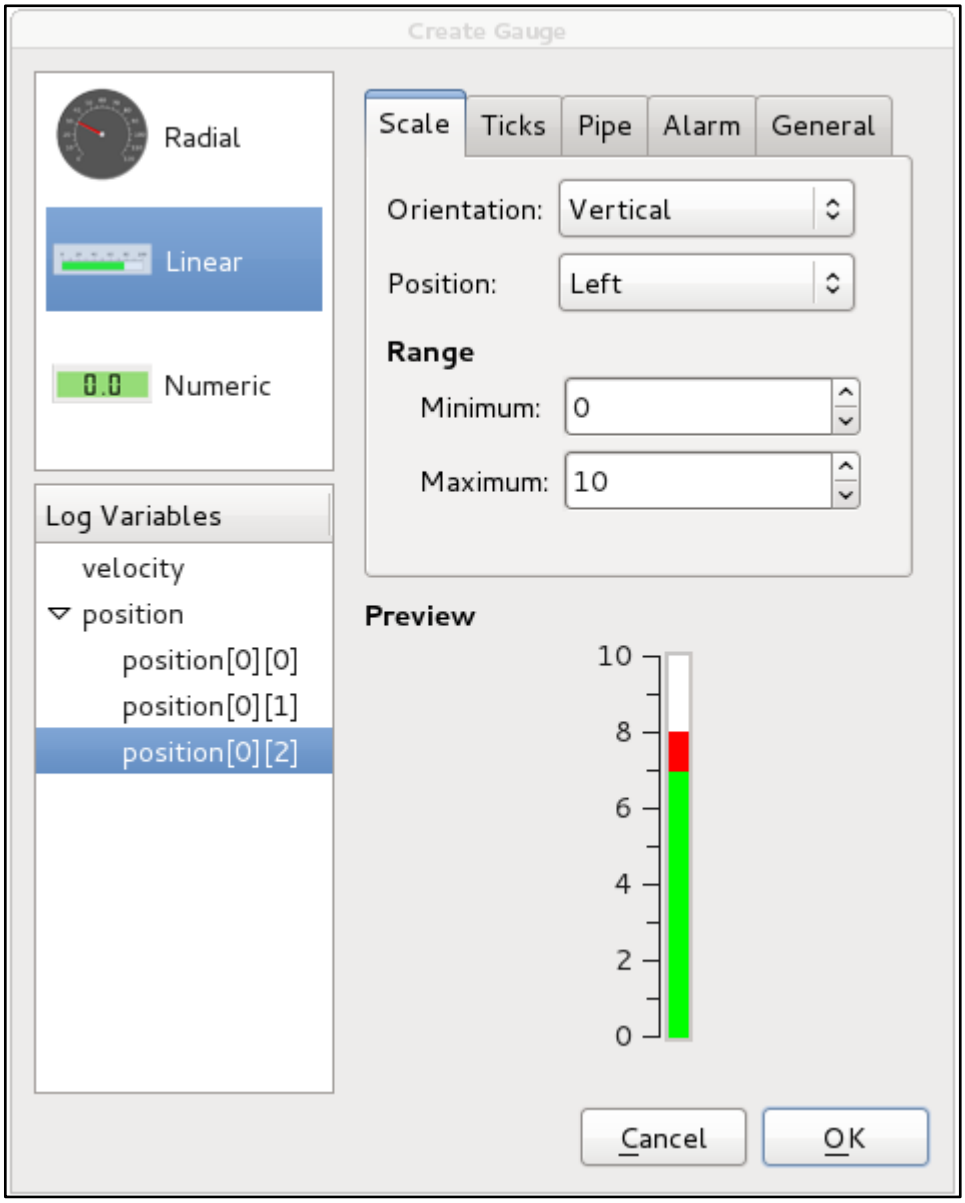
için bir satır bulunur. Bu pencerede kullanıcı, değerini takip etmek istediği değişkenlerin örnekleme parametrelerinde değişiklik yapılabilir. Her bir "LogVariable" için örnekleme frekansı, örnekleme başlangıç zamanı ve örnekleme süresi belirlenebilir. "Log Variables" penceresinin ekran görüntüsü Şekil 2.4 ile gösterilmiştir.



Şekil 2.4: "ZenomGUI" – "Log Variables" penceresi görüntüsü.

2.1.4.4. Create Gauge Penceresi

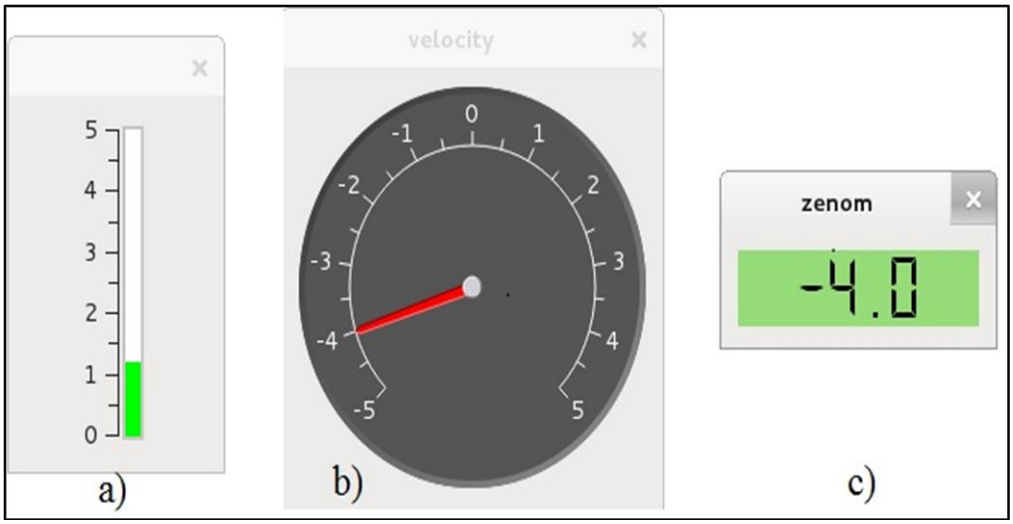
Kullanıcı benzetimini gerçeklerken kaydettiği "LogVariable" tipindeki değişkenlerin anlık değerlerini takip etmek için kullanacağı arayüzleri belirlemesine imkân veren penceredir. Kullanıcı bu pencerede takip etmek istediği "LogVariable" değişkenini ve görselleme tipini seçer. Ayrıca görselleme tipine göre en yüksek, en düşük ve kritik değer belirlenebilir, görselleme renkleri değiştirilebilir. "Create Gauge" penceresinin ekran görüntüsü Şekil 2.5 ile gösterilmiştir.



Şekil 2.5: "ZenomGUI" – "Create Gauge" penceresi görüntüsü.

2.1.4.5. Gauge Pencereleri

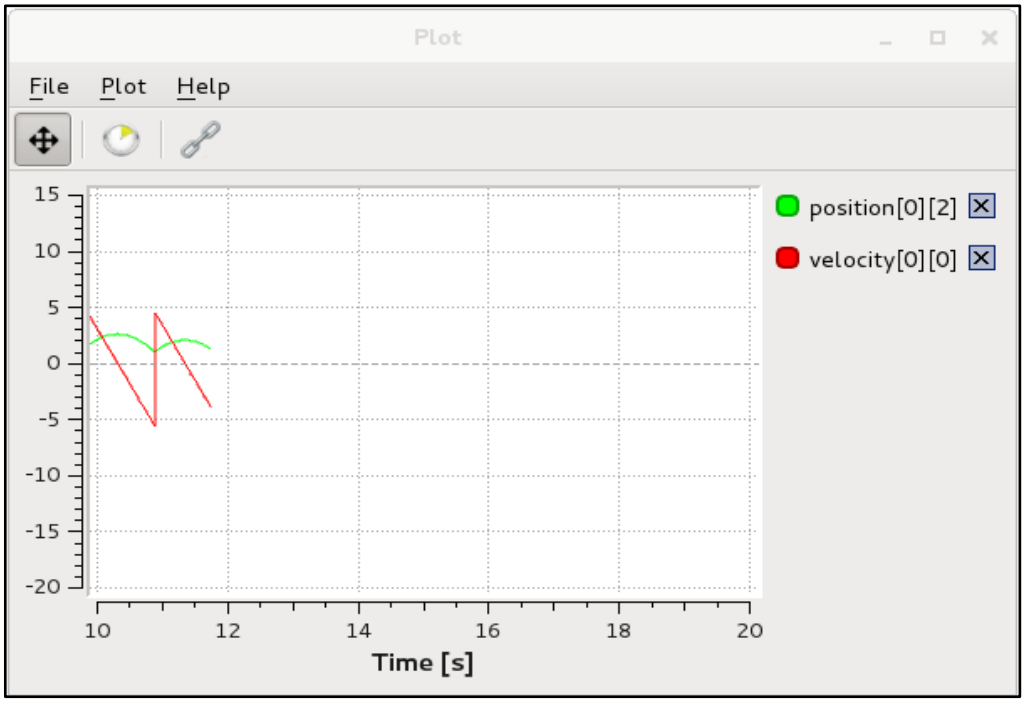
"Create Gauge" penceresi ile açılan pencerelerdir. Kullanıcının seçtiği "LogVariable" değişkenin anlık değerini görsellerler. Kullanıcının seçtiği tipe göre, sayısal, doğrusal ve dairesel gösterim yapabilirler. "Gauge" pencerelerinden lineer gösterge ekranı Şekil 2.6.a) ile dairesel gösterge ekranı Şekil 2.6.b) ile sayısal gösterge ekranı Şekil 2.6.c) ile gösterilmiştir.



Şekil 2.6: "ZenomGUI" – a) Lineer gösterge ekranı, b) Dairesel gösterge ekranı, c) Sayısal gösterge ekranı.

2.1.4.6. Plot Penceresi

Kullanıcının kaydettiği "LogVariable" değişkenlerini 2 boyutlu grafikler ile görselleyen penceredir. Aynı anda birden fazla değişkenin değeri bir pencere içinde görsellenebilir. Grafikler belli bir zaman penceresi gösterecek şekilde çalışır. Kullanıcı zaman penceresinin süresini, grafikteki eğrilerin renklerini değiştirebilir. Ayrıca değişkenlerin zamana göre değişim bilgisini "MATLAB" formatında çıktı olarak üretebilmektedir. "Plot" penceresinin geliştirmesinde Qt kütüphanesi ile geliştirilmiş "Qwt" [Web 20, 2014] kütüphanesi kullanılmıştır. "Qwt" kütüphanesi 2 boyutlu grafiksel arayüz geliştirmek yaygın olarak kullanılan bir kütüphanedir [Maschotta, 2013]. "Plot" penceresinin ekran görüntüsü Şekil 2.7 ile gösterilmiştir.

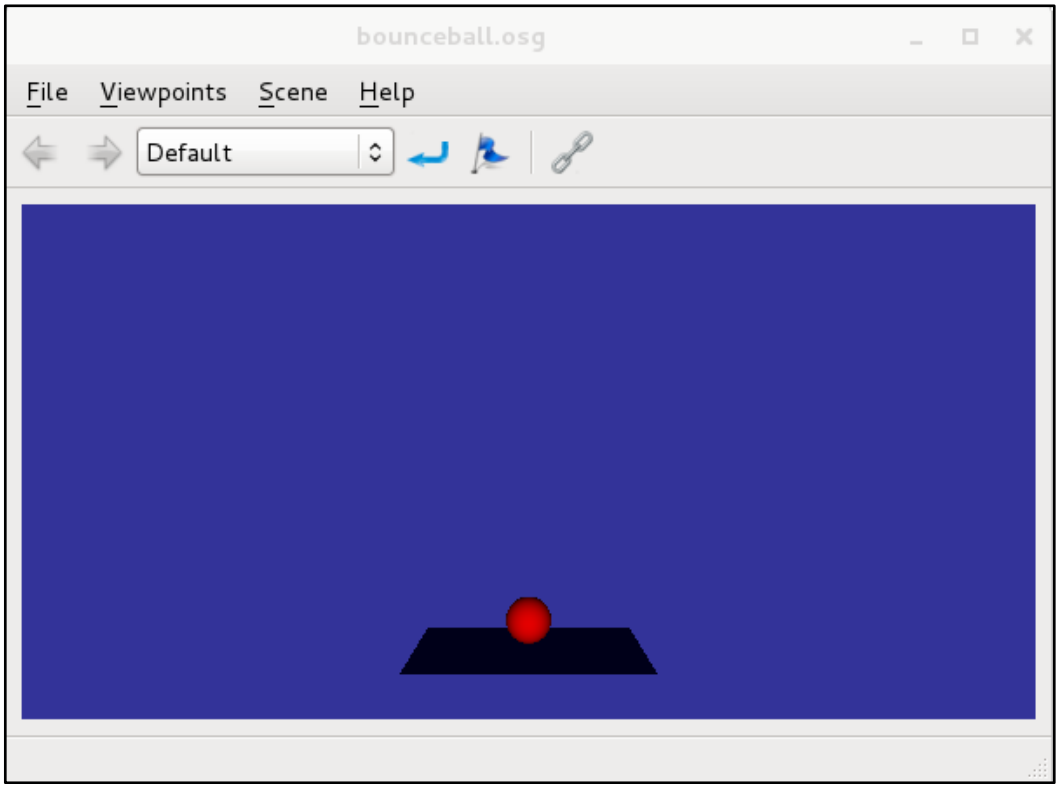


Şekil 2.7: "ZenomGUI" – "Plot" penceresi görüntüsü.

2.1.4.7. Scene Penceresi

Kullanıcının kaydettiği "LogVariable" değişkenlerinin 3 boyutlu VRML formatındaki model dosyası ile eşleştirip, değişken değerlerindeki değişikliklerin model üzerinde takip edilmesine imkân sağlayan penceredir. Kullanıcı kullanmak istediği model dosyasını yükler ve model içindeki konum, renk, ışık ve rotasyon gibi parametreleri, Zenom'da tanımladığı "LogVariable" değişkenleri ile eşleştirir. "LogVariable" değişkenlerindeki değişiklikler model içindeki değerlere yansıtılır. Bu sayede benzetim sonuçlarındaki değişiklikler 3 boyutlu model üzerinden takip edilebilir.

"Scene" penceresinde kullanıcı kamera oluşturabilir ve bu kameraları kaydedebilir. Koşum esnasında kayıtlı kameralar arasında geçişler yapılabilmektedir. Bu özellik sayesinde kullanıcı 3 boyutlu sahneyi birden fazla görüş noktasından takip edebilir ve dikkat etmek istediği detaylara hızlıca erişebilir. "Scene" penceresinin ekran görüntüsü Şekil 2.8 ile gösterilmiştir.



Şekil 2.8: "ZenomGUI" – "Scene" penceresi görüntüsü.

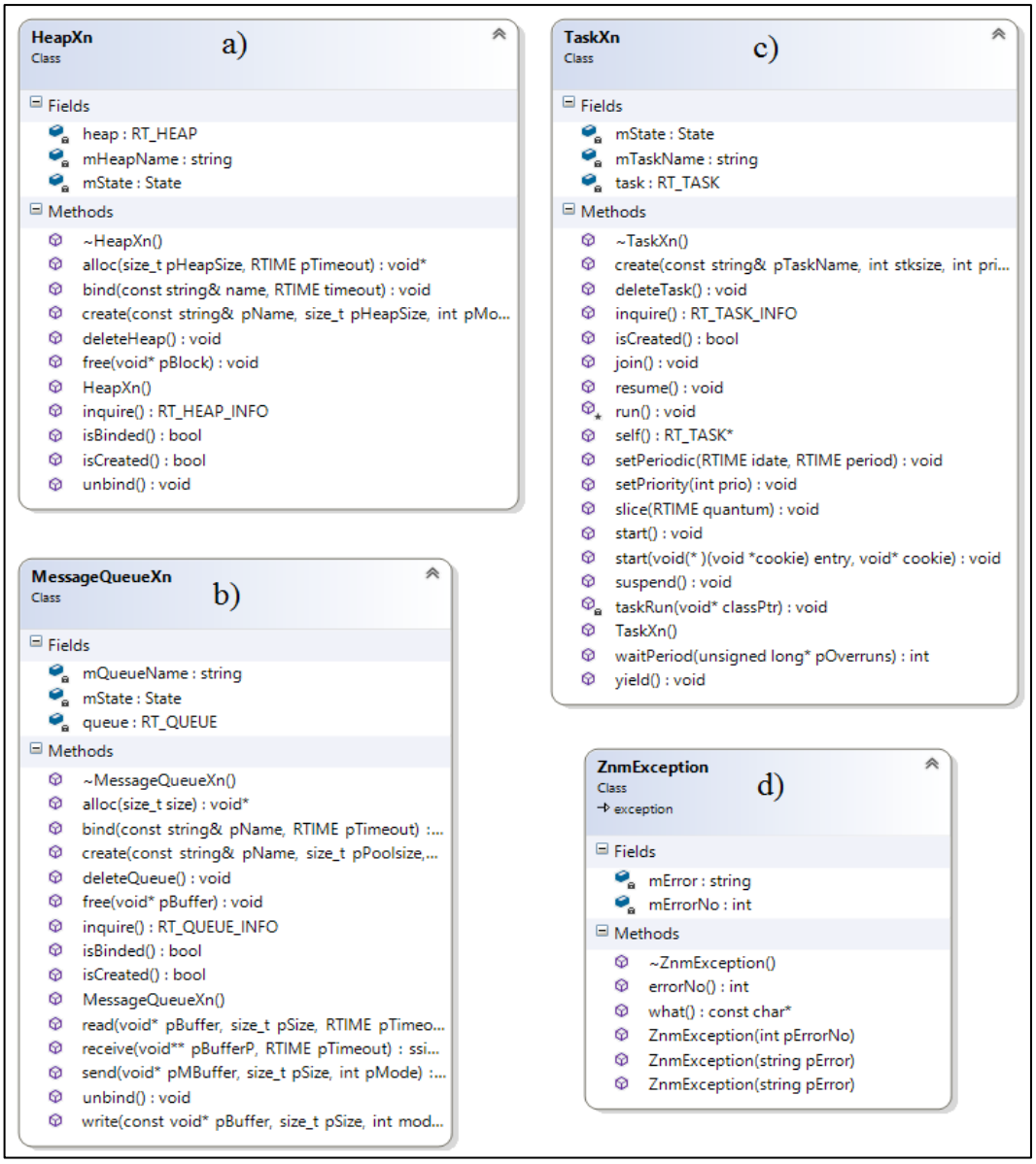
2.2. Zenom Tasarımı

Zenom benzetim ortamı modüler bir yapıda tasarlanmış ve benzer işlevleri yerine getiren sınıflar kütüphaneler içine toplanmıştır. Geliştirilen kütüphaneler kullanılarak Zenom benzetim ortamı gerçeklenmiştir. Geliştirilen kütüphaneler projeye dinamik olarak bağlanmıştır. Bu sayede hata düzeltmeleri ve kodlama arayüzü değiştirmeyen kütüphane iyileştirmeleri, kullanıcının gerçekleştiği benzetimlere yeniden derleme gerekmeden yansıtılabilmektedir.

