T.C.
SELÇUK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

GÖMÜLÜ BİLGİSAYAR TABANLI MOBİL ROBOT SİSTEMİ

İSMAİL TAHA ÇÖMLEKCİLER

YÜKSEK LİSANS TEZİ

ELEKTRİK-ELEKTRONİK MÜHENDİSLİĞİ
ANABİLİM DALI
KONYA, 2009

ÖZET

YÜKSEK LİSANS TEZİ

GÖMÜLÜ BİLGİSAYAR TABANLI MOBİL ROBOT SİSTEMİ

İsmail Taha ÇÖMLEKCİLER

Selçuk Üniversitesi Fen Bilimleri Enstitüsü

Elektrik-Elektronik Mühendisliği Anabilim Dalı

Danışman : Yrd. Doç. Dr. Nihat YILMAZ

2009, 165 Sayfa

Jüri : Doç. Dr. Salih GÜNEŞ
Yrd. Doç. Dr. Mesut GÜNDÜZ
Yrd. Doç. Dr. Nihat YILMAZ

Gömülü bilgisayar sistemlerinin giderek yaygınlaştığı günümüzde gömülü bilgisayar tabanlı robotik sistemlerin tasarımı ve yazılım geliştirilmesi bu tez çalışmasının ana amacını oluşturmaktadır. Bu amaçla Microsoft Robotics Studio yazılım geliştirme ortamı tarafından desteklenen ve ARM7 tabanlı bir gömülü bilgisayar sistemi içeren LEGO Mindstorms NXT robot platformu kullanılmış ve bu robot sistemi üzerinde yazılım geliştirilmiştir.

Tez kapsamında gömülü bilgisayar sistemleri genel olarak incelenmiş ve robotik çalışmalarda kullanılan bu sistemlerin programlanması üzerinde durulmuştur. Bu amaçla aşağıda belirtilen 3 farklı yazılım geliştirme ortamı üzerinde denemeler yapılmıştır. Mindstorms için geliştirilen NXT programı, RWTH Mindstorms NXT MATLAB kütüphanesi ve özellikle bu tez çalışmasında detaylı olarak incelenen Microsoft Robotics Studio programı pek çok gömülü robot sisteminde kullanılabilecek niteliktedir.

Microsoft Robotics Studio programı, nesne tabanlı ve sürükle – bırak yöntemini kullanan bir yazılım geliştirme ortamıdır. Sağladığı programlama kolaylığı sayesinde ileri programlama bilgisi iyi olmayan birçok amatör kullanıcıların da profesyonel anlamda çalışabileceği şekilde robotik alanına girmelerini sağlamıştır. Pek çok robot üreticisi de ürettikleri gömülü bilgisayar sistemli robotların yazılımlarının daha kolay ve hızlı bir şekilde yapılabilmesi için fiziksel bir simülasyon ortamını da bünyesinde barındıran Microsoft Robotics Studio programına hazır servis modülleri geliştirerek destek vermişlerdir.

Bu tez çalışması ile robot tasarımında donanım gereksinimlerini büyük ölçüde azaltan gömülü bilgisayar tabanlı robot sistemleri için yazılım geliştirilmesi konusunda türkçe kaynak eksikliği giderilmeye çalışılarak bu yazılımların, robot tasarımı ve yazılım geliştirme sürelerini büyük ölçüde kısalttığı ve kolaylaştırdığı gösterilmiştir.

Anahtar Kelimeler: Gömülü Bilgisayar Sistemi, Lego NXT Mindstorms Robotu, Microsoft Robotik Stüdyo, Gömülü Bilgisayar Tabanlı Mobil Robot

ABSTRACT

MS Thesis

MOBILE ROBOT SYSTEM BASED ON EMBEDDED COMPUTER

İsmail Taha ÇÖMLEKCİLER

Selcuk University

Graduate School of Natural and Applied Sciences

Department of Electrical Electronic Engineering

Supervisor : Assist. Prof. Dr. Nihat YILMAZ

2009, 165 Page

Jury : Assoc. Dr. Salih GÜNEŞ
Assist. Prof. Dr. Mesut GÜNDÜZ
Assist. Prof. Dr. Nihat YILMAZ

Nowadays embedded computer systems become increasingly common, design and software development of embedded computer-based robotic systems have been the main aim of this thesis. For this aim, Lego Mindstorms NXT robot platform supported by robotic studio software development environment and based on ARM7 microprocessor, are used. Software development studies are made on it.

In this thesis, embedded computer systems are analyzed generally, and programming of these systems which are used in robotic studies are detailed. Therefore, experiments are made on three different software development environments which are written as following. NXT program developed for Mindstorms, RWTH Mindstorms NXT MATLAB Library and especially Microsoft Robotics Studio analyzed in this thesis. Microsoft Robotics Studio has the quality which provides to use in a lot of embedded robot system.

Microsoft Robotics Studio is a software development environment which is object based and uses drag and drop methods. Users, who haven't got advanced programming knowledge, can study on robotics as professionally because of the environment provides facility of programming. For easy and rapid software development, lots of robot producers give support with software package to Robotics Studio which have physical simulation environment.

With this thesis, a complete resource in Turkish about software development for embedded computer based robot systems is provided. It is shown that the software shortens the development time and simplifies design.

Key Words: Embedded Computer System, Lego NXT Mindstorms Robot, Microsoft Robotics Studio, Mobile Robot Based On Embedded Computer

TEŞEKKÜR

Tez çalışmam süresince değerli bilgi ve tecrübeleriyle bana yol gösteren, gerekli araştırma ve geliştirme çalışmalarımda yardımlarını esirgemeyen tez danışmanım Sayın Yrd. Doç. Dr. Nihat YILMAZ'a, Selçuk Üniversitesi Mühendislik Mimarlık Fakültesi Elektrik-Elektronik Mühendisliği bölümünün değerli öğretim elemanlarına, teşekkürü bir borç bilirim.

Ayrıca tez çalışmam süresince bana her zaman moral veren ve destek olan değerli eşime ve aileme sonsuz teşekkür ederim.