2.2.1. Araçlar Kütüphanesi

"Zenom Tools" kütüphanesi "Xenomai" servislerini kullanan sınıflarının toplandığı kütüphanedir. Bu kütüphanede bulunan "TaskXn" sınıfı, gerçek zamanlı izlek oluşturan sınıftır. . "TaskXn" sınıfının diyagramı Şekil 2.9.c) ile gösterilmiştir. "HeapXn" sınıfı "Xenomai" tarafından kontrol edilen paylaşılan hafıza bloğu oluşturmakla sorumlu sınıftır. "HeapXn" sınıfının diyagramı Şekil 2.9.a) ile gösterilmiştir. "MessageQueueXn" sınıfı, mesajlaşma için kullanılan mesaj kuyruğu

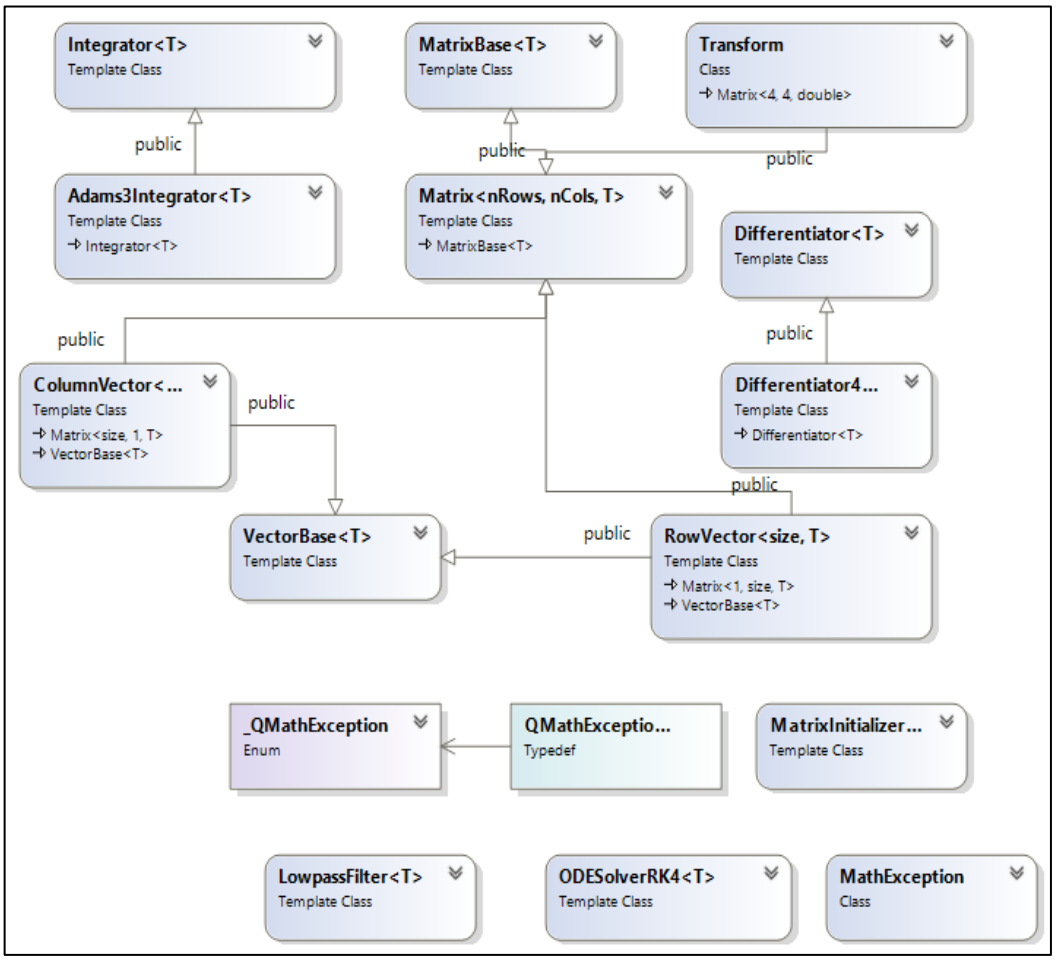
yapısını oluşturmak için kullanılan sınıftır. “MessageQueueXn” sınıfının diyagramı Şekil 2.9.b) ile gösterilmiştir. Zenom projesinde doğrudan “Xenomai” servis çağrımlarının yapıldığı bütün sınıflar kütüphane içinde bulunmaktadır. “Xenomai” servislerinde değişiklikler olması ya da farklı bir servis altyapısı kullanılması durumunda, sadece bu kütüphanenin güncellenmesi Zenom benzetim ortamının çalışmaya devam etmesi için yeterli olacaktır. Kütüphane içinde oluşabilecek hataların tanımlanması için “ZnmException” sınıfı kütüphaneye eklenmiştir. . “ZnmException” sınıfının diyagramı Şekil 2.9.d) ile gösterilmiştir.



Şekil 2.9: Zenom – a) “HeapXn” sınıf diyagramı, b) “MessageQueueXn” sınıf diyagramı, c) “TaskXn” sınıf diyagramı, d) “ZnmException” sınıf diyagramı.

2.2.2. Matematik Kütüphanesi

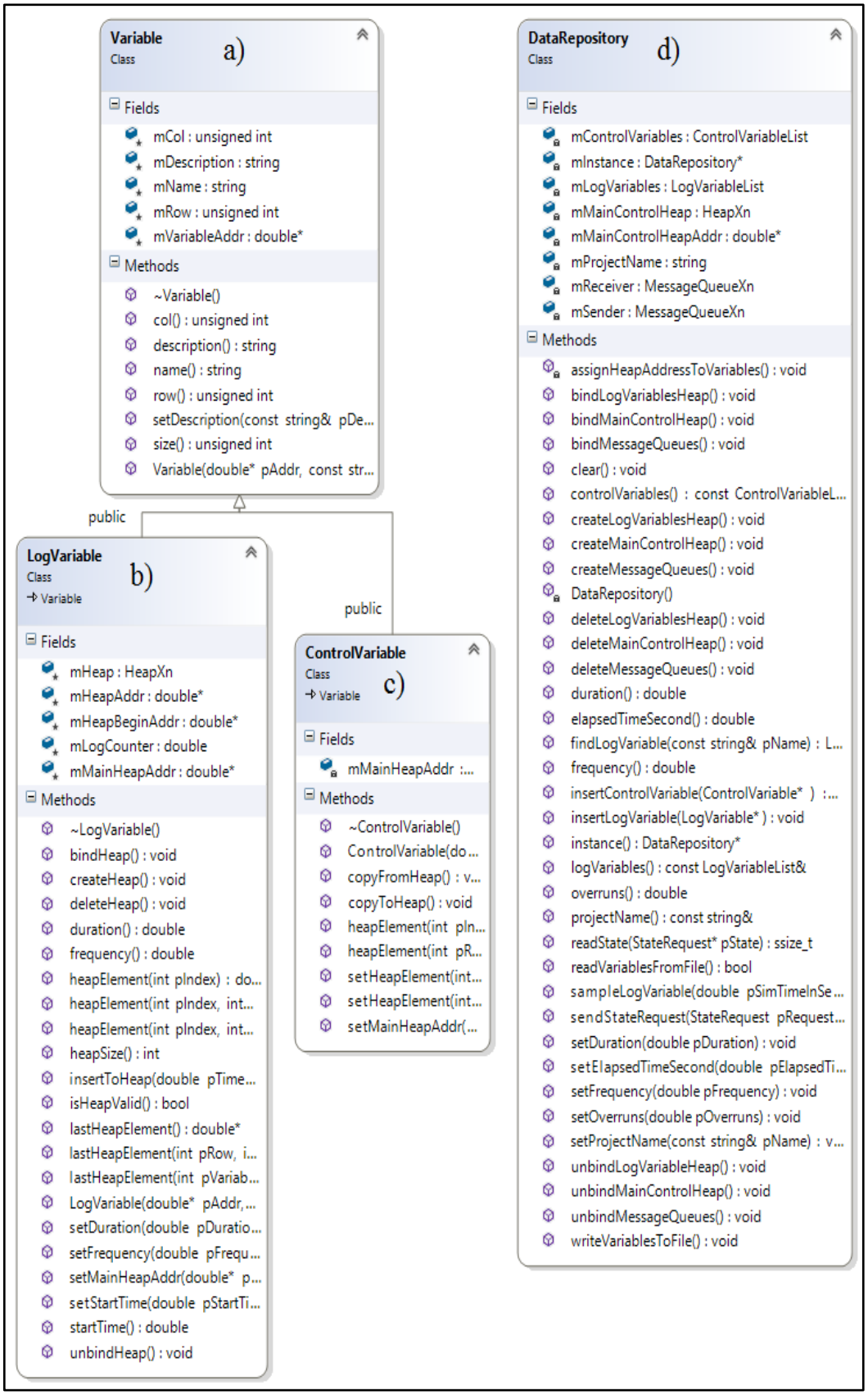
"Zenom Math" kütüphanesi matematiksel hesaplamalara yardımcı olan sınıfların toplandığı kütüphanedir. Zenom benzetim ortamı dışında başka projelere eklenerek kullanılabilecek şekilde geliştirilmiştir. Türev ve integral alma, matris çarpımı gibi işlemleri gerçekleştirmektir. Zenom proje oluşturma aracı oluşturan projede, matematik kütüphanesi projesi bağlanmıştır ve kullanıcı kullanmak istediği sınıfın tanımlama dosyasını kod içine ekleyerek kullanabilmektedir. "Zenom Math" kütüphanesinin bileşenleri Şekil 2.10 ile gösterilmiştir.



Şekil 2.10: Zenom – "Zenom Math" kütüphanesi bileşenleri.

2.2.3. Çekirdek Kütüphanesi

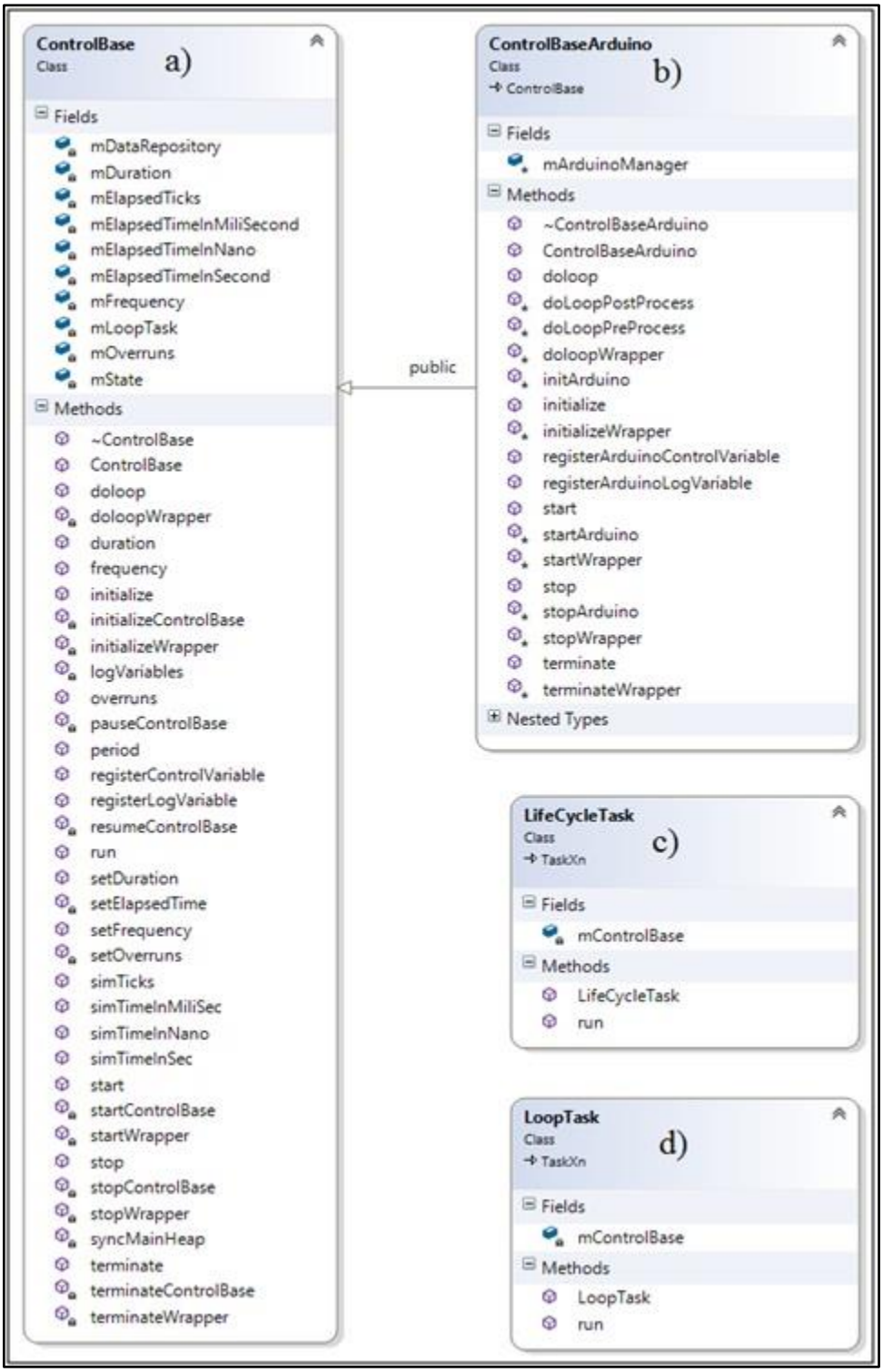
"Zenom Core" kütüphanesi Zenom benzetim ortamının temeli olan "ControlVariable" ve "LogVariable" sınıflarının tanımlandığı kütüphanedir. "ControlVariable" ve "LogVariable" sınıfları "Variable" sınıfından türetilerek gerçeklenmiştir. "LogVariable" sınıfının diyagramı Şekil 2.11.a) ile gösterilmiştir. "LogVariable" sınıfı geçmiş verileri tutmak zorunda olduğu için, paylaşılmış hafıza üretme ve bu hafızayı yönetme işlemlerini de gerçekleştirmektir. "LogVariable" sınıfının diyagramı Şekil 2.11.b) ile "ControlVariable" sınıfının diyagramı Şekil 2.11.c) ile gösterilmiştir. "ControlBase" ve "ZenomGUI" uygulamalarının değişkenlere erişimini kolaylaştırmak ve hafıza yönetimini merkezileştirmek "Repository Pattern" [Web 21, 2014] tasarım deseni kullanılarak "DataRepository" sınıfı gerçeklenmiştir. "DataRepository" sınıfı gerçeklenirken "Singleton Pattern" [Web 22, 2014] tasarım deseni kullanılarak uygulama içinde tekil olması ve kod içinde her yerden erişim yapılabilmesi sağlanmıştır. Bu sayesinde "ZenomGUI" uygulamasındaki arayüz sınıflarının görselleyeceği değişkenlere erişimi kolaylaşmıştır. "DataRepository" sınıfı uygulamanın başlama anında "LogVariable" ve "ControlVariable" için gerekli olan hafıza bloklarının sisteme kaydedilmesi, arayüz ve benzetim uygulamalarının alınan hafıza bloklarına erişimi gibi görevlerden de sorumludur. Uygulamanın çalışma süresi boyunca kullanılan dinamik hafızanın yönetimi, hafıza durumu ile bilgilerin uygulamalar arasında paylaşılması "DataRepository" sınıfı ile yapılır. "DataRepository" sınıfının diyagramı Şekil 2.11.d) ile gösterilmiştir.



Şekil 2.11: Zenom – a) "Variable" sınıf diyagramı, b) "LogVariable" sınıf diyagramı, c) "ControlVariable" sınıf diyagramı, d) "DataRepository" sınıf diyagramı.

2.2.4. Benzetim Kütüphanesi

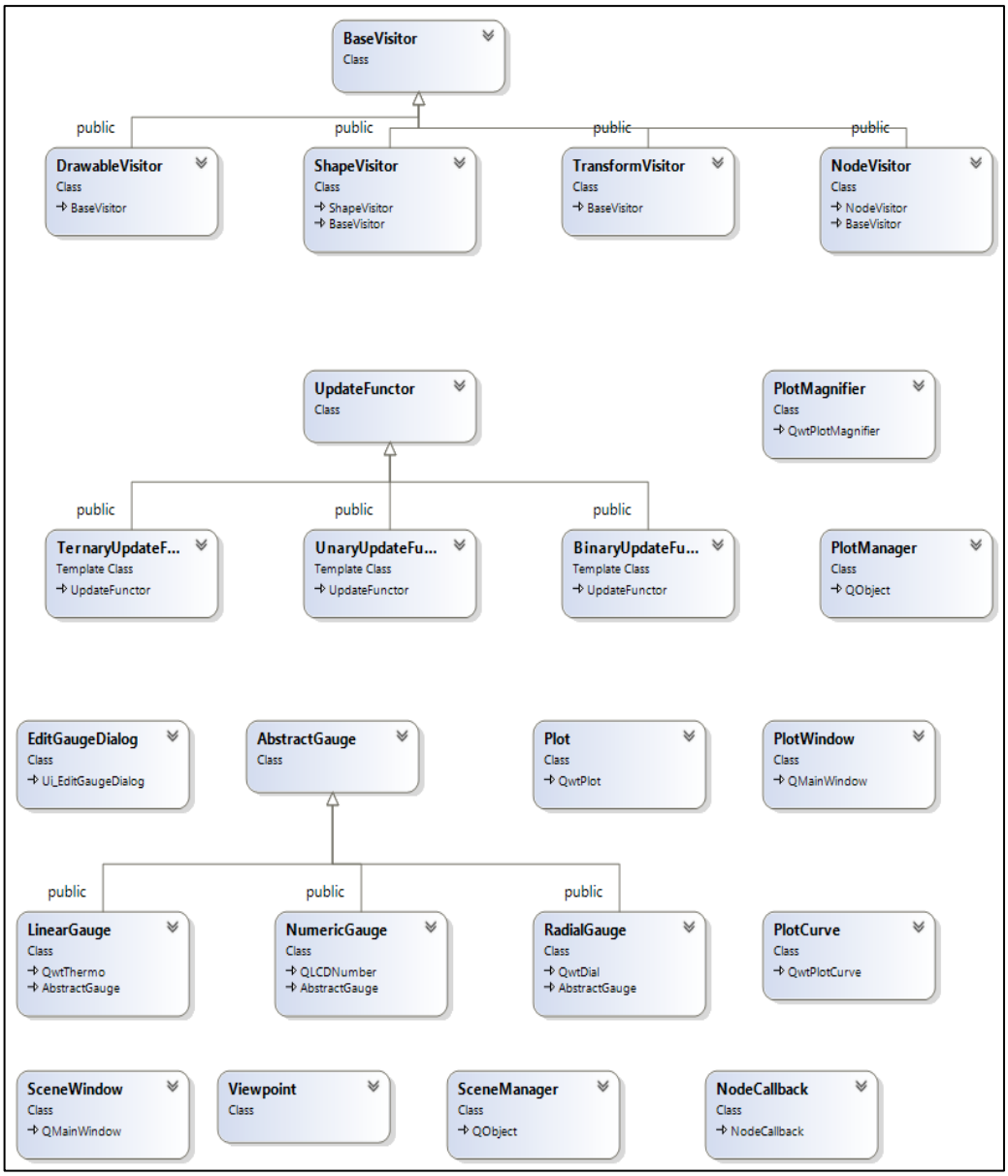
"Zenom ControlBase" kütüphanesi Zenom benzetim ortamının gerçek zamanlı çalışan uygulamanın türetildiği sınıfları içeren kütüphanedir. Bu kütüphane üç temel sınıftan oluşmaktadır. Bu sınıflar "ControlBase", "LifeCycleTask" ve "LoopTask" sınıflarıdır. Kullanıcının beş metodu gerçekleyerek oluşturduğu benzetim uygulaması "ControlBase" sınıfından türetilmektedir. "ControlBase" sınıfını "Template Method Pattern" [Web 23, 2014] tasarım deseni kullanılarak geliştirilmiştir. Benzetimin temel beş adımı soyut metot olarak tanımlanmıştır ve bu sınıftan türetilen sınıflar, bu metotları gerçekleyerek oluşturulmak zorundadır. "ControlBase" sınıfı benzetim geliştiren kullanıcıya benzetim koşumu hakkında bilgi veren metotlar içerir. Kullanıcı bu metotlar ile algoritma ihtiyaç duyabileceği koşum frekansı, koşum süresi, koşum saati, gerçek zamanlı çalışmanın korunması gibi bilgilere ulaşabilir. Ayrıca "LogVariable" ve "ControlVariable" değişkenlerinin benzetime kaydedilme işlevlerinden sorumlu metotlar da bu sınıfta bulunmaktadır. "ControlBase" sınıfına eklenen "Adapter Pattern" [Web 24, 2014] tasarım deseni ile geliştirilen "Wrapper" etiketli metotlar ile "ControlBase" sınıfından koşum süresinde paralel işlemlere izin veren sınıflar türetilebilmektedir. "ControlBase" sınıf diyagramı Şekil 2.12.a) ile gösterilmiştir. "Arduino" platformu ile Zenom benzetim ortamını birlikte kullanmayı sağlayan sınıflardan birisi olan "ControlBaseArduino" sınıfı, "Wrapper" etiketli sınıflar gerçeklenerek geliştirilmiştir. "Wrapper" etiketli metotlar gerçeklenerek, benzetim adımlarının tanımlandığı beş metot ile birlikte farklı işlemler de koşum süresince çalıştırılabilmektedir. "ControlBaseArduino" sınıfı benzetim koşumu süresince, "doloop" metodunun başında ve sonunda seri port haberleşmesi için gereken işlemleri çalıştırmaktadır. "ControlBaseArduino" sınıf diyagramı Şekil 2.12.b) ile gösterilmiştir. "Zenom ControlBase" kütüphanesinde bulunan "LifeCycleTask" sınıfı kendine bir izlek oluşturur ve benzetim koşumu süresince ZenomGUI uygulamasında gelen koşum döngüsü bilgisini okur. Koşum döngü bilgisine göre "ControlBase" uygulaması başlatabilir, sonlandırabilir ya da duraklatabilir. "LoopTask" sınıfı oluşturduğu izlek ile "ControlBase" sınıfının periyodik çağrılması gereken metotlarını çağırır ve koşum zaman bilgisini hesaplar. "LifeCycleTask" sınıf diyagramı Şekil 2.12.d) ile "LoopTask" sınıf diyagramı Şekil 2.12.d) ile gösterilmiştir.



Şekil 2.12: Zenom – a) “ControlBase” sınıf diyagramı, b) “ControlBaseArduino” sınıf diyagramı, c) “LifeCycleTask” sınıf diyagramı, d) “LoopTask” sınıf diyagramı.

2.2.5. Arayüz Kütüphanesi

"Zenom GUI" kütüphanesi Zenom benzetim ortamının kullanıcı arayüzünü oluşturan ZenomGUI uygulamasının kullandığı sınıfları içeren kütüphanedir. "Qt", "OpenSceneGraph" ve "Qwt" kütüphaneleri kullanılarak geliştirilmiştir. Benzer yapıdaki arayüzleri oluşturan sınıflar ortak bir sınıftan türetilmiştir. Değişkenlerin anlık değerlerini gösteren arayüzler "AbstractGauge" sınıfından türetilerek geliştirilmiştir. Zenom benzetim ortamına farklı tipte anlık değer gösteren arayüz eklemek için bir "AbstractGauge" sınıfından bir sınıf türetmek ve bu sınıf içinde arayüzü belli bir değer için çizimi yapacak metodu gerçeklemek yeterlidir. Arayüz ile "LogVariable" arasında ilişkinin kurulması ya da arayüz seçim ekranın güncellenmesi gibi işlemler ata sınıf tarafından yapılmaktadır. 2 boyutlu grafik çizimlerinde grafiklerde görsellenen eğriler ve grafiğin bulunduğu arayüz birbirinden ayrı geliştirilmiştir. Grafiği görselleyen arayüz, grafik eğrisi sınıfından nesneleri yönetecek şekilde tasaralanmıştır. İki boyutlu grafiklerde farklı eğriler görsellemek için "PlotCurve" sınıfından bir sınıf türetilir ve bu sınıfın eğriyi çizen metodu gerçeklenir. 2 boyutlu grafik arayüzündeki "LogVariable" ile eşleme, çizim rengi, çizim boyutu, otomatik boyutlandırma gibi işlemler gerçeklenen eğri sınıfı için de kullanılabilir olur. "Zenom GUI" kütüphanesinin temel bileşenleri Şekil 2.13 ile gösterilmiştir.



Şekil 2.13: Zenom – "Zenom GUI" kütüphanesi temel bileşenleri.

2.3. Örnek Uygulama

Zenom benzetim ortamının kullanımını anlatmak için "BounceBall" örneği gerçeklenmiştir. Bu örnek uygulama ile proje oluşturma, "LogVariable" ve "ControlVariable" tanımlamaları, "ControlBase" içindeki beş metodun gerçeklenmesi ve benzetimin "ZenomGUI" ile çalıştırılması anlatılmıştır.

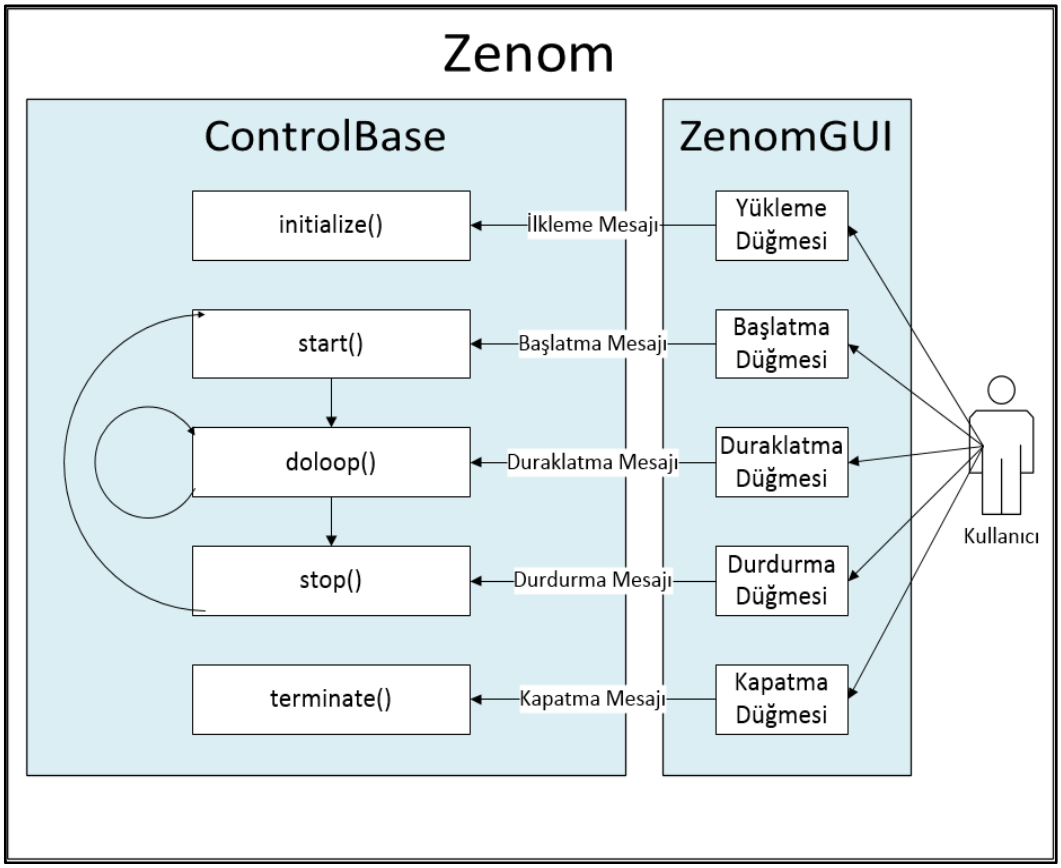
2.3.1. Proje Oluşturma

Zenom benzetim ortamında proje oluşturmak için “Zenom Project” uygulaması kullanılır. Bu uygulama ile parametre olarak verilen isimde bir klasör oluşur. Klasör içinde kullanıcının gerçeklemesi gereken beş metodun bulunduğu “.cpp” uzantılı kod dosyası, proje yapılandırma dosyası ve derleme için kullanılacak “makefile” dosyası bulunur.

Örnek uygulama “BounceBall” olarak adlandırıldığı için “Zenom Project” “BounceBall” parametresi ile çalıştırır ve proje klasörü oluştur.

2.3.2. ControlBase Gerçeklenmesi

“BounceBall” projesi çalıştırılabilir bir benzetim olması için “Zenom Project” uygulamasının oluşturduğu klasör içindeki “main.cpp” dosyasında bulunan, benzetimin yaşam döngüsü içinde çağrılması gereken beş metodun gerçeklenmesi gerekir. Zenom benzetim ortamındaki çalışan bir benzetim yaşam döngüsü Şekil 2.14 ile gösterilmiştir.



Şekil 2.14: Zenom – Benzetim yaşam döngüsü görüntüsü.

"BounceBall" örneğinde topun zamana göre hareketi hesaplanır. Benzetim koşum esnasında topun güncel pozisyonunu ve hızını hesaplar. Bu değerlerin benzetim koşumu süresince takip edilmesi gerekmektedir. Değerleri takip edebilmek için konum ve hız bilgisini tutan değişkenleri "LogVariable" olarak tanımlanır. Bu örnekte yerçekimi kuvveti ve topun zemine çarpıp tekrar sekmesi esnasındaki enerji kaybı çarpanı ise kullanıcının değiştirilebileceği değerler olarak tanımlanmıştır. Bu değerleri tutan değişkenler ise "ControlVariable" olarak tanımlanır.

Topun zamana göre hareketini hesaplamak için gerekli olan matematik işlemleri için Zenom projesi içinde bulunan "ZenomMath" kütüphanesinin "Integrator" bileşeni kullanılmıştır.