İÇİNDEKİLER

ÖZET ..... **i**
ABSTRACT ..... **ii**
TEŞEKKÜR ..... **iii**
İÇİNDEKİLER ..... **iv**
SİMGELER VE KISALTMALAR ..... **vii**
ŞEKİLLER LİSTESİ ..... **viii**
ÇİZELGELER LİSTESİ ..... **xi**
1 GİRİŞ **1**
1.1 Projenin Amacı ve Önemi 2
2 KAYNAK ARAŞTIRMASI **3**
3 GÖMÜLÜ BİLGİSAYAR SİSTEMLERİ **8**
3.1 Gömülü Sistem Örnekleri 9
3.2 Gömülü Sistemlerin Tarihçesi 10
3.3 Gömülü Sistemlerin Karakteristikler 12
3.4 Gömülü Sistemlerin Tasarımı 13
3.4.1 Kullanıcı arayüzleri 13
3.4.2 Platform 15
3.4.3 Araçlar 15
3.4.4 Hata ayıklama (Debug) 16
3.4.5 İşletim sistemi 17
3.4.6 Başlangıç 17
3.4.7 Yerleşik Self-Test 17
3.4.8 Güvenilirlik rejimleri 18
3.5 Gömülü Bilgisayar Sistemi Kartları 18
3.5.1 Zotac ION-ITX-A gömülü sistemi 19
3.5.2 Nano 8050 ITX gömülü sistemi 20
3.5.3 WADE-8656 ITX gömülü sistemi 22
3.5.4 SBC2440-IV gömülü sistemi 24
3.5.5 Beagleboard OMAP 3530 gömülü sistemi 26
4 ROBOTLARA GENEL BİR BAKIŞ VE MICROSOFT ROBOTICS STUDIO **28**
4.1 Günümüz Robotları 28
4.2 Robotlardaki Aşamalar 30
4.3 Microsoft'ta Robotlar 31
4.4 Microsoft Robotics Studio'ya Giriş 31
4.5 Bir Geliştirme Dili Seçmek 32
4.6 Microsoft Robotics Studio'nun Anahtar Bileşenleri 33

4.6.1 Runtime (İşlem / İşletim Takibi) ... 33
4.6.2 Eş Zamanlı ve Koordineli İşlem Takibi (CCR) ... 34
4.6.3 Merkezi Olmayan Yazılım Servisleri (DSS) ... 35
4.6.4 Merkezi Olmayan Yazılım Servisleri Protokolü (DSSP) ... 37
4.6.5 İşlem Takibi (RunTime) Servisleri ... 37
4.7 Görsel Simülasyon Ortamı (Visual Simulation Environment) ... 38
4.8 Görsel Programlama Dili (Visual Programming Language [VPL]) ... 41
4.9 Tipik Bir Robot Uygulaması Neye Benzer? ... 42
4.10 MSRS'nin Desteklediği Robotlar ... 43
4.10.1 Desteklenen robotların ve çözüm ortaklarının listesi ... 43
4.10.2 Desteklenmeyen donanımlarla çalışmak ... 49
4.11 Robot Programlama Konusunda Kısa Bir Bilgi ... 49
5 SERVİSLER ... 51
5.1 Robot Uygulamalarının Tanımı ... 51
5.2 Servis Nedir ... 52
5.3 Yeni Bir Servis Oluşturmak ... 52
5.4 Bir Servise Başlamak ... 54
5.5 Servis Projesi Oluşturmak ... 57
5.6 Mesajları Göndermek ve Karşılamak ... 59
5.6.1 Bir mesajın anatomisi ... 60
5.6.2 PortSet kullanımı ... 61
5.6.3 Servis işleyicileri kullanmak ... 63
5.7 Durum Yönetimi ... 64
5.7.1 Dâhili port'a veri nakli ... 65
5.7.2 Durumu Bir Dosyaya Kaydetmek ... 66
5.8 Diğer Servislerle Haberleşme ... 68
5.9 Abonelikleri İşlemek ... 71
5.9.1 Yayımcıyı Yapılandırmak ... 71
5.9.2 Aboneyi Yapılandırmak ... 73
5.10 Çıkışı İşlemek ... 73
5.11 Hataları İşlemek ... 76
5.12 Mesajları Düzeltmek ve İzlemek ... 77
6 GÖRSEL PROGRAMLAMA DİLİ (VPL) ... 80
6.1 VPL'ye Genel Bir Bakış ... 81
6.2 Aktivitelerin Tanımı ... 81
6.3 VPL Dizayn Aracının Kullanımı ... 85
6.4 Bağlantıları Gerçekleştirmek ... 86
6.5 Bir Servisi Yapılandırmak ... 88
6.6 BOE-BOT'la Çalışmak ... 91
6.7 Basit Bir VPL Uygulaması Yazmak ... 92

6.8 Bir VPL Uygulamasını Çalıştırmak ve Hata Ayıklamak 97
6.9 Bir Servis Olarak Derlemek 101
6.10 Özel Bir Activite Oluşturmak 102
7 SİMÜLASYON 105
7.1 Simülasyonların Yapılma Amacı 105
7.2 Görsel Simülasyon Ortamının Kullanılması 106
7.2.1 Donanım gereksinimleri 106
7.2.2 Saniye başına çerçeveleri izlemek 108
7.3 Simülasyon Çalıştırma 109
7.4 Simple Dashboard Servisinin Kullanılması 111
7.4.1 Simple Dashboard servisinin gelişmiş versiyonu 113
7.5 Editör Ayarları 114
7.6 Koordinatlarla Çalışma 116
7.7 Entity Nedir 117
7.8 Yeni Bir Entity Oluşturmak 117
7.9 Fizik (Physics) Modun Aktif ve Pasif Yapılması 124
7.10 Simülatör Durum Dosyasının Düzenlenmesi 124
7.11 Servisten Simülasyon Oluşturma 126
7.12 Yeni Entity'ler Oluşturmak 129
7.12.1 Robot Entityleri Oluşturmak 129
7.12.2 Yeni Entity Türleri Oluşturmak 130
8 GÖMÜLÜ BİLGİSAYAR TABANLI MOBİL ROBOT SİSTEMİ 131
8.1 Donanım 131
8.1.1 Lego NXT Mindstorms robotu 131
8.1.2 Kablosuz CMOS kamera 140
8.2 Yazılım 142
8.2.1 Mindstorms için geliştirilen NXT programı 143
8.2.2 RWTH Mindstorms NXT MATLAB araç kütüphanesi 147
9 DENEYSEL SONUÇLAR 150
9.1 Mindstorms için Geliştirilen NXT Programı Uygulaması 150
9.2 RWTH Mindstorms NXT MATLAB Araç Kütüphanesi Uygulaması 152
9.3 Microsoft Robotics Studio Uygulaması 157
10 SONUÇ VE ÖNERİLER 161
11 KAYNAKLAR 163

SİMGELER VE KISALTMALAR

CCR: Concurrency & Coordination Runtime (Eş zamanlı & Koordineli İşlem Takibi)

CLR: The .NET Common Language Runtime 2.0 (.NET Programlama Dili Tabanlı İşlem Takibi 2.0)

DLL: Dynamic-Link Library (Dinamik-Link Kütüphane)

DSS: Decentralized Software Services (Merkezi Olmayan Yazılım Servisleri)

DSSP: Decentralized Software Services Protocol (Merkezi Olmayan Yazılım Servisleri Protokolü)

GUID: Globally Unique Identifier (Küresel Eşsiz Kimlikleme)

HTTP: Hyper Text Transfer Protocol (Hiper Metin Transferi Protokolü)

IIC (I2C): Inter-Integrated Circuit (Gömülü Tümleşik Devre)

IR: Infrared (Kızılötesi)

MSRS: Microsoft Robotics Studio (Microsoft Robotik Stüdyo)

REST: Temsili Durum Transfer (Representational State Transfer)

SAPI: Speech Application Programming Interface (Konuşma Uygulama Programlaması Arayüzü)

SDK: Software Development Kit (Yazılım Geliştirme Kiti)

SOAP: Simple Object Access Protocol (Tek Obje Erişim Protokolü)

URI: Uniform Resource Identifier (Tek Biçimli Kaynak Belirleyici)

VPL: Visual Programming Language (Görsel Programlama Dili)

VSE: Visual Simulation Environment (Görsel Simülasyon Ortamı)

XSLT: Extensible Stylesheet Language Transformations (Açılabilir Stil Dili Dönüştürücüsü)

XML: Extensible Markup Language (Açılabilir Biçimleme Dili)

ŞEKİLLER LİSTESİ

Şekil 3.1 Gömülü bilgisayar sistem kartı 8
Şekil 3.2 İlk kayda geçen modern gömülü sistem, Apollo Rehber Bilgisayarı 10
Şekil 3.3 Zotac ION-ITX-A gömülü sistemi 19
Şekil 3.4 Nano 8050 ITX gömülü sistemi 21
Şekil 3.5 Nano 8050 ITX gömülü sistemi arka bağlantıları 21
Şekil 3.6 WADE-8656 ITX gömülü sistemi 23
Şekil 3.7 WADE 8656 ITX gömülü sistemi arka bağlantıları 23
Şekil 3.8 SBC2440-IV gömülü sistemi 25
Şekil 3.9 Grafik ekranlı SBC2440-IV gömülü sistemi 25
Şekil 3.10 Beagleboard OMAP 3530 gömülü sistemi 27
Şekil 4.1 Popüler bir oyuncak robot olan "Robosapien" 29
Şekil 4.2 MSRS işlem takibi (Runtime) servisleri 34
Şekil 4.3 VSE simülasyonu servisleri 39
Şekil 4.4 Görsel model simülasyon sahnesi 40
Şekil 4.5 VPL'de tasarım ortamı 42
Şekil 4.6 iRobot firmasının ürettiği "Create" isimli robot 47
Şekil 4.7 Parallax firmasının ürettiği Boe-Bot robotu 47
Şekil 4.8 LEGO Mindstorms NXT robotu 48
Şekil 5.1 LEGO Mindstorms NXT'yi çalıştıran servisler 53
Şekil 5.2 DSS Servislerin tasarlandığı Visual Studio programı 54
Şekil 5.3 MSRS Sistem Servisi Web Sayfası 56
Şekil 5.4 Service Instance Directory'de çalışan servis ve ilişkili partnerlerin listesi 66
Şekil 5.5 Referans ekleme diyalog kutusu. 69
Şekil 5.6 Abone servis ile yayımcı servis arasındaki ilişki. 71
Şekil 5.7 XSLT dosya biçimli ekran görüntüsü 75
Şekil 5.8 Hata ve İzleme Mesajları penceresi 77
Şekil 5.9 Bilgi olarak görüntülen mesaj penceresi 78
Şekil 6.1 VPL robot uygulamaları için sürükle-bırak dizayn ekranı 80
Şekil 6.2 Temel Aktiviteler ve Servisler 81
Şekil 6.3 Boş bir VPL diyagramı 85
Şekil 6.4 Define Variables diyalog kutusu 86
Şekil 6.5 Connections diyalog kutusu ve From – To değerlerinin seçimi. 87
Şekil 6.6 Bağlantı oluşturulduktan sonra oluşan görüntü 88
Şekil 6.7 Servis özelliklerini ayarlama sayfası 89
Şekil 6.8 The Import Manifest diyalog kutusu 90
Şekil 6.9 Parallax Firmasının ürettiği BOE-BOT robottu 92
Şekil 6.10 "Services" araç kutusundan bir servis eklemek 93