"BounceBall" örneğinde sistemden herhangi bir kaynak kullanılmadığı için "stop()" ve "terminate()" aşamalarında herhangi bir işlem gerekmemektedir. Bu nedenle bu iki metot "Zenom Project" uygulamasının oluşturduğu "main.cpp" içindeki haliyle bırakıldı. "int initialize()", "int start()" ve "int doloop()" metotları gerçeklendi. Gerçeklenen "BounceBall" örneğinin sınıf tanımlama kodu Şekil 2.15

ile gösterilmiştir. Gerçeklenen "initilize()" methodu Şekil 2.16, "start()" metodu Şekil 2.17, "doloop()" metodu Şekil 2.18 ile gösterilmiştir.

```
class BounceBall : public ControlBase
{
public:
    /*
    *   no change in this section, same as main.cpp
    */
private:
    Integrator< double > mVelocityIntegrator;   // velocity integrator
    Integrator< double > mPositionIntegrator;   // position integrator

    // ----- Log Variables -----
    double velocity;    // velocity of ball
    double position[3]; // position of ball

    // ----- Control Variables -----
    double g;       // gravity
    double cor;     // coefficient of restitution
};
```

Şekil 2.15: Zenom – "BounceBall" örneği sınıf tanımlası görüntüsü.

```
int BounceBall::initialize()
{
    registerLogVariable(&velocity, "velocity"); // hiz değerini tutan LogVariable
    registerLogVariable(position, "position", 1, 3); // konum değerini tutan LogVariable

    registerControlVariable(&g, "g"); // yercekimi kuvveti değerini tutan ControlVariable
    registerControlVariable(&cor, "cor"); // sönümleme katsayısını tutan ControlVariable

    g = -9.81;   // m/s^2
    cor = -0.8;  // coefficient of restitution

    position[0] = 0;
    position[1] = 0;

    return 0;
}
```

Şekil 2.16: Zenom – "BounceBall" örneği "initialize()" metodu görüntüsü.

```
int BounceBall::start()
{
    mVelocityIntegrator.setSamplingPeriod(period());
    mVelocityIntegrator.reset(0);  // initial speed 0 m/s

    mPositionIntegrator.setSamplingPeriod(period());
    mPositionIntegrator.reset(15.0);  // initial height 15 m

    return 0;
}
```

Şekil 2.17: Zenom – "BounceBall" örneği "start" metodu görüntüsü.

```
int BounceBall::doloop()
{
    double ballRadius = 1;

    velocity = mVelocityIntegrator.integrate(g);
    position[2] = mPositionIntegrator.integrate(velocity);   // position of ball

    // saturation
    if (position[2] <= ballRadius)
    {
        position[2] = ballRadius;
    }

    // Hits the floor
    if (position[2] == ballRadius)
    {
        mVelocityIntegrator.reset(velocity * cor);
        mPositionIntegrator.reset(ballRadius);
    }

    return 0;
}
```

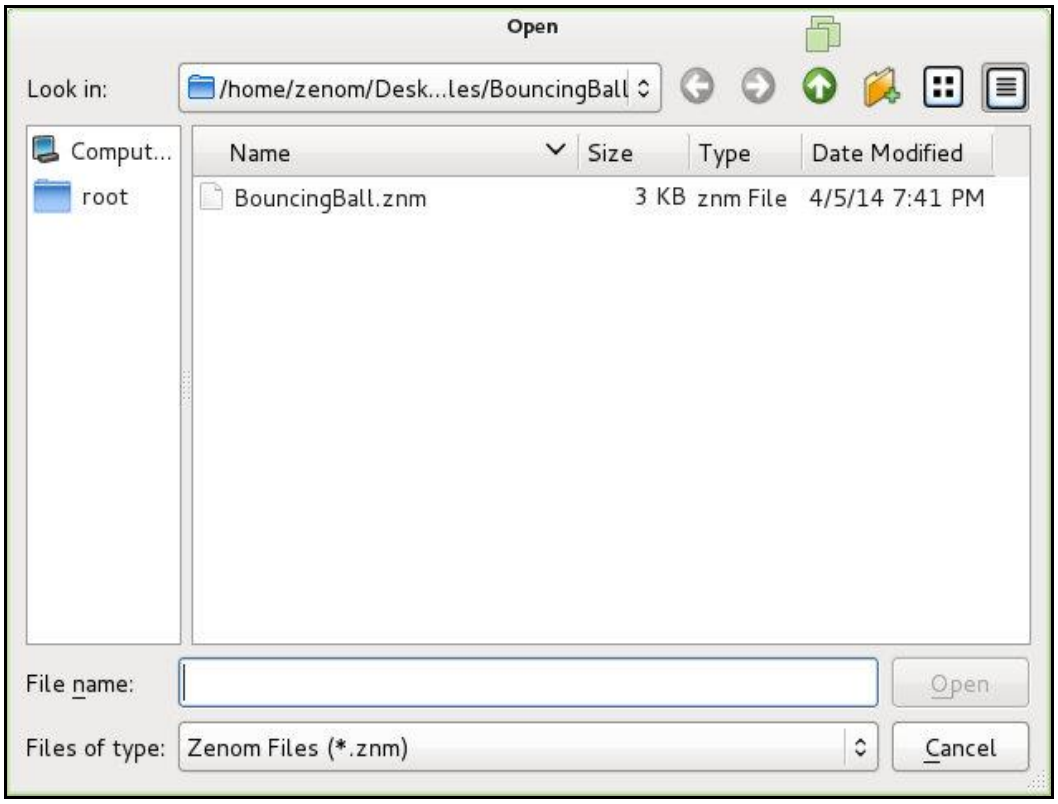
Şekil 2.18: Zenom – "BounceBall" örneği "doloop()" metodu görüntüsü.

"BounceBall" projesi klasöründeki "main.cpp" dosyası şekillerde gösterilen biçimde gerçeklendikten sonra kaydedilir ve "make" çağımı yapılarak derlenir. Derleme işlemi tamamlandığında "BounceBall" klasörü içinde ".znm" uzantılı çalıştırılabilir bir dosyanın oluştuğu gözlenir. Bu dosya "ZenomGUI" ile çalıştırılabilir.

2.3.3. Örneğin Çalıştırılması

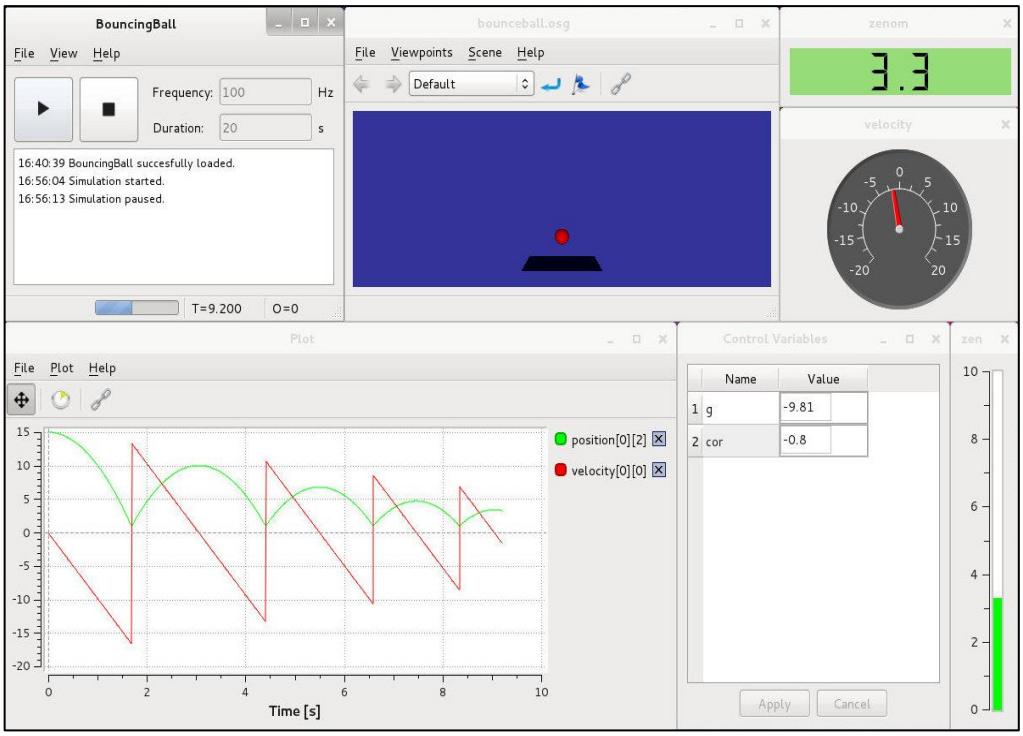
"BounceBall" projesinin çalıştırılması için derlendikten sonra oluşan ".znm" uzantılı çalıştırılabilir dosyanın "ZenomGUI" uygulaması ile yüklenmesi gerekir. "ZenomGUI" "MainWindow" altındaki "File" menüsünden "Open Project" seçeneği

ile “BounceBall.znm” yüklenir ve benzetim koşuma hazır hale gelir. Benzetim yükleme penceresi ekran görüntüsü Şekil 2.19 ile gösterilmiştir.



Şekil 2.19: “ZenomGUI” – “BounceBall” Örneği yükleme ekranı görüntüsü.

“BounceBall” örneği yüklendikten sonra “ZenomGUI” bölümünde anlatılan arayüzler ile “LogVariable” ve “ControlVariable” değişkenleri grafik ve sahne ekranları ile ilişkilendirir. Zenom projesinin içinde gelen örnek “BounceBall” projesindeki arayüzler önceden ilişkilendirilmiş ve kaydedilmiştir. Kullanıcı arayüzlerdeki değişiklerini projeyi kaydederek saklayabilir. Bu sayede her yüklemede değişken ilişkilendirme ve arayüzleri düzenleme işlemlerini tekrar yapmak zorunda kalmaz. Çalışan “BounceBall” projesi ekran görüntüsü Şekil 2.20 ile gösterilmiştir.



Şekil 2.20: "ZenomGUI" – "BounceBall" örneği ekranı görüntüsü.

3. ARDUINO

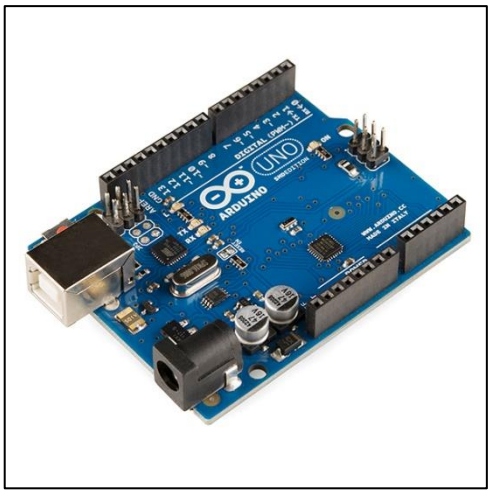
Gelişen teknoloji ile birlikte fiziksel platformlar yüksek satış rakamları ile daha yaygın hale gelmektedir [Web 25, 2014]. Arduino ve "Rasberry Pi" [Web 26, 2014] günümüzce oldukça popüler olan fiziksel platformlardır. Arduino basit yapısı ile donanım projelerinde daha yaygın olarak kullanılmaktadır. "Rasberry Pi" bilgisayara daha yakın yapısı sayesinde yazılım projelerinde kullanılmaya uygundur [Web 27, 2014].

Çalışmamızda kullanacağımız platformun donanım yeteneklerinin daha yüksek olmasına dikkat ettiğimiz için Arduino platformunu kullandık. Ardunio, Giriş/Çıkış kartı ve Process/Wiring dilinin bir uygulamasını içeren geliştirme ortamı özelliklerine sahiptir [Web 28, 2014]. Uygun fiyatı, kullanımı ve programlamasının kolay olması nedeniyle geniş bir kitleye hitap etmektedir [Barrett S, 2013]. Arduino kendi başına çalışabilen interaktif sistemler geliştirmek için kullanılabileceği gibi bilgisayar üzerinde çalışan yazılımlar ile haberleşerek birlikte çalışabilir [Al-Busaidi et al., 2012].

Arduino geliştirilmek istenilen projelerin çeşitlerine göre farklı yapılarda olabilir ama genel olarak bütün modelleri tek kart mikro kontrolcü yapısındadır. Üzerinde USB portu, analog ve sayısal giriş/çıkış birimleri bulunur. Programlama için C/C++ dili kullanılır ve Java üzerinde geliştirilmiş, geliştirme, derleme, yükleme özelliklerine sahip Arduino IDE ile programlanır.

3.1. Arduino Uno

Arduino modelleri arasında en yaygın olarak kullanılan modeldir. Uygun fiyatı ve küçük yapısı ile basit projelerin geliştirilmesinde kullanılmaktadır. 2010 yılında piyasaya sürülmüş ve şu an üçüncü revizyonu satılmaktadır. Üzerinde bir adet USB portu, 16 MHz ATmega328P işlemcisi ve programlanabilir 32 KB hafızaya sahiptir. 14 adet sayısal ve 6 adet analog giriş/çıkış birimileri bulunur [Web 29, 2014]. Arduino Uno fiziksel platformunun görüntüsü Şekil 3.1 ile gösterilmiştir.



Şekil 3.1: Arduino – Arduino uno görüntüsü.

3.2. Arduino IDE

Arduino kullanımı ve programlaması kolay bir platform olarak tanınmaktadır. Bu özelliğini kullanımı kolay bir IDE ve C++ dilinde programlanabilir olmasına borçludur. Arduino proje açık kaynak kodludur ve birçok geliştirme kütüphanesi ücretsiz olarak kullanılabilmektedir. Temel işlemler için ihtiyaç duyulacak bir çok kütüphane Arduino projesine dâhil edilmiştir. Ayrıca Arduino IDE Windows, Linux, Macintosh OSX işletim sistemlerine destek vermektedir. Programlama için C++ dili kullanılır ve C++ dilinin sahip olduğu nesneye dayalı programlama avantajlarına sahiptir. Arduino programlanırken kullanıcıdan iki metodu gerçeklemesi beklenir. Bu metotlardan birisi cihaz çalışmaya başladığında sadece bir kez çalıştırır. Bu metodun imzası "void setup()" olarak belirlenmiştir ve tanımlama, ilk değer atama gibi işlemler bu metot içinde yapılır. Gerçeklenmesi gereken diğer metot ise "void loop()" imzalı metottur. Bu metot içinde kullanıcı cihaz çalıştığı sürece yapılmasını istediğini işlemleri gerçekler ve cihaz çalıştığı süre boyunca bu metot çağırılır. Kullanıcı iki metodu gerçekledikten sonra kod dosyasını Arduino IDE ile derleyip, cihaza yükleyebilmektedir.

Bir buton ve bir LED ışık kaynağı ile oluşturulan Arduino devresini kontrol eden Arduino kodu Şekil 3.2 ile gösterilmiştir. Bu örnekte devredeki butona basıldığında LED ışık kaynağı ışık vermektedir. Butona basılmadığı sürece LED ışık kaynağı ışık vermemektedir.

```arduino
const int buttonPin = 2;
const int ledPin = 13;

int buttonState = 0;

void setup()
{
    pinMode(ledPin, OUTPUT);
    pinMode(buttonPin, INPUT);
}

void loop()
{
    buttonState = digitalRead(buttonPin);
    if (buttonState == HIGH)  // HIGH ise buton basili
    {
        digitalWrite(ledPin, HIGH);
    }
    else
    {
        digitalWrite(ledPin, LOW);
    }
}
```

Şekil 3.2: Arduino – Arduino örnek kod görüntüsü.

4. ZENOM-ARDUINO

Zenom benzetim ortamının kolay geliştirilebilir yapısını ve gerçek zamanlı benzetimlerin ihtiyaç duyduğu donanım ile iletişim kurabilme yeteneklerini göstermek için Zenom benzetim ortamına Arduino fiziksel programlama platformunu entegre edecek bileşenler geliştirilmiştir. Bu bileşenler ile Zenom için benzetim geliştirme yapısı ile aynı yapıda Arduino ile haberleşebilen benzetimler geliştirilebilmektedir.

Zenom ile Arduino entegre edilirken, "ZenomGUI" uygulamasında değişiklik yapılmamıştır. Kullanıcının gerçeklediği "ControlBase" uygulaması üzerine Arduino ile haberleşecek ControlBaseArduino uygulaması geliştirilmiş ve Arduino ile etkileşimi sağlayacak değişkenler "LogVariable" ve "ControlVariable" yapısı içine dâhil edilmiştir. Bu sayede "ZenomGUI", çalıştırdığı uygulamanın "ControlBase" ya da ControlBaseArduino uygulaması olmasından etkilenmeden işlevlerini yerine getirebilmektedir.

Zenom ile Arduino haberleşmesini sağlayabilmek için, Arduino Manager adında seri port ile haberleşmeyi sağlayacak, seri porttan gelen verinin yorumlanmasını ve gönderilecek verinin biçimlendirilmesini sağlayacak bir kütüphane geliştirilmiş ve Zenom projesine eklenmiştir.

Kullanıcının geliştireceği "ControlBase" uygulamasının Arduino Manager ile birlikte çalışabilmesi için ControlBaseArduino adında, "ControlBase" sınıfından türetilmiş bir sınıf eklenmiştir. Bu sınıf Arduino ile haberleşme için kullanılacak değişkenleri tanımlama imkânı tanımaktadır ve tanımlanan değişkenlerin Arduino Manager üzerinden Arduino ile iletişimi kontrol etmektedir. Arduino ile Zenom arasında kullanılacak değişkenler "ArduinoLogVariable" ve "ArduinoControlVariable" olarak isimlendirilmiştir. Kullanıcı Arduino üzerinde izlemek istediği değişkenleri "ArduinoLogVariable" olarak tanımlayarak "ZenomGUI" üzerinden görselleyebilmektedir. Kullanıcı Arduino üzerinde çalışan uygulamadaki değişkene müdahale etmek istediğinde bu değişkeni "ArduinoControlVariable" olarak tanımlayarak, "ZenomGUI" üzerinden değerinde değişiklik yapabilmektedir.

Kullanıcının gerçekleyeceği benzetimin projesini oluşturan "Zenom Project" uygulaması temel alınarak "Zenom Target Project" uygulaması geliştirilmiştir.

Geliştirilen bu uygulamaya geçirilen parametre ile ControlBaseArduino tipinde proje üretme imkânı sağlanmıştır. Ayrıca kullanıcının geliştirdiği benzetim kodundan Arduino için gerekli olan uygulama kodunun şablonunu üreten mekanizma "Zenom Target Project" uygulamasına eklenmiştir. Bu mekanizma sayesinde oluşturulan şablon kod, ControlBaseArduino içinde tanımlanan değişkenlerin Arduino içinde kullanılabilmesini sağlamaktadır.

4.1. ArduinoManager

"ArduinoManager", "Target Manager" kütüphanesi üzerine geliştirilmiş, Zenom benzetim ortamın ile Arduino fiziksel programlama platformu arasındaki verilerin biçimlendirilmesi ve veri alışverişini kontrol etmek için geliştirilmiş bir kütüphanedir. Arduino bilgisayar ile seri port üzerinden veri alışverişi yapmaktadır. "Arduino Manager" kütüphanesi seri port ile haberleşmeyi kontrol eden bileşenleri içerir ve seri port üzerinden taşınan verinin bilgisayar ve Arduino tarafından işlenebilmesini sağlayan biçimlendirmeleri yapar.