Şekil 6.11 Generic Contact Sensors servisini manifest dosyası ile yapılandırmak.... 94
Şekil 6.12 Generic contact sensors servisi ile Data aktivitesi arasındaki bağlantı..... 95
Şekil 6.13 Bloklar arası bağlantı 96
Şekil 6.14 Data Connections diyalog kutusu 96
Şekil 6.15 Basit bir uygulamanın VPL görüntüsü 97
Şekil 6.16 Run diyalog kutusu 98
Şekil 6.17 Microsoft VPL hata ayıklama (Debug) ekranı..... 99
Şekil 6.18 Değişkenlerin incelendiği Güncel Mod ekranı 100
Şekil 6.19 Servis olarak derlenen VPL'nin Visual Stüdyo'daki referans dosyaları 102
Şekil 6.20 Activity diyagram ekranı 103
Şekil 6.21 Actions and Notifications diyalog kutusu..... 104
Şekil 6.22 Örnek bir özel aktivite uygulaması..... 104
Şekil 7.1 Düzenleme modunda simülasyon sahnesinin görüntüsü 106
Şekil 7.2 MVSE hakkında diyalog kutusu. 107
Şekil 7.3 MSRS tarafından gerçekleştirilen arazili simülasyon görüntüsü..... 108
Şekil 7.4 MSRS tarafından sağlanan Dashboard servisi. 111
Şekil 7.5 Simple Dashboard Servisinin gelişmiş bir versiyonu. 113
Şekil 7.6 Grafik ayarları diyalog kutusu. 115
Şekil 7.7 VSE sahnesinde X,Y ve Z eksen hatlarının yönleri..... 116
Şekil 7.8 LEGO NXT'nin arkadan görüntüsü..... 119
Şekil 7.9 Yeni entity diyalog kutusu. 120
Şekil 8.1 Farklı şekillerde dizayn edilmiş NXT robotu 132
Şekil 8.2 NXT Brick gömülü sistemi..... 133
Şekil 8.3 NXT modülüne motor ve sensörlerin bağlantısı..... 135
Şekil 8.4 NXT Mindstorms robotunun ultrasonik sensörü 136
Şekil 8.5 NXT Mindstorms robotunun ışık sensörü 137
Şekil 8.6 (a) İnsan gözü ile görülen renkler (b) Işık sensörü ile görülen renkler..... 137
Şekil 8.7 NXT Mindstorms robotunun ses sensörü 138
Şekil 8.8 NXT Mindstorms robotunun dokunma sensörü 139
Şekil 8.9 Dokunma sensörünün algılama şekilleri..... 139
Şekil 8.10 NXT Mindstorms robotunun servo motoru 140
Şekil 8.11 Kablosuz CMOS kamera ve alıcı modülü 141
Şekil 8.12 National Instrument tarafından geliştirilen NXT programı 143
Şekil 8.13 NXT bağlantı kutusu..... 144
Şekil 8.14 NXT bağlantı kutusunun hafıza bölümü..... 145
Şekil 8.15 NXT programı kontrol butonları..... 145
Şekil 8.16 NXT programı kontrol servisleri 146
Şekil 8.17 Display servisi ve ayar menüsü 147
Şekil 8.18 RWTH Mindstorms NXT MATLAB araç kütüphanesi 148
Şekil 8.19 RWTH Mindstorms NXT MATLAB araç kütüphanesi yardımı..... 149

Şekil 9.1 Kaşif robotu 150
Şekil 9.2 NXT programındaki Kaşif robotu programı görüntüsü 151
Şekil 9.3 Akrep robotu 153
Şekil 9.4 Akrep robotunun MATLAB yazılım ortamı görüntüsü 154
Şekil 9.5 Web kameralı Kaşif robotu 157
Şekil 9.6 Microsoft Robotics Studio programı 158
Şekil 9.7 Web kamera ile takip edilen siyah çizgi 159

ÇİZELGELER LİSTESİ

Çizelge 3.1 Zotac ION-ITX-A gömülü sistemi teknik özellikleri 20
Çizelge 3.2 Nano 8050 ITX gömülü sistemi teknik özellikleri 22
Çizelge 3.3 WADE 8656 ITX gömülü sistemi teknik özellikleri 24
Çizelge 3.4 SBC2440-IV gömülü sistemi teknik özellikleri 26
Çizelge 3.5 Beagleboard OMAP 3530 sistemin teknik özellikleri 27
Çizelge 4.1 MSRS'nin Desteklediği Tüm Robot Üreticileri 44
Çizelge 4.2 MSRS'nin Çözüm Ortakları 46
Çizelge 5.1 DSS ile çalışan komut satırı araçları 53
Çizelge 5.2 İşletilebilir DSSHost parametre listeleri 55
Çizelge 5.3 MSRS'nin sunduğu DSSP işlemleri 62
Çizelge 5.4 HTTP işlemleri 63
Çizelge 5.5 Arbiter sınıfındaki yardımcı metotlar 70
Çizelge 5.6 DSSP servis modelinin sunduğu DSSP Hata Kodları 76
Çizelge 6.1 VPL'deki temel aktiviteler listesi 82
Çizelge 6.2 VPL Aktiviteleri ve MSRS'nin Generic Servisleri 83
Çizelge 6.3 Miscellaneous Servisler 84
Çizelge 7.1 Temel entity özellikleri 118
Çizelge 7.2 MSRS'de bulunan entity türleri 120
Çizelge 7.3 Temel entity durum özellikleri 122
Çizelge 7.4 Özel simülasyon fonksiyonu için kullanılan ilave MSRS servisleri 128
Çizelge 8.1 NXT teknik özellikleri 134
Çizelge 8.2 Kablosuz kameranın teknik özellikleri 141
Çizelge 8.3 Kablosuz kamera alıcısının teknik özellikleri 142

1 GİRİŞ

Mobil robotlar, insanın çalışabilmesinin mümkün olmadığı veya çok zor ve tehlikeli olduğu koşullarda ve süreklilik arz eden işlerde kullanılmaktadır. Günümüzde mobil robotlar; nükleer çalışmalar, bomba imha etme, uzay çalışmaları, gözetleme ve savunma, doğal afet çalışmaları, zararlı kimyasal maddelerin taşınması ve işlenmesi sahalarında daha fazla yer bulmaktadır.

Gömülü bilgisayar tabanlı mobil robotlar, bünyesinde ON-BOARD özellikli bilgisayar modülü barındırması ve bir işletim sistemi çalıştırabilmesi sayesinde profesyonel uygulamalar için birçok ekstra avantaj ve kolaylıklar sunmaktadır. Birden fazla haberleşme protokolünü barındırması neticesinde ister bilgisayar ile isterseniz bir başka mobil robot ile haberleşmesine imkân sağlamaktadır. Yüksek duyarlıklı aritmetik ve lojik işlemleri hatasız ve yüksek hızlarda yapabilmesi özelliği ile de bize birçok alanda (örneğin görüntü işleme, yapay sinir ağları, bulanık mantık… vb. gibi) hesaplama gerektiren alanları da kolayca kontrol etmemizi sağlamaktadır.

Gerçekleştirilen bu tez çalışmasında gömülü sistemlerin mobil robotlarda uygulaması üzerine çalışılmıştır. Çalışmanın ilk bölümünde robotik alanına genel bir giriş yapılmış ve ilerleyen bölümlerde gömülü sistemlerin robotik alanına sağladığı kolaylıklardan bahsedilerek Microsoft'un geliştirmiş olduğu MSRS ayrıntılı bir şekilde anlatılmıştır. MSRS, Microsoft Windows tabanlı, servise yönelik işletim aşamaları incelenebilen, görsel kullanıcı araçları bulunan, örnekler ve dokümanlarla desteklenmiş bir yazılımdır. Ticari ve akademik araştırmacılara üzerinde çalışabilecekleri bir platform sunan MSRS araç kutusu, aynı zamanda Görsel Programlama Dili (Visual Programming Language) denilen bir programlama dilini de kullanabilme imkânı sunmaktadır. VPL bir tasarım bir grafiksel araçtır. VPL, basit veya karmaşık robot uygulamaları üzerinde çalışılabilen grafiksel bir uygulama tasarım aracıdır. Kullanımı oldukça kolaydır. Dizayn yüzeyi üzerine sürükle – bırak şeklindeki programlama öğelerini kullanılır ve giriş – çıkışları belirlemek için bağlantı uçları kullanılır. MSRS'nin önemli özelliklerinden biri de gerçekleştirilen uygulamaların sanal ortamda simüle edilebiliyor olmasıdır. Görsel Simülasyon

Ortamı (VSE) tasarımcılara herhangi bir robot satın almadan istedikleri özellikte bir robotu oluşturarak amatör veya profesyonel olarak çalışmalarını sağlar. Bu özelliği sayesinde grup çalışması yapan robot tasarımcıları aynı robotu gruptaki her bir kişi için satın almak yerine, VSE ortamında yüksek maliyet ve zaman kayıpları olmadan test edip geliştirebilirler. VSE 'ye ilave olarak programcılar kendi tasarladıkları bir robotu da, MSRS desteklemese bile Visual Stüdyo'yu kullanarak MSRS'nin desteklediği robotlar şekline getirip simülasyon oluşturabilirler.

Bu tez çalışmasında gömülü bilgisayar sistemli bir mobil robot tasarlanarak, gömülü bilgisayar sistemlerinin robotik alanındaki sağladığı avantajlar gerçek uygulamalar yapılarak gösterilmeye çalışılmıştır.

1.1 Projenin Amacı ve Önemi

Mobil bir robotun bünyesinde ON-BOARD özellikli gömülü bir bilgisayar modülünün bulunmasının sağladığı ayrıcalıklar ve kolaylıklar bu tez projesi kapsamında gösterilmeye çalışılacaktır. Üzerinde işletim sistemi yüklü olan bu robotlarda daha karmaşık algoritmaların denenmesi daha da kolaylaşacaktır. Yüksek seviyeli bir programlama dili kullanılarak yazılım geliştirilebilecek olması da bu projenin önemini artırmaktadır. Bu sayede akademik çalışmaların hızlanması ve daha profesyonel bir çalışmanın gerçekleştirilebilmesinde ne kadar büyük katkılarının olduğu ve projelerde robotun donanım ve yazılımı için harcanan zamanın gömülü bilgisayar tabanlı mobil robotlar sayesinde en az düzeye ulaştığı gösterilmeye çalışılacaktır.

Bu tez projesinde, mobil robotlarda kullanılan sensörlerin, haberleşme elemanlarının ve robotu kontrol eden mekanik ve elektronik düzeneklerin gömülü bilgisayar özelliği ile daha rahat ve daha kolay kontrol edilebileceği gösterilecektir.

Bu tez çalışması ile gömülü bilgisayar tabanlı mobil robot sistemleri için yazılım geliştirilmesi konusunda türkçe kaynak eksikliği giderilmeye çalışması amaçlanmıştır.

2 KAYNAK ARAŞTIRMASI

A. Kosaka ve A. C. Kak (1992) tarafından yapılan çalışmada, bina içi navigasyon için FINALE sistemi önerilmiştir. FINALE sistemi sıradan bir PC tabanlı mimariyi ve robotun kendi yer belirleme yöntemini kullanarak yol almıştır. FINALE sistemi, NEURO-NAV ve FUZZY-NAV yöntemleri kullanılarak oluşturulmuştur.

M. Meng ve A. C. Kak (1993) tarafından yapılan "NEURO-NAV: A Neural Network Based Architecture for Vision-Guided Mobile Robot Navigation Using Non-Metrical Models of the Environment," ve "Mobile Robot Navigation Using Neural Networks and Nonmetrical Environment Models" isimli çalışmalar, belirteçlerin algılanması ve özelliklerin açığa çıkartılması için sinir ağlarını ve boşluğun topoğrafik bir sunumunu kullanmıştır.

J. Santos-Victor, G. Sandini, F. Curotto ve S. Garibaldi tarafından 1993 yılında yazılan makalede arıların görsel davranışını taklit eden optik-akış tabanlı bir sistem geliştirmişlerdir. Robotun, bir arının merkezleme refleksini taklit etmesi için farklı bir stereo yaklaşımı kullanmışlardır. Robot, bir koridorun merkezindeyse, sol ve sağ kamerayla görülen görüntünün hızı arasındaki fark yaklaşık olarak sıfırdır ve robot, koridorun ortasında kalır. Ancak hızlar farklıysa, robot, görüntünün daha küçük hızla değiştiği kenara doğru hareket eder. Robotik uygulamasına göre temel fikir, sol ve sağ görüntü hızları arasındaki farkı ölçmek ve robota rehberlik etmesi için bu bilgiyi kullanmaktır.

J. Pan, D.J. Pack, A. Kosaka ve A.C. Kak (1995) tarafından yazılan makalede, robotun navigasyon davranışının kontrolü için kural tabanlı yüksek düzeyli denetleyicide birleştirici bulanık mantık kullanılmıştır ve böylelikle NEURO-NAV geliştirilmiştir.

Kim ve Nevatia (1995) tarafından yazılan makalede farklı bir yaklaşım önerilmiştir. Bu yaklaşımda kapı ve masa gibi belirteçler kullanılmıştır. Simgesel bir navigasyon olan bu yaklaşımda robot, "kapıya git" veya "önündeki masaya git" gibi emirleri alır ve tanımak için ihtiyaç duyduğu belirteçleri ve yolu saptamak amacıyla bu emirlerin içerdiği simgesel bilgiyi kullanır. Örneğin, "Önündeki masaya git" emri, belirtecin masa olduğunu ve yolun ileride olduğunu işaret ederek robota söyler.

A. Dev, B. Kröse ve F. Groen tarafından 1997 yılında yazılan makalede, yine optik akış kullanarak optik akıştan derinlik bilgisi çıkartılmasıyla duvar takibi uygulaması gerçekleştirilmiştir.