Arduino cihazı veri tipleri ile bilgisayar veri tipleri arasında farklılıklar bulunmaktadır. Bu nedenle iki cihaz arasında ham veri alışverişi yapılamamaktadır. Platform farklılıklarından dolayı ham veri içindeki bitlerin tamsayı ya da kesirli sayı gibi tiplere çevrilmesi hatalara neden olmaktadır. Örneğin, günümüz bilgisayarında 32 bit programlarda "double" tipi 8 bayt ile ifade edilirken Arduino programlarında "double" tipi 4 bayt ile ifade edilmektedir. Ayrıca tamsayı ve kesir kısımlarını ifade eden bit sayıları da farklı olduğu için bu iki tip verinin arasında kolay bir dönüşüm yapılamamaktadır. Bu nedenle Zenom ile Arduino arasında biçimlendirilmiş karakter dizileri ile veri alışverişi yapılmaktadır.

Seri port ile veri transferi yaparken cihazlar arasında herhangi bir el sıkışma rutini kullanılmamaktadır. Bu nedenle veri transferi esnasında veri paketinin başlangıcı olmayan bir noktasından itibaren veri alınmaya başlayabilir. Bu iki taraf içinde geçerli sorundur. Bu sorunu çözmek için Arduino Manager Kütüphanesi gönderilecek veri paketinin başını ve sonunu işaretleyecek karakterler kullanmaktadır. Kullanıcının tanımladığı "ArduinoControlVariable" değişkenleri, Arduino Manager tarafından, veri başlangıç karakteri, değişken etiketi, değişken verisi ve veri bitiş karakteri şeklinde biçimlendirir ve seri port üzerinden Arduino

cihazına gönderir. Aynı biçimdeki “ArduinoLogVariable” değişkenlerinin verisi ise Arduino Manager tarafından seri porttan okunur ve “ZenomGUI” üzerinde görsellenebilmesi için “LogVariable” tipine çevrilir.

Seri port ile veri transferi karakter dizileri ile yapıldığı için iki tarafta da verinin yorumlanması gerekmektedir. Bilgisayar işlem gücünün yüksel olması nedeniyle karakter dizisi üzerinde yapılan işlemlerin çokluğundan etkilenmemektedir. Ancak Arduino cihazının işlem gücü sınırlı olduğu için veri yorumlama işlemleri üzerinde optimizasyonlar yapmak gerekmektedir. Veri transferi sırasında gönderilen değişken isminin karakter sayısı belirsizdir ve bu veriyi işlemek Arduino üzerinde zaman almaktadır. Bunun önüne geçmek için veri içine değişken ismi yerine değişken etiketi yazılmaktadır. Değişken etiketleri tek karakterden oluşacak şekilde belirlenmiştir. “ArduinoLogVariable” değişkenleri ‘A’ karakterinden, “ArduinoControlVariable” değişkenleri ise ‘a’ karakterinden başlayarak etiketlendirilmiştir. Hem Arduino cihazının işlem gücü ve hafıza kısıtlarından dolayı hem de etiketlerinin çakışmaması için “ArduinoLogVariable” ve “ArduinoControlVariable” değişkenlerinin sayısı 25 ile sınırlandırılmıştır. “Zenom-Arduino” veri alışveriş sırasında verilerin biçimlendirilmiş hali Şekil 4.1 ile gösterilmiştir.

```
// Zenom için gerçeklenen kod içindeki değişken tanımlaması
double logVar1;
double logVar2;
double controlVar1 = 5.0;
double controlVar2 = 25.4;

registerArduinoLogVariable(&logVar1, “logVar1”);
registerArduinoLogVariable(&logVar2, “logVar2”);
< A : 0.1 > // değişken değerini Arduino üzerindeki program belirliyor
< B : 4.5 > // değişken değerini Arduino üzerindeki program belirliyor

registerArduinoControlVariable(&controlVar1, “controlVar1”);
registerArduinoControlVariable(&controlVar2, “controlVar2”);
// Arduino Manager tarafından gönderilen veri mesajları
< a : 5.0 >
< b : 25.4 >
```

Şekil 4.1: “Zenom-Arduino” – Veri alışverişinin biçimlendirilmiş görüntüsü.

Arduino Manager Kütüphanesi işlevlerini yerine getirmek için bir izlek oluşturur. Bu izlek seri port üzerinde veri alışverişini yapar. Seri porttan gelen veriler

sürekli takip edilir ve yeni veri geldikçe yorumlanır ve değerler yerel olarak tutulan değişkenlere atılır. “ControlBase” uygulaması her “doloop()” metodunu çağırdığında seri porttan okunan güncel değerleri yerel değişkenler üzerinden okur ve “ArduinoLogVariable” tipindeki değişkenlerini günceller. “doloop()” metodu sonlanırken “ArduinoControlVariable” tipindeki değişkenlerin değerlerini Arduino Manager üzerinden Arduino cihazına gönderir. Bu yapı sayesinde seri porttan yapılan veri alışverişi sırasında oluşabilecek herhangi bir gecikme “ControlBase” uygulamasının işleyişini etkilemez ve gerçek zamanlı çalışma kriterlerinin dışına çıkmasına neden olmaz.

4.2. ZenomArduinoManager

Zenom benzetim ortamın ile Arduino fiziksel programlama platformu arasında veri alışverişini sağlayabilmek için Zenom tarafında “ArduinoManager” Kütüphanesi geliştirilmiştir. Veri alışverişinin iki taraflı olması nedeniyle Arduino fiziksel platformu üzerinde çalıştırılan uygulamanın Zenom ile uyumlu bir şekilde veri alıp göndermesi gerekmektedir. Bu ihtiyacı karşılamak için Arduino platformu için “ZenomArduinoManager” kütüphanesi geliştirilmiştir.

ZenomArduinoManager kütüphanesi, “ArduinoManager” kütüphanesinde olduğu gibi gönderilip alınacak olan verinin biçimlendirilmesi ve seri port haberleşmesi işlemlerini yerine getirebilmek şekilde tasarlanmıştır. “ZenomArduinoManager” kütüphanesi Zenom ile Arduino arasında alınıp, gönderilecek değişkenlerin kayıt edilmesi, verinin biçimlendirilmesi ve periyodik olarak verinin seri port üzerinden gönderilip alınmasını sağlayacak metotları içerir.

Arduino fiziksel platformu kısıtlı hesaplama gücüne sahiptir. Bu nedenle gelişmiş veri yapılarını içeren kütüphaneler içermemektedir. Zenom haberleşmesi için gerekli olacak veri yapıları “ZenomArduinoManager” içinde tanımlanmıştır.

4.3. ArduinoControlBase

Zenom benzetim ortamında çalıştırılacak uygulamanın “ControlBase” sınıfından türetilerek gerçeklenmesi gerekmektedir. Zenom projesine eklenen “ArduinoManager” kütüphanesinin Arduino ile birlikte çalışabilmesi için

“ControlBase” içinde gerekli metot çağrımlarını yapması gerekmektedir. Mevcut “ControlBase” sınıfı kullanılarak Arduino ile birlikte çalıştırılacak benzetimlerde kullanıcı “ArduinoManager” metotlarını belli bir sıra ve zamanda çağırmak zorunda kalmaktadır. Bu durum hem kullanıcının gerçeklediği kod miktarını artırmakta hem de kullanıcı kaynaklı oluşabilecek hataları artırmaktadır. Bu sorunun önüne geçmek için ArduinoControlBase sınıfı proje eklenmiştir.

ArduinoControlBase sınıfı “ControlBase” sınıfından türetilerek geliştirilmiştir. “ControlBase” ile geliştirilen benzetimler ile uyumluluğu korunmuştur. Bu sayede “ControlBase” ile geliştirilmiş bir benzetim, “ControlBase” yerine “ArduinoControlBase” sınıfından türetilerek derlenip, çalıştırılabilmektedir.

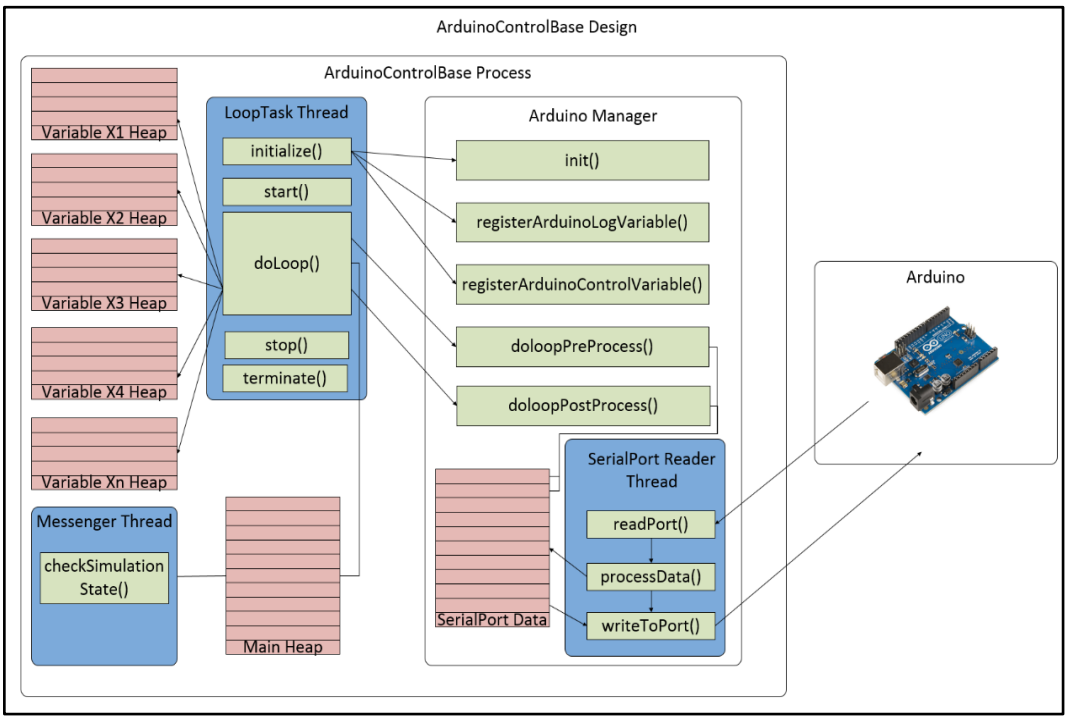
“ArduinoControlBase” sınıfı kullanıcının gerçeklediği metotları ve “ArduinoManager” kütüphanesini kullanarak Arduino ile haberleşmeyi sağlayan metotları çağırarak benzetim koşumunun ve Arduino haberleşmesinin eş zamanlı olarak çalıştırılmasını sağlar. “ArduinoControlBase” sınıfı kullanılarak, kullanıcının “ArduinoManager” kütüphanesi kullanma ihtiyacı giderilmiştir.

“ArduinoControlBase” sınıfına kullanıcının Arduino platformu ile haberleşmede kullanacağı değişkenleri tanımlamasına imkân sağlayan metotlar eklenmiştir. Kullanıcı bu metotlar ile Arduino platformuna göndermek istediği ya da Arduino platformundan almak istediği değişkenleri tanımlayabilmektedir. Arduino platformundan alınacak değişkenler “ArduinoLogVariable” olarak adlandırılmıştır. Kullanıcı ControlBaseArduino sınıfındaki registerArduinoLogVariable metodunu çağırarak değişkenini “ArduinoLogVariable” olarak tanımlar. “ArduinoLogVariable” olarak tanımlanan değişkenler her “doloop()” çağrımından önce Arduino platformundan gelen değer ile güncellenir. Kullanıcı “doloop()” metodu içinde “ArduinoLogVariable” olarak tanımladığı değişkenin değerine eriştiğinde, Arduino platformunun üzerinde hesaplanan değere erişmiş olur. Arduino platformuna gönderilecek değişkenler “ArduinoControlVariable” olarak tanımlanmıştır. Kullanıcı ControlBaseArduino sınıfındaki registerArduinoControlVariable metodunu çağırarak değişkenini “ArduinoControlVariable” olarak tanımlar. “ArduinoControlVariable” değişkenlerinin değerlerini her “doloop()” çağrımından sonra Arduino platformuna gönderilir. Kullanıcın “doloop()” içinde güncellediği değişken Arduino platformuna üzerinde çalışan uygulamaya aktarılır.

ArduinoLogVariable ve “ArduinoControlVariable” değişken tipleri, “LogVariable” ve “ControlVariable” değişkenler tiplerinden türetilmiştir. Bu sayede

tanımlanan ArduinoLogVariable, “ArduinoControlVariable” değişkenleri “ZenomGUI” üzerinden “LogVariable” ve “ControlVariable” görselleyen arayüzler ile takip edilebilir ya da güncellenebilir.

“ArduinoControlBase” sınıfı “ControlBase” sınıfında bulunan ve kullanıcının gerçeklemesi gereken beş metodu içerir. Kullanıcı Arduino platformu ile haberleşmek için kullanacağı değişkenleri tanımlamak ve kayıt etmek dışında “ControlBase” uygulaması gerçeklerken yapması gereken işlemleri yapması yeterlidir. “ArduinoControlBase” sınıfı kullanıcının “ArduinoManager” kütüphanesine ihtiyacını ortadan kaldırmış ve daha kolay bir kullanım imkânı sunmuştur. “ArduinoControlBase” tasarımın diyagramı Şekil 4.2 ile gösterilmiştir.



Şekil 4.2: “Zenom-Arduino” – “ArduinoControlBase” tasarım diyagram görüntüsü.

4.4. Zenom Target Project

Zenom benzetim ortamı ve Arduino fiziksel platformunu birlikte kullanarak benzetim geliştirme işlemlerini kolaylaştırmak için "Zenom Target Project" uygulaması geliştirilmiştir. "Zenom Target Project" uygulaması, kullanıcının gerçeklediği "ArduinoControlBase" uygulaması ile uyumlu değişken alışverişi yapabilecek ".ino" uzantılı Arduino platformu üzerinde çalışabilecek kod dosyasını oluşturur. "Zenom Target Project" uygulaması kullanıcının gerçeklediği dosya içindeki "ArduinoLogVariable" ve "ArduinoControlVariable" değişken tanımlamalarını bularak, bu değişkenlerin alınıp verildiği şablon bir ".ino" dosyası üretir. Oluşturulan bu dosyada kullanıcı Arduino platformu üzerinde çalışmasını istediği uygulamayı gerçekler. Oluşturulan şablon dosya içinde seri port alışverişi ile sorumlu çağrımlar Arduino platformunun bir parçalısı olan "loop()" metodu içinde çağrılmıştır. Ayrıca veri alışverişi için kullanılacak değişkenlerin tanımlanması ve kayıt edilmesi için gereken çağrımlar "setup()" metodu ve global seviyede yapılmıştır. Arduino platformu için geliştirilen "ZenomArduinoManager" çağrımları şablon içinde kullanılmıştır ve kullanıcının kendi gerçekleyeceği kod içinde "ZenomArduinoManager" kütüphanesini kullanmasına gerek kalmamıştır. Kullanıcının hata yapmasını önlemek için değiştirmemesi gereken kod bölümleri şablon dosya içinde belirtilmiştir.

"Zenom Target Project" uygulaması Arduino platformu için geliştirilen ".ino" uzantılı dosyayı derleme yeteneğine sahiptir. Uygulama derleme işlevi parametreleri ile çalıştırıldığında, ".ino" uzantılı dosyası derler ve Arduino platformu üzerinde çalışabilecek dosyayı oluşturur. Oluşturulan bu dosya "Zenom Target Project" uygulamasının yükleme yeteneği ile Arduino platformuna yüklenir. Bu işlemin yapılabilmesi için derleme işleminden sonra "Zenom Target Project" uygulaması yükleme parametresi ile çalıştırır. Bu işlemler esnasında hata oluşması durumunda kullanıcıya gerekli bilgiler gösterilir.

4.4.1. Zenom Uygulama Geliştirme

Zenom benzetim ortamı ile Arduino platformu için benzetim geliştirmek için öncelikle "Zenom Target Project" uygulaması ile proje oluşturulması gerekmektedir.

Kullanıcı oluşturmak istediği proje ismi ve Arduino projesi oluşturma parametresi ile proje oluşturma uygulamasını çalıştırır ve proje klasörü oluşur. Oluşan klasör içinde ControlBaseArduino sınıfından türetilmiş şablon benzetim kodunun bulunduğu "main.cpp" dosyası geliştirilmek istenilen benzetimin işlemlerini gerçekleyecek şekilde gerçeklenir.

Bu işlemleri açıklamak için "ArduinoEchoTester" isminde bir örnek geliştirilmiştir. Bu örneğin geliştirilme aşamaları aşağıda anlatılarak "Zenom-Arduino" üzerinde bir benzetimin geliştirilme ve çalıştırılması açıklanmıştır. "ArduinoEchoTester", bir "ArduinoLogVariable" ve bir "ArduinoControlVariable" içermektedir. Zenom uygulaması ile Arduino platformuna bir değişken gönderilmekte ve gönderilen değişken farklı bir değişken üzerinden Zenom uygulamasına geri gönderilmektedir. Proje oluşturma işlemi için yapılması işlemin görüntüsü Şekil 4.3 ile gösterilmiştir. Oluşturulan projenin sınıf tanımlama kodu Şekil 4.4, uygulama kodu Şekil 4.5 ile gösterilmiştir.

Şekil 4.3: "Zenom-Arduino" – Proje oluşturma işlemleri görüntüsü.

```
#include "controlbasearduino.h"
/**
* Zenom - Hard Real-Time Simulation Enviroment
* ArduinoEchoTester
*/
class ArduinoEchoTester : public ControlBaseArduino
{
public:
    /**
    * Initializes and registers the log variables and control variables.
    * @return
    */
    virtual int initialize();
    /**
    *
    * @return
    */
    virtual int start();
    /**
    * @return
    */
    virtual int doloop();
    /**
    * @return
    */
    virtual int stop();
    /**
    * @return
    */
    virtual int terminate();
private:
    // ----- Log Variables -----
    double echoTesterLogVariable;
    // ----- Control Variables -----
    double echoTesterControlVariable;
};
```

Şekil 4.4: “ArduinoEchoTester” projesi “ControlBase” sınıf tanımlama kodu.

```
int ArduinoEchoTester::initialize()
{
    registerArduinoLogVariable(&echoTesterLogVariable, "echoTesterLogVariable");
    registerArduinoControlVariable(&echoTesterControlVariable, "echoTesterControlVariable");

    return 0;
}

int ArduinoEchoTester::start()
{
    return 0;
}

int ArduinoEchoTester::doloop()
{
    return 0;
}

int ArduinoEchoTester::stop()
{
    return 0;
}

int ArduinoEchoTester::terminate()
{
    return 0;
}

int main(int argc, char *argv[])
{
    ArduinoEchoTester c;
    c.run(argc, argv);

    return 0;
}
```

Şekil 4.5: "ArduinoEchoTester" projesi "ControlBase" uygulama kodu.

4.4.2. Arduino Uygulama Geliştirme

"ArduinoControlBase" uygulaması gerçeklendikten sonra benzetimin Arduino platformu üzerinde çalışacak uygulama için ".ino" uzantılı dosya gerçeklenmelidir. Arduino üzerinde çalışacak uygulamanın Zenom ile veri alışverişi yapabilmesi için aynı biçimde olan verileri kullanmaları gerekmektedir. Bu nedenle "ZenomArduinoManager" kütüphanesindeki metot çağrımları, haberleşme için kullanılacak değişken tanımlamaları aynı olmadır. Bu sağlamak için "Zenom Target Project" uygulaması geliştirilmiştir. Kullanıcı "ArduinoControlBase" sınıfını gerçeklediği "main.cpp" dosyasını "Zenom Target Project" uygulamasına parametre olarak geçerek çalıştırır ve uygulama çıktısı olarak ".ino" uzantılı dosya oluşturur. Oluşan dosyada Zenom ile Arduino arasındaki haberleşme için gereken çağrımlar ve değişkenler hazır olarak bulunur. Bu sayede kullanıcı sadece Arduino platformu üzerinde çalışacak uygulama kısmını gerçeklemek zorundadır.

"Zenom-Arduino" kullanımını göstermek için oluşturulan "ArduinoEchoTester" uygulamasının Arduino platformu için hazırlanması gereken kısımları aşağıda anlatılmıştır. Öncelikle "Arduino Project" uygulaması ile ".ino" uzantılı dosya oluşturur ve bu dosya gerçeklenir. Arduino üzerinde çalışacak uygulamanın kod şablonunu oluşturmak için yapılması gereken işlem Şekil 4.6 ile gösterilmiştir. Oluşturulan şablon dosyanın gerçeklenmiş hali Şekil 4.7 ile gösterilmiştir.

```
C:\usr\src\zenom\bin\ArduinoEchoTester>znm-target-project -generateino main.cpp
```

Şekil 4.6: "ArduinoEchoTester" projesi Arduino proje oluşturma işlemleri görüntüsü.

```
/* Arduino Project Code Begin - Do not change this block */
ZenomArduinoManager zenomManager;
double echoTesterLogVariable;
double echoTesterControlVariable;
/* Arduino Project Code End */

void setup()
{
    /* Arduino Project Code Begin - Do not change this block */
    Serial.begin(9600);
    zenomManager.registerLogVariable(&echoTesterLogVariable, "echoTesterLogVariable");
    zenomManager.registerControlVariable(&echoTesterControlVariable, "echoTesterControlVariable");
    /* Arduino Project Code End */

    // User Code
}
void loop()
{
    /* Arduino Project Code Begin - Do not change this block */
    zenomManager.loopPreProcess();
    /* Arduino Project Code End */

    // User Code Begin
    echoTesterLogVariable = echoTesterControlVariable;
    // User Code End

    /* Arduino Project Code Begin - Do not change this block */
    zenomManager.loopPostProcess();
    delay(500);
    /* Arduino Project Code End */
}
```

Şekil 4.7: "ArduinoEchoTester" projesi Arduino kodu.

Arduino Project uygulaması ile oluşturulan ".ino" dosyası gerçeklendikten sonra "Zenom Target Project" Project uygulaması ile derlenir. Derleme işlemi başarılı ise "Zenom Target Project" uygulaması yükleme parametresi ile çalıştırır ve uygulama Arduino platformuna yüklenir. Uygulama bu işlemden sonra Arduino platformu üzerinde koşmaktadır. Arduino ile Zenom uygulamasını birlikte kullanmak için

“ZenomGUI” çalıştırır. Oluşturulan “ArduinoEchoTester” uygulaması “ZenomGUI” proje yükleme arayüzü ile yüklenir ve koşum başlatma butona tıklanır.

5. ÖRNEK UYGULAMA

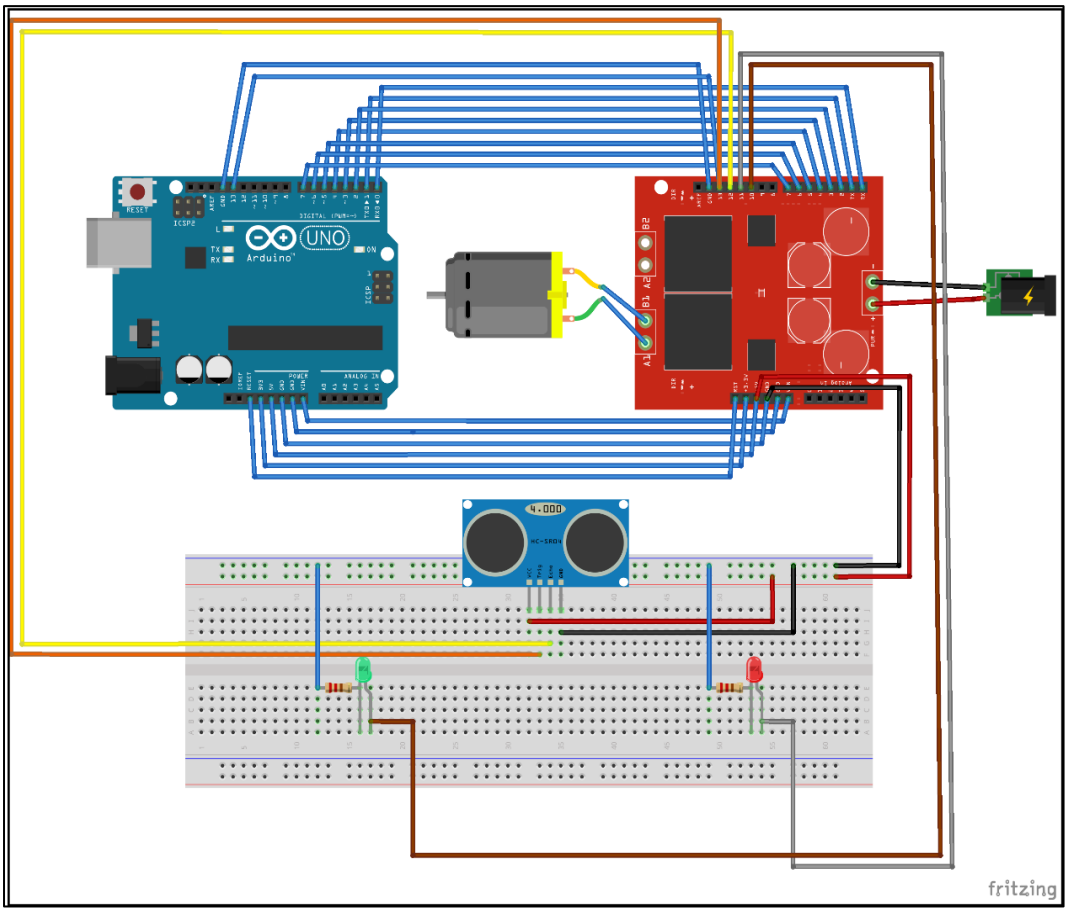
"Zenom-Arduino" entegrasyonunu göstermek için Arduino ile kontrol edilen bir uygulama düzeneği oluşturduk. Oluşturduğumuz düzenek Zenom benzetimin ortamının yeteneklerinin ve Arduino fiziksel platformunun birlikte kullanımını test etmek ve Zenom benzetim ortamının donanım platformları ile birlikte kullanıldığında sistemlere kazandırdığı yetenekleri göstermek için tasarlandı.

Uygulama düzeneği Arduino fiziksel platformu, ultrasonik mesafe ölçer ve elektrik motoru ile tek eksende hareket edebilen bir platformdan oluşmaktadır. Uygulama düzeneği hareketli bir engele sürekli eşit mesafede kalmaya çalışacaktır. Bu işlemi gerçekleştirirken mesafe bilgisini ultrasonik mesafe ölçerden alacak ve gerekli hesaplamaları yaparak elektrik motorunu çalıştırıp platformu hareket ettirecektir. Düzenek için geliştirilecek "Zenom-Arduino" uygulaması iki farklı şekilde çalıştırılmıştır.

Uygulamanın birinci çalışma şeklinde, mesafe bilgilerini alıp Arduino üzerinde hesaplamaları yaparak, elektrik motorunu çalıştıracaktır. Bu çalışma şeklinde Zenom uygulaması ile Arduino üzerinde hesaplanan değerler izlenecek ve platformun koruması istenilen mesafe değeri değiştirilebilmektedir.

Uygulamanın ikinci çalışma şeklinde, mesafe bilgileri Arduino cihazından Zenom uygulamasına gönderilecek, Zenom uygulaması gerekli hesaplamaları yaparak, motorun çalışma süresi ve yönünü belirleyecektir. Belirlenen değerler Arduino üzerinden motora aktarılacak ve platformun engele olan mesafesi korunacaktır.

Birinci çalışma şeklinde Zenom sadece Arduino tarafından ölçülen ve üretilen değerleri görselleyen bir yapıyı canlandırmaktadır, ikinci çalışma şeklinde ise Zenom, Arduino platformunun hesaplama gücünün yetersiz olduğu durumlarda hesaplamaların bilgisayar ile yapılıp sonuçların Arduino cihazına aktarıldığı bir yapı canlandırmaktadır. Geliştirilen örnek uygulamanın devre şeması Şekil 5.1 ile gösterilmiştir.



Şekil 5.1: Örnek Uygulama – Örnek uygulama devre şeması.

Geliştirilen örnek uygulama için Arduino ile kontrol edilen bir devre hazırlanmıştır. Hazırlanan bu devrede bir adet HC-SR04 modeli ultrasonik mesafe ölçer, "VNH2SP3" motor sürücü köprüsü kullanan "Hummer Driver" Arduino motor kontrolcü devresi ve "Pittman GM9236S019" modeli 12V DC motor kullanılmıştır. Oluşturulan devre ile ultrasonik mesafe ölçerden gelen veriler mesafe tespiti yapılabilir ve 12V DC motor iki yöne istenilen hızlarda çalıştırılabilir.

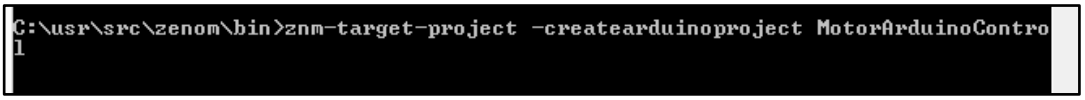
12V DC motorun çalıştırılması için "ZenomArduinoManager" kütüphanesine motor kontrolü için gerekli olan bileşenler eklenmiştir. Bu bileşenler ile Arduino uyumlu motor kontrolcüleri kolay bir şekilde benzetim içinde kullanılabilmektedir.

5.1. Zenom Gözlemci – Arduino Hesaplayıcı Uygulama

Örnek uygulamanın Zenom gözlemci, Arduino hesaplayıcı çalışma şeklinde "Zenom-Arduino" entegrasyonunun, benzetimin çalıştığı fiziksel platform üzerindeki verileri izleme ve benzetime müdahale etme yeteneklerini göstermek için tasarlandı. Örnek uygulamanın Zenom kısmında hiçbir hesaplama ya da işlem yapılmamıştır, sadece üretilen sonuçları görselleyecek ve benzetim için belirlenen "ControlVariable" değişkenlere müdahale etme imkânı sunulmuştur. Arduino kısmında ise mesafe sensöründen alındandan mesafe bilgisine göre motorun çalışma yönüne ve hızına karar verilmiştir. Karar verme işleminden sonra motor çalıştırılarak hareketli platformun engele olan mesafesi korunmaya çalışmıştır.

Örnek uygulamanın gerçeklenmesi için Zenom tarafında çalışacak ControlBaseArduino ve Arduino tarafında çalışacak uygulamaların gerçeklenmesi gerekmektedir. Bu işlemlerin yapılabilmesi için "Zenom Target Project" uygulaması kullanılarak proje klasörü oluşturur ve gerekli dosyalar gerçeklenir.

Örnek uygulamanın bu sürümü "MotorArduinoControl" olarak adlandırılmıştır. Örnek uygulamanın geliştirilmesi için oluşturulacak proje için proje üretme uygulaması proje ismi çalıştırılmalıdır. Proje oluşturma işlemi Şekil 5.2 gösterilmiştir.



Şekil 5.2: "MotorArduinoControl" proje oluşturma işlemleri görüntüsü.

Yukarıda belirtilen işlem çalıştırıldığında "MotorArduinoControl" isimli bir klasör oluşur ve içinde gerçeklenmesi gereken "ControlBaseArduino" dosyaları bulunur. Uygulamanın Zenom tarafında çalışacak kısmı için "main.cpp" gerçeklenir. "MotorArduinoControl" örneğinin sınıf tanımlama kodu Şekil 5.3, sınıf gerçekleme kodu Şekil 5.4 ile gösterilmiştir.

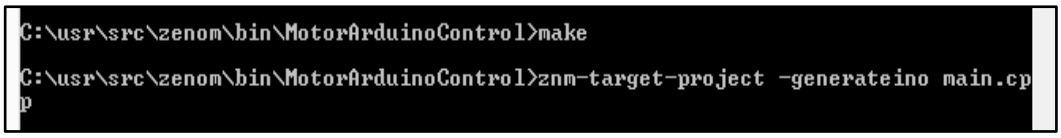
```
#include "controlbasearduino.h"

/**
* Zenom - Hard Real-Time Simulation Enviroment
* @author
*
* MotorArduinoControl
* -- Description --
*
*/
class MotorArduinoControl : public ControlBaseArduino
{
public:

    /**
    * Initializes and registers the log variables and control variables.
    * @return
    */
    virtual int initialize();

    /**
    *
    * @return
    */
    virtual int start();

    /**
    *
    * @return
    */
    virtual int doloop();

    /**
    *
    * @return
    */
    virtual int stop();

    /**
    *
    * @return
    */
    virtual int terminate();

private:
    // ----- Log Variables -----
    double Distance;
    double Error;


    // ----- Control Variables -----
    double CriticalProximity;
    double CriticalDistance;

};
```

Şekil 5.3: “MotorArduinoControl” projesi “main.cpp” sınıf tanımlama kodu.

```
int MotorArduinoControl::initialize()
{
    CriticalProximity = 15;
    CriticalDistance = 35;

    registerLogVariable(&Error, "Error");
    registerArduinoLogVariable(&Distance, "Distance");

    registerArduinoControlVariable(&CriticalProximity, "CriticalProximity");
    registerArduinoControlVariable(&CriticalDistance, "CriticalDistance");

    return 0;
}

int MotorArduinoControl::start()
{

    return 0;
}

int MotorArduinoControl::doloop()
{
    if (Distance < CriticalProximity)
    {
        Error = CriticalProximity - Distance;
    }
    else if (Distance > CriticalDistance)
    {
        Error = Distance - CriticalDistance;
    }
    else
    {
        Error = 0;
    }

    return 0;
}

int MotorArduinoControl::stop()
{
    return 0;
}

int MotorArduinoControl::terminate()
{
    return 0;
}


int main(int argc, char *argv[])
{
    MotorArduinoControl c;
    c.run(argc, argv);
```

Şekil 5.4: “MotorArduinoControl” projesi “main.cpp” sınıf gerçekleme kodu.