Guilherme N. DeSouza ve Avinash C. Kak tarafından 2002 yılında yapılan çalışmada, mobil robotun görme tabanlı navigasyonu, bina içi ve bina dışı olmak üzere iki temel alt başlıkta incelenmiştir. Her bir başlık kendi içinde yapılandırılmış ve yapılandırılmamış çevre temelli navigasyon olarak alt başlıklara ayrılmıştır. Yapılandırılmış çevreler için, alanın topoğrafik ve geometrik şekillerinin durumları dikkate alınırken, yapılandırılmamış çevreler için; optik akış, özel nesnelerin tanınması ve görünüş tabanlı modeller kullanan navigasyon türleri açıklanmıştır.

Parhi'nin 2005 yılında yaptığı çalışmada bulanık mantık tabanlı olarak yönlendirilmesi yapılan bir robot sistemi kurulmuştur. Tüm bulanık kontrol sistemi bir gömülü bilgisayar sisteminde çalıştırılmaktadır. Bu kontrol sistemi ile farklı çalışma ortamlarında çarpışmaları önleyecek şekilde tasarlanmış ve başarı ile test edilmiştir.

Ning ve Yang 2006 yılında yayınladıkları makalelerinde, gömülü bir bilgisayar sistemi üzerinde kompleks mekatronik bir sistem için MAS tabanlı bir kontrol sistemi geliştirmişlerdir. Bu sistemde ARM7 mikrodenetleyicisini kullanan ve yaptıkları robota PBJ-2 ismini veren ekip aynı zamanda telerobot da olan bu robot üzerinde geliştirdikleri algoritmanın etkinliğini başarı ile test etmişlerdir.

Albus ve arkadaşlarının 2006 yılında yaptığı ve DARPA (Defense Applied Research Projects Agency) projelerinden birini konu alan bu çalışmada kullanılan gömülü bilgisayar yapısı ile öğrenebilen ve öğrendikleri ile navigasyon sistemlerine destek sağlayabilen bir robot sistemi sunulmuştur. Gömülü bilgisayar sistemi üzerinden 4D/RCS ismi verile model-referans kontrol sistemi kurulmuştur. Bu öğrenebilen gömülü bilgisayar sistemi ile sensör verilerinin işlenmesi, ortamın modellenmesi ve robot davranışlarının üretilmesi işlemleri yürütülmektedir.

Kulyukin, V. ve C. Gharpure tarafından 2006 yılında yayınlanan makalelerinde, görsel olarak bozuk yollara sahip havaalanı veya süpermarketlerde kullanılmak üzere dizayn edilmiş bir mobil robot platformu bu çalışmada geliştirilmiştir. Pasif RFID sensörler yardımıyla yolunu bulan bu robot sisteminin performansı test edilmiştir.

J.B. Hayet, F. Lerasle, M. Devy (2007) tarafından yazılan makalesinde, tek bir kamerayla yakalanan planar dörtgenler olan görsel belirteçlerin tanınması ve çıkartılmasında kullanılan, görsel fonksiyonlar açıklanır. Belirteçler parça setleri üzerinde geometrik, topolojik ve görünüş sınırlamalarına uygulanmak için kullanılan bir gevşeme şemasındaki (relaxation scheme) kenar bölümlerinden çıkartılır. Belirteç gibi çıkartılanlardan birisi, değişmeyen nitelikler tarafından karakterize edilir böylece geniş bir görüş açısından tanımanın yapılması mümkün olacağı açıklanmıştır.

J.L. Guzman, M. Berenguela, F. Rodrigueza, S. Dormidob (2007) tarafından yazılan makalesi, mobil robot hareketi problemlerini çözümünü içeren iyi bilinen pek çok algoritmayı ve teknikleri anlamamızı kolaylaştırmayı hedefleyen etkileşimli araçları anlatır. Bu araçlar, hareketliliğin modellenme mekanizmalarından navigasyondaki kullanımlarına kadar uzanır. Bu araç farklı disiplinler boyunca eğitimsel olarak faydalı olması amacıyla bu problemlerin basit bir anlatımla açıklanmasına çalışır.

Xiuqing Wanga, Zeng-Guang Houa, Anmin Zoua, Min Tana, Long Cheng (2007) tarafından yazılan makalesinde bağlayıcı sinir ağları (Spiking Neural Networks, SNN' ler) üçüncü nesil yapay sinir ağları olarak, eşsiz avantajlara sahiptir ve robot denetleyicileri için uygun adaylardır. Mobil robotlar için ultrasonik sensör sinyalleri kullanarak engellerden sakınmak için bağlayıcı sinir ağı temelli bir davranış kontrolörü tasarlanır. Yapının ayrıntıları ve kontrolörün uygulamasından bahsedilmiştir. Kontrolörde tümleşik çalıştırma modeli kullanılır ve SNN Hebbian öğrenme algoritması ile eğitilir. SNN'lerin çatısı altında, kontrolörde klasik sinir ağlarındakinden (NN) daha az nöron görevlendirilir. Deneysel sonuçlar göstermiştir ki önerilen kontrolör daha verimlidir ve uygulanması daha basittir.

Luigi Freda ve Guiseppe Oriolo (2007) tarafından yazılan makalesinde iki seviyeli bir yapı önerilmiştir. Düşük seviyede, pan-tilt platformu üzerinde taşınan kamera hedefi görüntü düzleminin merkezine mümkün olduğunca yakın tutabilmesi için kontrol edilmektedir. Yüksek seviyede, hedefin bağıl pozisyonu basit geometriyle kameranın pan-tilt açısından ve görüntü koordinatlarından elde edilmiştir ve robotu hedefe götürecek kontrol hesaplamasında kullanılmıştır. Mümkün olan farklı seçenekler yüksek seviyeli robot kontrolörleri için ele alınmıştır ve ilişkilendirilen kararlılık özellikleri yaklaşık olarak analiz edilmiştir. Önerilen

görme yakalama metodu MagellanPro mobil robotu üzerindeki deneylerle simülasyonlar üzerinde onaylanmıştır.

Squire ve Levinso'nun 2007 yılındaki yaptıkları robot sisteminde ise bir konuşma dili öğretme üzerine çalışma yapılmıştır. Dil öğretim algoritması olarak gizli Markov modeli tercih edilmiş ve tüm bu algoritmalar robotun üzerinde bulunan bir gömülü bilgisayar üzerinde yürütülmüştür.