Zenom tarafında çalışacak uygulama belirtilen şekilde gerçeklenmiştir. Gerçeklenen kısımda mesafe kontrolü için gerekli olan hiçbir işlem yapılmamış, sadece Arduino platformundan gelen mesafe bilgisine göre oluşan hata miktarı hesaplanmıştır. Gerçeklenen dosyanın derlenerek Zenom uygulaması oluşturulur ve derleme başarılı ise gerçeklenen koddan Arduino platformu için gerçeklenmesi gereken dosyayı oluşturacak uygulama çalıştırılır. Derleme ve şablon kod üretme işlemleri Şekil 5.5, oluşturulan şablon kod dosyasının değişken tanımlama ve ilkleme işlemleri Şekil 5.6 ile gösterilmiştir.



Şekil 5.5: "MotorArduinoControl" proje derleme ve şablon kod üretme işlemleri görüntüsü.

```
#define USE_MOTOR
#include "zenomarduinomanager.h"

ZenomArduinoManager zenomManager;

#define echoPin 12
#define led 11
#define led2 10
#define statpin 13

// Log Variables
double Distance;

// Control Variables
double CriticalProximity = 15;
double CriticalDistance = 35;

void setup()
{
    // Log Variable Register
    zenomManager.registerLogVariable(&Distance, "Distance");

    // Control Variable Register
    zenomManager.registerControlVariable(&CriticalProximity, "CriticalProximity");
    zenomManager.registerControlVariable(&CriticalDistance, "CriticalDistance");

    Serial.begin(9600);

    // User Code
    pinMode(statpin, OUTPUT);
    pinMode(echoPin, INPUT);
    pinMode(led, OUTPUT);
    pinMode(led2, OUTPUT);

    CriticalProximity = 15;
    CriticalDistance = 35;
}
```

Şekil 5.6: “MotorArduinoControl” projesi “MotorArduinoControl.ino” değişken tanımlama ve ilkleme kodu.

Yukarıda işlemler çalıştıklarında Zenom tarafında çalışacak uygulama ve proje klasörü içindeki “src” klasöründe “MotorArduinoControl.ino” dosyası oluşur. Bu dosya Arduino platformu tarafından yapılması gereken işlemleri içerecek şekilde gerçeklenir. “MotorArduinoControl.ino” dosyasının gerçeklenmiş hali Şekil 5.7 ile gösterilmiştir.

```
void loop()
{
    zenomManager.loopPreProcess();
    // User Code Begin
    long duration;

    digitalWrite(statpin, LOW);  // Adde
    delayMicroseconds(2); // Added this line
    digitalWrite(statpin, HIGH);
    delayMicroseconds(10); // Added this line
    digitalWrite(statpin, LOW);
    duration = pulseIn(echoPin, HIGH);
    Distance = (duration / 2) / 29.1;
    if (Distance > 200)
    {
        Distance = 200;
    }
    if (Distance < CriticalProximity)
    {
        motorGo(0, CW, 1023);
        motorGo(1, CW, 1023);
        digitalWrite(led, HIGH);
        digitalWrite(led2, LOW);
    }
    else if (Distance > CriticalDistance)
    {
        motorGo(0, CCW, 1023);
        motorGo(1, CCW, 1023);
        digitalWrite(led, LOW);
        digitalWrite(led2, HIGH);
    }
    else
    {
        motorOff(0);
        motorOff(1);
        digitalWrite(led, LOW);
        digitalWrite(led2, LOW);
    }

    // User Code End
    zenomManager.loopPostProcess();
    //delay(100); // delay time editable
}
```

Şekil 5.7: Örnek Uygulama – “MotorArduinoControl” “MotorArduinoControl.ino” çalışan algoritma kodu.

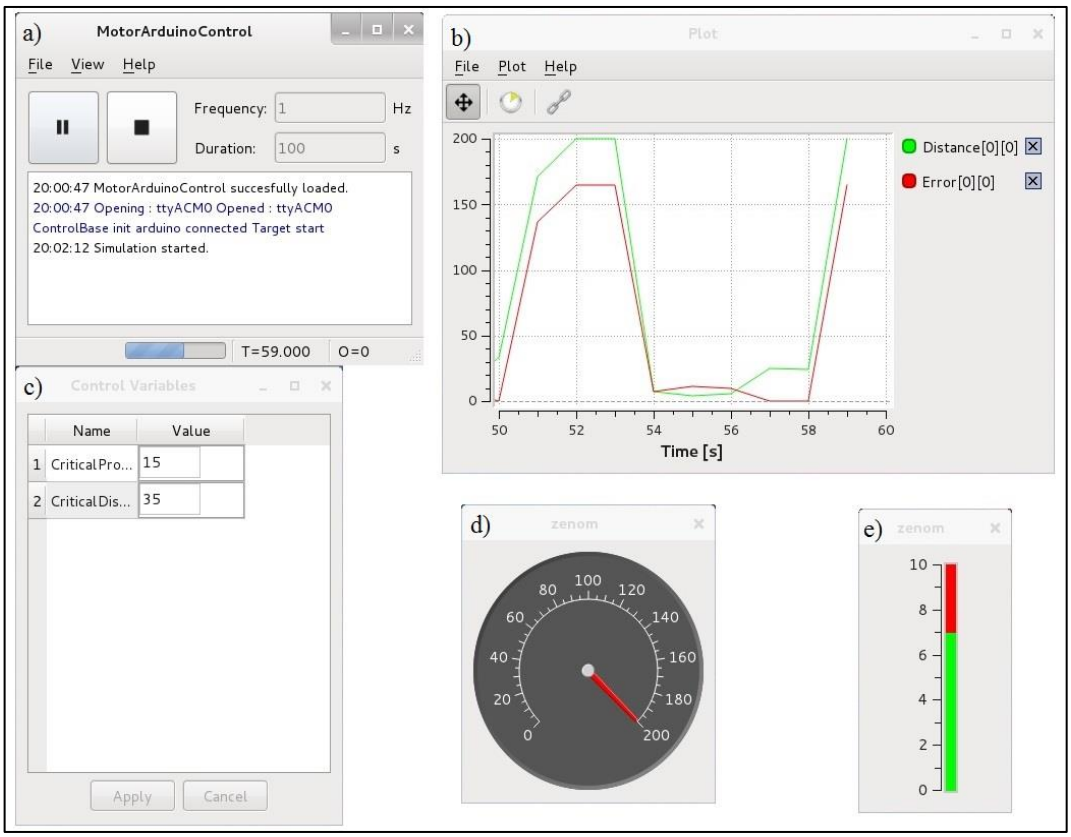
Arduino tarafında çalışacak işlemlerin gerçeklendiği “MotorArduinoControl.ino” dosyası derlenmeli ve Arduino platformuna yüklenmelidir. Bu işlemleri gerçekleştirmek için “Zenom Target Project” uygulaması

kullanılır. Şekil 5.8 ile gösterilen işlemler sırasıyla çalıştırılır ve örnek uygulama çalışmaya hazır hale getirir.

```
C:\usr\src\zenom\bin\MotorArduinoControl>znm-target-project -buildarduinoproject
znm-target-project -uploadarduinoproject
```

Şekil 5.8: Örnek Uygulama – Arduino uygulama derleme ve yükleme işlemleri görüntüsü.

Arduino platformuna uygulama yükleme işlemi tamamlandıktan sonra örnek uygulama çalıştırılmaya hazırdır. "ZenomGUI" uygulaması çalıştırılır ve proje yükleme menüsünde "MotorArduinoControl.znm" uygulaması seçilir. Çalışan örnek uygulama Şekil 5.9 ile gösterilmiştir.



Şekil 5.9: : "MotorArduinoControl" – a) Ana pencere görüntüsü, b) Grafik pencere görüntüsü, c) Kontrol değişkenleri pencere görüntüsü, d) Mesafe değeri pencere görüntüsü, e) Hata değeri pencere görüntüsü.

5.2. Zenom Hesaplayıcı – Arduino Gözlemci Uygulama

Örnek uygulamanın Arduino gözlemci sürümü, motor kontrol kararlarının Zenom tarafında çalışan ControlBaseArduino uygulaması tarafından verilerek, verilen kararların Arduino platformuna aktarıldığı bir yapıdadır. Arduino fiziksel platformu mesafe ölçerden aldığı bilgiyi Zenom uygulamasına gönderir. Zenom uygulaması kullanıcının belirlediği kritik mesafe eşik değerlerine göre motorun çalışmasına ve yönüne karar verir. Verilen kararlar Arduino platformuna gönderilir ve Arduino motoru çalıştırarak engel ile olan mesafeyi korumaya çalışır.

Örnek uygulamanın gerçeklenmesi için Zenom tarafında çalışacak ControlBaseArduino ve Arduino tarafında çalışacak uygulamaların gerçeklenmesi gerekmektedir. Bu işlemlerin yapılabilmesi için "Zenom Target Project" uygulaması kullanılarak proje klasörü oluşturur ve gerekli dosyalar gerçeklenir. Bu işlemler "MotorArduinoControl" örneğinin açıklamasında detaylı şekilde anlatıldığı için bu uygulama açıklamasında daha kısa anlatılacaktır.

Örnek uygulamanın bu versiyonu "MotorZenomControl" olarak adlandırılmıştır. Örnek uygulamanın geliştirilmesi için oluşturulacak proje için proje üretme uygulaması proje ismi çalıştırılmalıdır. Proje oluşturma işlemi Şekil 5.10 gösterilmiştir.

```
C:\usr\src\zenom\bin>znm-target-project -createarduinoproject MotorZenomControl
```

Şekil 5.10: "MotorZenomControl" proje oluşturma işlemleri görüntüsü.

Oluşan "MotorZenomControl" klasöründeki "main.cpp" dosyası motorun çalışma kararını verecek şekilde gerçeklenir. "MotorZenomControl" uygulamasının sınıf tanımlama kodu Şekil 5.11, sınıf gerçekleme kodu Şekil 5.12 ile gösterilmiştir.

```
/**
* Zenom - Hard Real-Time Simulation Enviroment
* @author
*
* MotorZenomControl
* -- Description --
*
*/
class MotorZenomControl : public ControlBaseArduino
{
public:

    /**
    * Initializes and registers the log variables and control variables.
    * @return
    */
    virtual int initialize();

    /**
    *
    * @return
    */
    virtual int start();

    /**
    *
    * @return
    */
    virtual int doloop();

    /**
    *
    * @return
    */
    virtual int stop();

    /**
    *
    * @return
    */
    virtual int terminate();

private:
    // ----- Log Variables -----
    double Distance;
    double Error;

    // ----- Control Variables -----
    double MotorDirection;
    double CriticalProximity;
    double CriticalDistance;

};
```

Şekil 5.11: “MotorZenomControl” projesi “main.cpp” sınıf tanımlama kodu.

```
int MotorZenomControl::initialize()
{
    MotorDirection = 0;
    CriticalProximity = 15;
    CriticalDistance = 35;
    registerLogVariable(&Error, "Error");
    registerArduinoLogVariable(&Distance, "Distance");
    registerArduinoControlVariable(&MotorDirection, "MotorDirection", LOG_VARIABLE);
    registerControlVariable(&CriticalProximity, "CriticalProximity");
    registerControlVariable(&CriticalDistance, "CriticalDistance");

    return 0;
}
int MotorZenomControl::start()
{

    return 0;
}

int MotorZenomControl::doloop()
{
    if (Distance < CriticalProximity)
    {
        MotorDirection = 1;
        Error = CriticalProximity - Distance;
    }
    else if (Distance > CriticalDistance)
    {
        MotorDirection = 2;
        Error = Distance - CriticalDistance;
    }
    else
    {
        MotorDirection = 0;
        Error = 0;
    }
    return 0;
}
int MotorZenomControl::stop()
{
    return 0;
}
int MotorZenomControl::terminate()
{
    return 0;
}
int main(int argc, char *argv[])
{
    MotorZenomControl c;
    c.run(argc, argv);

    return 0;
}
```

Şekil 5.12: “MotorZenomControl” projesi “main.cpp” sınıf gerçekleme kodu.

Zenom tarafında çalışacak uygulama belirtilen şekilde gerçeklenmiştir. Zenom tarafında çalışacak uygulamanın derlenmesi ve Arduino platformunda çalışacak uygulamanın gerçeklenmesi için gerekli olan dosyanın oluşması için gereken işlemler Şekil 5.13 gösterilmiştir.

```
C:\usr\src\zenom\bin\MotorZenomControl>make
C:\usr\src\zenom\bin\MotorZenomControl>znm-target-project -generateino main.cpp
```

Şekil 5.13: "MotorZenomControl" proje derleme ve şablon kod üretme işlemleri görüntüsü.

Yukarıda işlemler çalıştıklarında Zenom tarafında çalışacak uygulama ve proje klasörü içindeki "src" klasöründe "MotorZenomControl.ino" dosyası oluşur. Bu dosya Arduino platformu tarafından yapılması gereken işlemleri içerek şekilde gerçeklenir. "MotorZenomControl.ino" dosyasının değişlen tanımlama ve ilkleme işlemler Şekil 5.14, gerçekleme kodu Şekil 5.15 ile gösterilmiştir.

```
#include "controlbasearduino.h"

#define USE_MOTOR
#include "zenomarduinomanager.h"
ZenomArduinoManager zenomManager;
#define echoPin 12
#define led 11
#define led2 10
// Log Variables
double Distance;
// Control Variables
double MotorDirection = 0;
void setup()
{
    // Log Variable Register
    zenomManager.registerLogVariable(&Distance, "Distance");
    // Control Variable Register
    zenomManager.registerControlVariable(&MotorDirection, MotorDirection");
        Serial.begin(9600);
    // User Code
    initMotor();
    pinMode(statpin, OUTPUT);
    pinMode(echoPin, INPUT);
    pinMode(led, OUTPUT);
    pinMode(led2, OUTPUT);
}
```

Şekil 5.14: "MotorZenomControl" projesi "MotorZenomControl.ino" değişken tanımlama ve ilkleme kodu.

```
void loop()
{
    zenomManager.loopPreProcess();

    // User Code Begin
    long duration;

    digitalWrite(statpin, LOW);  // Adde
    delayMicroseconds(2); // Added this line
    digitalWrite(statpin, HIGH);
    delayMicroseconds(10); // Added this line
    digitalWrite(statpin, LOW);
    duration = pulseIn(echoPin, HIGH);
    Distance = (duration / 2) / 29.1;
    if (Distance > 200)
    {
        Distance = 200;
    }

    if ((int)MotorDirection == 0)
    {
        motorOff(0);
        motorOff(1);
    }
    else if ((int)MotorDirection == 1)
    {
        motorGo(0, CW, 1023);
        motorGo(1, CW, 1023);
    }
    else if ((int)MotorDirection == 2)
    {
        motorGo(0, CCW, 1023);
        motorGo(1, CCW, 1023);
    }

    // User Code End

    zenomManager.loopPostProcess();
    //delay(500); // delay time editable
}
```

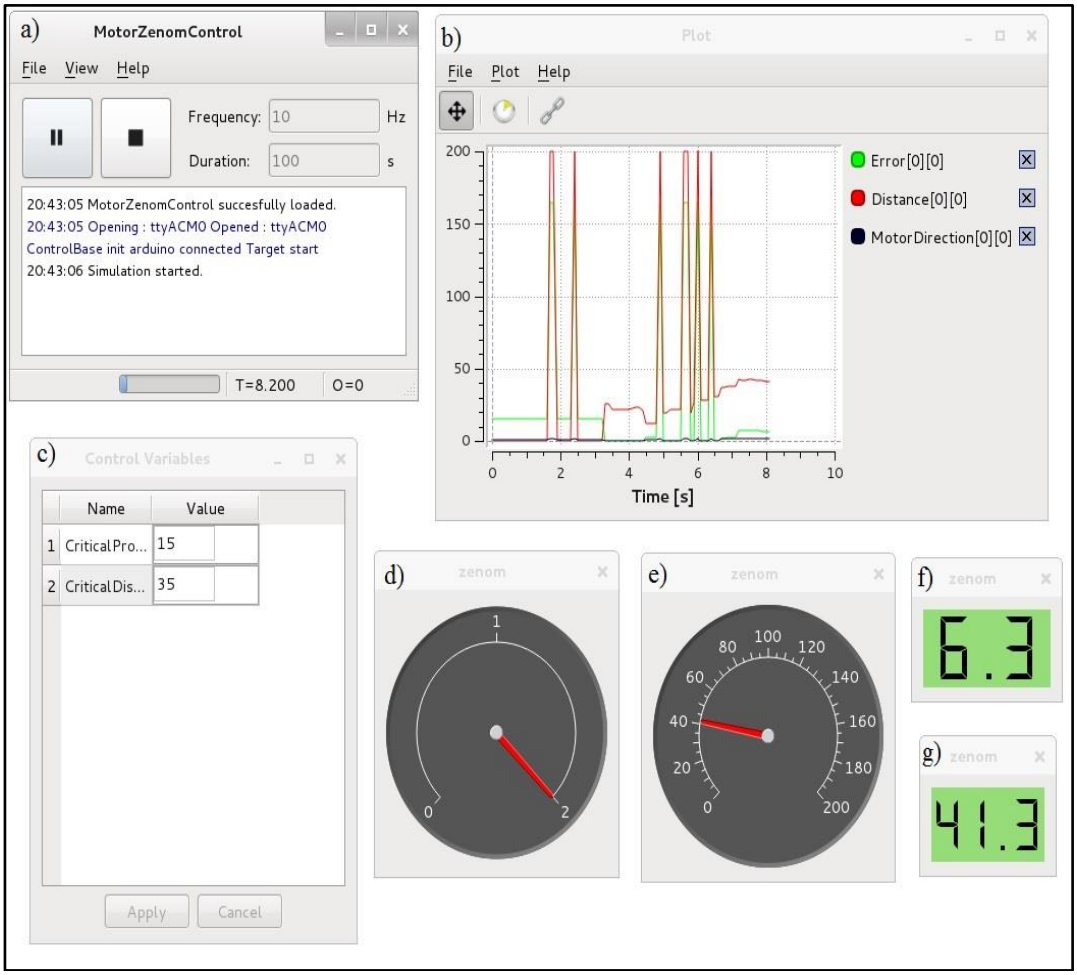
Şekil 5.15: "MotorZenomControl" projesi "MotorZenomControl.ino" çalışan algoritma kodu.

Arduino tarafında çalışacak işlemlerin gerçeklendiği "MotorZenomControl.ino" dosyası derlenmeli ve Arduino platformuna yüklenmelidir. Bu işlemleri gerçekleştirmek için "Zenom Target Project" uygulaması kullanılır. Şekil 5.16 ile gösterilen işlemler sırasıyla çalıştırılır ve örnek uygulama çalışmaya hazır hale getirir.

```
C:\usr\src\zenom\bin\MotorZenomControl>znm-target-project -buildarduinoproject
znm-target-project -uploadarduinoproject
```

Şekil 5.16: "Zenom-Arduino" – Arduino uygulama derleme ve yükleme işlemleri görüntüsü.

Arduino platformuna uygulama yükleme işlemi tamamlandıktan sonra uygulama çalıştırılmaya hazırdır. "ZenomGUI" uygulaması çalıştırılır ve proje yükleme menüsünde "MotorZenomControl.znm" uygulaması seçilir. Çalışan örnek uygulama Şekil 5.17 ile gösterilmiştir.



Şekil 5.17: "MotorZenomControl" – a) Ana pencere görüntüsü, b) Grafik pencere görüntüsü, c) Kontrol değişkenleri pencere görüntüsü, d) Motor yönü değer pencere görüntüsü, e) Mesafe değeri pencere görüntüsü, e) Mesafe değeri pencere görüntüsü, g) Hata değeri pencere görüntüsü.

6. SONUÇ VE YORUMLAR

Bu çalışmada Xenomai yaması ile çekirdeği yamalanmış Linux işletim sistemi ile çalışan standart bir bilgisayar üzerinde gerçek zamanlı benzetimlerin çalıştırılmasını sağlayan bir benzetim ortamı geliştirilmiştir. Geliştirilen bu benzetim ortamı, kullanıcının benzetim koşumu sırasında benzetim sonuçlarını gelişmiş arayüzler ile takip edebilmesine ve benzetim değişkenleri üzerinde değişiklik yaparak benzetim sonuçlarına müdahale edebilmesine imkân sağlamaktadır.

Zenom benzetim ortamı yeni geliştirilen Zenom-Qt sürümü ile eski sürümde karşılaşın sorunlar giderilmiş ve benzetim geliştirme ve çalıştırma işlemleri için kullanılan araçlar yenilenmiştir. Geliştirilen benzetim geliştirme ortamı, mevcut benzetim geliştirme ortamları incelenerek geliştirilmiş ve mevcut geliştirme ortamlarının olumlu yanlarını içermektedir. 3 boyutlu görselleme ve C++ dilinde benzetim geliştirme yeteneklerinin ücretsiz olarak sunulması ve gerçek zamanlı, açık kaynak kodlu yapısı Zenom benzetim ortamını diğer benzetim ortamlarından ayırmaktadır. Mevcut benzetim ortamları genel olarak “Simulink” ile benzetim geliştirmeye imkân sağlamaktadır, Zenom hem “Simulink” hem de C++ ile benzetim geliştirmeye imkân sağladığı için temel seviyede programlama bilgisi ile de benzetim geliştirilmesini sağlamıştır. Benzetim geliştirme için sunulan yardımcı araçlar ile geliştirme süreci hızlandırılmış ve kolaylaştırılmıştır. İncelenen benzetim ortamları ile Zenom benzetim ortamının özelliklerinin karşılaştırılması Şekil 6.1 ile gösterilmiştir.

Şekil 6.1: Sonuçlar ve Yorumlar – Benzetim ortamları özellik karşılaştırma tablosu.

Ayrıca Zenom projesinin açık kaynak kodlu bir proje olması nedeniyle, uygulamanın kodu da kullanıcı dostu olacak şekilde tasarlanmıştır. Kullanıcının ihtiyaç duyacağı yeni özellikleri ve yeni sistemleri Zenom içinde kullanması için yapması gereken geliştirme en aza indirilmiştir. Tasarımdaki modüler yapı sayesinde projenin üzerine eklemeler yapmak oldukça kolaydır.

Zenom benzetim ortamına yeni özellik ve sistemlerin eklenmesi ve gerçek zamanlı benzetimlerin ihtiyacı olan fiziksel bir sistemin Zenom ile kullanılması eklenen yeni bileşenler mümkündür. Yeni bileşen eklemenin anlatılması ve Zenom benzetim ortamının fiziksel sistemler ile kullanılmasının örneklenmesi için Zenom ile Arduino fiziksel platformunu birlikte kullanmayı sağlayan bileşenler geliştirilmiş ve örnek bir uygulama ile kullanımı gösterilmiştir.

Geliştirilen örnek uygulama Zenom ile fiziksel bir platformun birlikte farklı şekillerde kullanılabileceğini göstermek için iki farklı şekilde tasarlanmıştır. Tasarlanan birinci uygulama fiziksel platformdan gelen verilerin bilgisayar ortamında takip edilmesini sağlamıştır. İkinci tasarımda ise fiziksel platform ile çevreden alınan veriler bilgisayar tarafında işlenerek üretilen sonuçlar fiziksel

platforma aktarılmıştır. Bu örnek uygulama ile fiziksel platform ile bilgisayarın işlem gücü birlikte kullanılmıştır. Bu sayede düşük maliyetli fiziksel sistemlerin yetersiz olan işlem gücü, ek maliyet olmadan artırılabilmektedir.

Geliştirilen örnek uygulamanın çalıştırılması için engelin elektrik motoru ile kontrol edildiği bir düzenek kurulmuştur. Engel elektrik motoru ile kontrollü bir şekilde engele yakınlaştırılmış ve uzaklaştırılmıştır. Uygulamanın iki çalışma şeklinde de engelin davranışının aynı olması sağlanmıştır.

Örnek uygulama düzeneğinde engel ile hareketli platform arasında korunması istenilen mesafelerin işaretli olduğu bir cetvel yerleştirilmiştir. Hareketli platform ile cetvel 60 kare/saniye çekim hızında kaydeden bir kamera kullanılarak video kaydedilmiştir. Uygulamalar kameranın çekim hızından daha yüksek bir frekansta çalıştırılmıştır. Kaydedilen videoların 30 saniyelik bölümleri karşılaştırma için incelenmiştir. Karşılaştırma için motorun çalıştığı kare sayısı, engel ile mesafenin korunamadığı kare sayısı ve motorun çalışması gerekirken çalışmadığı ardışık kare sayıları hesaplanmıştır. Hesaplanan sonuçlar Şekil 6.2 ile gösterilmiştir.

	MotorArduinoControl	MotorZenomControl
Toplam Kare Sayısı	1800	1800
Motorun Çalıştığı Kare Sayısı	386	438
Engel ile mesafenin korunmadığı kare sayısı	415	473
Motorun çalışması gerekirken çalışmadığı kare sayısı (Gecikmenin olduğu en yüksek ardışık kare sayısı)	4	22

Şekil 6.2: Sonuçlar ve Yorumlar – "MotorArduinoControl" uygulaması ve "MotorZenomControl" uygulaması video analiz sonuçları.

Video analizi sonucuna göre "MotorZenomControl" uygulamasında mesafe bilgisine göre motorun hareketinde ~350 ms gecikme olmuştur. Bu değer "MotorArduinoControl" uygulamasında bu değer ~60 ms olmuştur. Bu gecikme seri port haberleşmesinden kaynaklanmaktadır. Ayrıca "MotorZenomControl"

uygulamasında, engel ile olan mesafe korunması gereken değerler arasında bir değere ulaştığında motor hareketinin geç sonlandığı görülmüştür. Bu da motorun çalıştığı kare sayısının "MotorArduinoControl" uygulamasına göre daha fazla olmasına neden olmuştur.

Bu sonuca göre hesaplama maliyeti, iletişim gecikmesinden fazla olan işlemlerin Zenom tarafında, geriye kalan işlemlerin fiziksel platform ile yapılması sistemin performansını artıracaktır.

Bu çalışmada Zenom benzetim ortamı ve "Zenom-Arduino" birlikte çalışması anlatılmıştır. Bu çalışma ile Zenom benzetim ortamının fiziksel platformlar ile birlikte çalışma yetenekleri gösterilmiştir. Yapılan bu çalışma ile yeni platformların Zenom benzetim ortamına eklenmesi kolaylaştırılmıştır. Popüler platformlar bu çalışmadan faydalanılarak az bir maliyet ile Zenom ile birlikte çalışabilir hale getirebilir.

KAYNAKLAR

Al-Busaidi A. M., (2012), "Development of an educational environment for online control of a biped robot using MATLAB and Arduino", 13th International Workshop Mechatronics (MECATRONICS), 9th France-Japan & 7th Europe-Asia Congress on and Research and Education in Mechatronics (REM), 337 - 344, Sump ca Paris, Paris, France, 21-23 November.

Barbalace A., Luchetta A., Manduchi G., Moro M., Soppelsa A., Taliercio C., (2007), "Performance Comparison of VxWorks, Linux, RTAI and Xenomai in a Hard Real-time Application", 15th Real-Time Conference IEEE-NPSS, 1-5, Batavia, IL, USA, 29 April - 4 May.

Barrett S., (2013), "Arduino Microcontroller Processing for Everyone!", 3rd Edition, Morgan & Claypool.

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.

Dufour C., Dumur G., Paquin J.-N., Belanger J., (2008), "A multi-core pc-based simulator for the hardware-in-the-loop testing of modern train and ship traction systems", 13th Power Electronics and Motion Control Conference, 1475-1480, Poland, 1-3 September.

Gumeniuk P. O., Tokarchuk V. V., Lysenko R. I, (2012), "Software and hardware to control a car model based on dSpace controller", International Conference on Modern Problems of Radio Engineering Telecommunications and Computer Science, 115-116, Lviv, Ukraine, 21-24 February.

Khanna A., Kumar A., Bhatnagar A., Tyagi R., Srivastava S., (2013), "Low-cost production CNC system", 7th International Conference on Intelligent Systems and Control (ISCO), 523-528, Tamil Nadu, India, 4-5 January.

Koren Y., Heisel U., Jovane F., Moriwaki T., Pritschow G., Ulsoy G., Brussel H. (1999), "Reconfigurable manufacturing systems", CIRP Annals-Manufacturing Technology, 48(2), 527-540.

Loffler M. S., Costescu N. P., Dawson D. M., (2002), "QMotor 3.0 and the QMotor robotic toolkit: a PC-based control platform", IEEE Control Systems, 22 (3), 12-26.

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

Maschotta R., Jager S., Jungebloud T., Zimmermann A, (2013), "A framework for agile development of simulation-based system design tools", IEEE International Systems Conference, 861-866, Orlando, IL, USA, 15-18 April.

Peng Y., Shujie W., Jiwei Z., Haiguang L., (2007), "Virtual Reality Platform Based on Open Sourced Graphics Toolkit OpenSceneGraph", 10th IEEE International Conference on Computer-Aided Design and Computer Graphics, 361-364, Beijing, China, 15-18 October.

Qian D., Zhao D., Tian L., Wang Q., (2007), "Linux/RTAI and Scicos in Low Cost High Performance Friction Testing Machine", 8th International Conference on Electronic Measurement and Instruments, 252-254, Xian, China, 16-18 August.

Sarker M. O. F., Kim C. H., Cho J., You B., (2006), "Development of a Network-based Real-Time Robot Control System over IEEE 1394: Using Open Source Software Platform", IEEE International Conference on Mechatronics, 563-568, Budapest, Hungary, 3-5 July.

Wang R., Qian X., (2010), "OpenSceneGraph 3.0: Beginner's Guide", 1st Edition, Packt Publishing.

Web 1, (2014), http://www.qnx.com, (Erişim Tarihi: 03/05/2014).

Web 2, (2001), http://www.clemson.edu/ces/crb/ece496/qmotor_qnx.pdf, (Erişim Tarihi: 25/07/2014).

Web 3, (2014), http://www.opal-rt.com/product/rt-lab-professional-real-time-digital-simulation-software, (Erişim Tarihi: 04/08/2014).

Web 4, (2014), https://www.rtai.org/?About_RTAI-Lab, (Erişim Tarihi: 04/08/2014).

Web 5, (2014), http://www.vissim.com, (Erişim Tarihi: 04/08/2014).

Web 6, (2014), http://www.dspace.com/en/pub/home.cfm, (Erişim Tarihi: 17/08/2014).

Web 7, (2014), http://tr.wikipedia.org/wiki/MATLAB, (Erişim Tarihi: 25/08/2014).

Web 8, (2014), http://www.mathworks.com/products/simulink, (Erişim Tarihi: 25/08/2014).

Web 9, (2014), http://tr.wikipedia.org/wiki/Microsoft_Windows, (Erişim Tarihi: 30/08/2014).

Web 10, (2014), http://www.opal-rt.com, (Erişim Tarihi: 30/08/2014).

Web 11, (2014), http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html, (Erişim Tarihi: 22/06/2014).

Web 12,(2013), http://www.xenomai.org, (Erişim Tarihi: 12/09/2013).

Web 13, (2014), http://www.windriver.com/products/vxworks, (Erişim Tarihi: 28/08/2014).

Web 14, (2014), http://en.wikipedia.org/wiki/PSOS_(real-time_operating_system), (Erişim Tarihi: 28/08/2014).

Web 15, (2013), http://www.xenomai.org/documentation/branches/v2.3.x/pdf/Native-API-Tour-rev-C.pdf, (Erişim Tarihi: 16/04/2013).

Web 16, (2012), http://www.fltk.org, (Erişim Tarihi: 16/11/2012).

Web 17, (2013), http://openvrml.org/doc/index, (Erişim Tarihi: 02/12/2013).

Web 18, (2014), http://www.qt-project.org, (Erişim Tarihi: 27/04/2014).

Web 19, (2013), http://www.openscenegraph.org, (Erişim Tarihi: 08/08/2013)

Web 20, (2014), http://www.qwt.sourceforge.net, (Erişim Tarihi: 14/07/2014) .

Web 21, (2014), http://msdn.microsoft.com/en-us/library/ff649690.aspx, (Erişim Tarihi: 02/09/2014).

Web 22, (2014), http://www.oodesign.com/singleton-pattern.html, (Erişim Tarihi: 02/09/2014).

Web 23, (2014), http://en.wikipedia.org/wiki/Template_method_pattern, (Erişim Tarihi: 05/09/2014).

Web 24, (2014), http://en.wikipedia.org/wiki/Adapter_pattern, (Erişim Tarihi: 06/09/2014).

Web 25, (2014), http://www.raspberrypi.org/two-million, (Erişim Tarihi: 01/07/2014).

Web 26, (2014), http://www.en.wikipedia.org/wiki/Raspberry_Pi, (Erişim Tarihi: 04/04/2014).

Web 27, (2014), http://www.makeuseof.com/tag/arduino-vs-raspberry-pi-which-is-the-mini-computer-for-you, (Erişim Tarihi: 24/02/2014).

Web 28, (2014), http://www.arduino.cc, (Erişim Tarihi: 22/06/2014).

Web 29, (2014), http:// www.en.wikipedia.org/wiki/Arduino, (Erişim Tarihi: 25/05/2014).

ÖZGEÇMİŞ

Hüsnü KARAKÜCÜK 1986 yılında Çorum'da doğdu. 2005 yılında başladığı Gebze Yüksek Teknoloji Enstitüsü (GYTE) Bilgisayar Mühendisliği Bölümü'nü 2010 yılında tamamlayarak, 2011 yılında yüksek lisans eğitimine GYTE Mühendislik ve Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Anabilim Dalında başladı. 2009 yılından bu yana TÜBİTAK BİLGEM'de Araştırmacı ünvanı ile yazılım mühendisi olarak çalışmaktadır.