Bu çalışmada, Ma, Z., Y. Hu, ve arkadaşları (2007) çok uzun boruların iç yüzeylerinin incelenmesi için kullanılan bir boru robotun tasarımı anlatılmıştır. Bu tasarımda bir gömülü bilgisayar sisteminin kontrol ettiği robot ve buna bağlı 3 boyutlu lazer tarayıcı, 4 serbestlik dereli robot mekanizması ve yazılım birimleri mevcuttur. Ayrıca bu çalışmada kullanılan gömülü bilgisayar tabanlı robot üzerinde adaptif kontrol teknikleri de denenmiştir.

Bu çalışmada, Kim, T. H., S. H. Choi, ve arkadaşları (2007) üzerlerinde gömülü bilgisayar sistemleri olan mobil robot takımlarında HYTECH yazılımı ile formal modelleme paradigmasını uygulamışlardır. Bu paradigma ile ilgili parametrelerin optimizasyonu ile çalışılmış ve bu çalışmanın, daha kompleks robot sistemlerinin otomatik olarak analizi çalışmalarına bir başlangıç olacağı öne sürülmüştür.

Clark ve Fierro'nun 2007 yılında yaptıkları araştırma makalesinde, mobil robotların en fazla ihtiyaç duydukları ekipmanlar, sensörler ve sensör şebekeleri olduğunu vurgulamışlardır. Askeri, kurtarma, keşif, görüntüleme gibi alanlarda çalışan tüm robotlar sensörleri ve bunların pek çoğunu içeren sensör ağlarını (sensör dizileri) kullanırlar. Bu çalışmada robot sistemlerinde kullanılan sensör ağlarında elde edilen verilerin işlemesi ve daha anlamlı verilerin üretilmesi için gömülü işlemcilerin kullanılması üzerinden durulmuştur.

Clark, J. ve R. Fierro 2007 yılında bir yazılım robotu (Sobot) olan Ubibot'ın mobil robotlar ve gömülü robotlar ile haberleşmesi ve bunların fiziksel sınırlamaları bu çalışmasını konusunu teşkil etmektedir. Bu robot türleri birbirleri ile çalışırken bir orta katman kullanırlar ve bu katmanın da fiziksel sınırları vardır. Bu katmanın verimliliği gerçek deneyler ile Sobot ve Robotlar üzerinde gösterilmiştir.

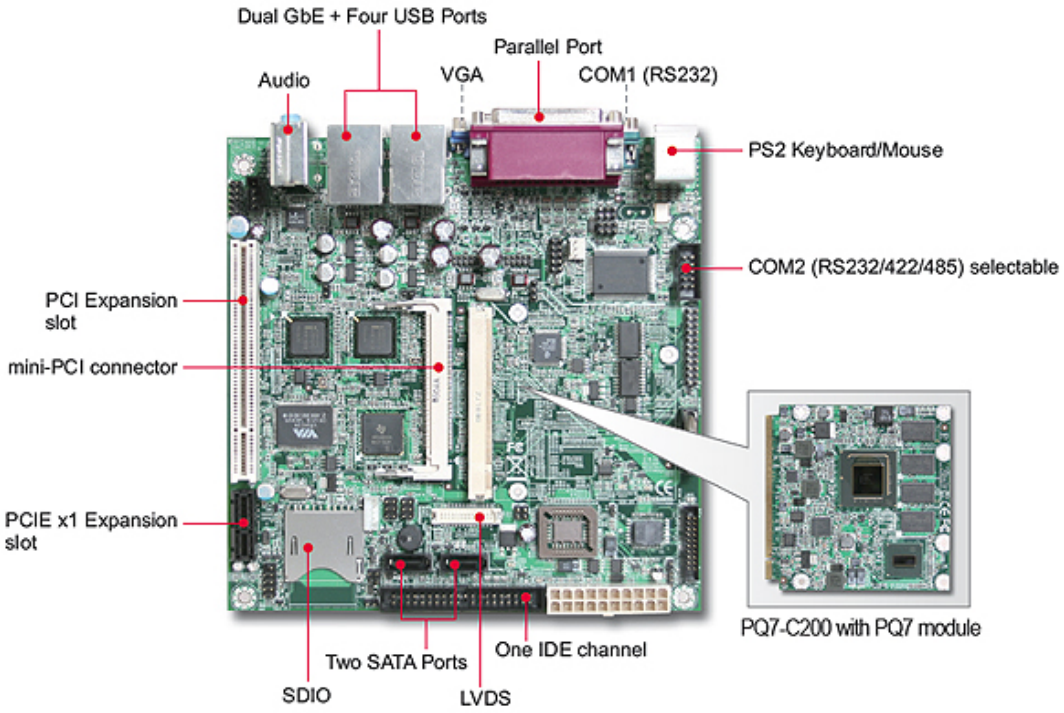
Chen ve arkadaşları 2007 yılında yaptıkları çalışmada, üzerinde bir kamera bulunan gömülü bir bilgisayar ile görsel olarak yönlendirilen bir mobil robot sistemi

tasarlanmıştır. Kamera kalibrasyonunun da sistemin çalışması sırasında gömülü bilgisayar kullanan robot sistemi tarafından yapılması üzerine çalışılmıştır. Kalibrasyon işleminin performansını artırma ve bu çalışmalarda kullanılmak üzere test hareketleri ve şartları hazırlanmıştır.

3 GÖMÜLÜ BİLGİSAYAR SİSTEMLERİ

Gömülü sistem, içerisinde kendisini kontrol eden bilgisayarı içeren özel amaçlı bir sistemdir. Genel amaçlı olup, örneğin kişisel bilgisayar gibi, gömülü bir sistem kendisi için önceden özel olarak tanımlanmış görevleri yerine getirir. Bu sistemler belirli bir amaca yönelik olduğu için tasarım mühendisleri ürünün boyutunu ve maliyetini azaltarak sistemi optimize edebilirler. Şekil 3.1'de gelişmiş bir gömülü bilgisayar sistemi kartı görülmektedir.

Gömülü bir sistemin çekirdeğini, belirli bir sayıda görevi yerine getirmek için programlanan mikroişlemciler ya da mikrodenetleyiciler oluşturur. Kullanıcıların üzerinde istediği yazılımları çalıştırabildiği genel maksatlı bilgisayarlardan farklı olarak, gömülü sistemlerdeki yazılımlar yarı kalıcıdırlar ve firmware ismiyle anılırlar.



Şekil 3.1 Gömülü bilgisayar sistem kartı

3.1 Gömülü Sistem Örnekleri

Hayatımızda her alanında gömülü sistem örnekleri o kadar çok ki neredeyse her işimizi dolaylı olarak gömülü sistemler hallediyor diyebiliriz. Gömülü sistemlerle ilgili birkaç örnek aşağıda verilmiştir.

- Banka ATM' leri
- Eylemsiz rehber sistemleri, uçuş kontrol donanım/yazılımı ve uçak ve füzelerdeki diğer tümleşik sistemlerden oluşan havacılık elektroniği modülleri
- Cep telefonları
- Router (yönlendirici), time server ve firewall (güvenlik duvarı) gibi bilgisayar ağ ekipmanları
- Bilgisayar yazıcıları
- Fotokopi makineleri
- Disket sürücüler (floppy disket sürücüler ve sabit disk sürücüler)
- Termostat, klima, sprinkler ve güvenlik izleme sistemleri gibi ev otomasyonu ürünleri
- Hesap makineleri
- Mikro dalga fırınlar, çamaşır makinesi, televizyon setleri ve DVD oynatıcı/kaydedici gibi ev elektroniği ürünleri
- Tıbbi ekipmanlar
- Çok fonksiyonlu kol saatleri
- İnternet, radyo alıcıları, TV set-top-box ve dijital uydu alıcılar gibi multimedya uygulamaları
- Çok fonksiyonlu yazıcılar
- PDA'lar gibi küçük avuç içi bilgisayarlar
- PDA ve Java destekli gelişmiş cep telefonları
- Endüstriyel otomasyon ve izleme için PLC' ler
- Video oyun konsolları ve avuç içi oyun konsolları
- Taşınabilir bilgisayarlar

3.2 Gömülü Sistemlerin Tarihçesi

Kayda geçen ilk gömülü sistem MIT Instrumentation Laboratory'da Charles Stark Draper tarafından geliştirilen Şekil 3.2'de gösterilen Apollo Guidance Computer (Apollo Rehber Bilgisayarı) olmuştur. Aya yapılan yolculuklarda iki tane kullanılırdı ve komuta modülü sistemlerini çalıştırıyordu.



Şekil 3.2 İlk kayda geçen modern gömülü sistem, Apollo Rehber Bilgisayarı

Projenin başlangıcında Apollo rehber bilgisayarı Apollo projesinin en riskli parçası olarak kabul ediliyordu. O zamanki tek parça tümleşik devrelerin kullanılması boyut ve ağırlığı azaltıyor ama riski artırıyordu.

İlk kitlesel gömülü sistem üretimi 1961 yılında Minuteman füzesi için yapılan Autonetics D–17 rehber bilgisayarı oldu. Ayrık transistor lojiğinden yapıldı ve ana bellek için bir hard diski vardı. 1966 yılında Minuteman II üretime girdiğinde, D–17 ilk defa yüksek hacimli tümleşik devrelerin kullanıldığı yeni bir bilgisayara yerini bıraktı. Bu program dörtlü NAND kapılı entegrelerin birim fiyatını 1000$'dan 3$'a çekti ve ticari kullanımlarının yolunu açtı

Minuteman bilgisayarının önemli tasarım özellikleri, füzenin hedefi daha hassas bulabilmesi için rehber algoritmasının yeniden programlanabilir olması ve bilgisayarın kablo ve konnektörden tasarruf sağlayarak füzeyi test edebilmesiydi.

Maliyetin dikkate alınmadığı 1960'lardaki bu ilk uygulamalardan itibaren gömülü sistemlerin fiyatları düşmeye başladı. Bunlarla birlikte işlem gücü ve fonksiyonellikte de yükseliş oldu.

İlk mikroişlemci, hesap makineleri ve diğer ufak sistemlerde kullanılan Intel 4004 oldu. Çalışabilmesi için harici bellek çipleri ve harici destek lojiklerine ihtiyaç duyuyordu. Intel 8080 gibi daha güçlü mikroişlemciler askeri projelerde geliştirildi ama diğer kullanıcılara da satıldı.

1970'lerin sonunda 8-bit mikroişlemciler standart olmakla birlikte çözümleme ve giriş/çıkış işlemleri için genellikle harici bellek çipleri ve lojiklere ihtiyaç duyuyorlardı. Öte taraftan, fiyatlar hızla düşüyor ve uygulamalar küçük gömülü sistemleri lojik tasarımların içine sokuyordu. Görünebilir uygulamaların bir kısmı enstrümantasyon ve pahalı aygıtlardı.

1980'lerin ortalarında harici olarak kullanılan sistem parçaları, işlemci ile beraber aynı çipin içine girmeye başladı. Bunun sonucu olarak boyutta ve gömülü sistemlerin maliyetinde çok büyük düşüşler oldu. Bu tip tümleşik devrelere mikroişlemci yerine mikrokontrolör veya mikrodenetleyici dendi ve gömülü sistemlerin yaygın bir şekilde kullanımı mümkün oldu.

Mikrokontrolör maliyeti bir mühendisin 1 saatlik maaşının altına indi ve bu gömülü sistemlerin sayısını ve gömülü sistemlerde kullanılmak üzere farklı firmalar tarafından üretilen parçaların sayısının patlamasına neden oldu. Örneğin, pek çok yeni özelliğe sahip entegreler, geleneksel paralel programlama ara yüzleri yerine mikrokontrolörlere daha az sayıda ara bağlantı sağlayan seri programlama ara yüzleri ile beraber gelmeye başladı. I2C'nin çıkış zamanı da bu döneme rastlamaktadır. Mikrokontrolörler 1$'ın altına düştüğünde, voltmetre ve değişken kapasitör gibi pahalı analog elemanların yerlerini küçük bir mikrokontrolör ile kontrol edilen dijital elektronik elemanlara bırakması mümkün oldu.

80'lerin sonundan itibaren, tüm elektronik cihazlar için gömülü sistemler bir istisna değil bir standart haline geldi ve bu trend halen hızlı bir şekilde yükselmeye devam etmektedir.

3.3 Gömülü Sistemlerin Karakteristikler

Gömülü sistemler en geniş manasıyla bilgisayar sistemidir. Gömülü sistemlerin örnekleri taşınabilir müzik çalıcılardan uzay araçlarındaki alt sistemler için kullanılan gerçek zamanlı kontrol sistemlerine kadar uzanır. En ticari gömülü sistemler, düşük maliyette bir takım işleri gerçekleştirmek için tasarlanıyor. Hepsi olmamakla birlikte büyük çoğunluğunun gerçek zamanlı sistem kısıtlamalarını karşılaması gerekir. Bazı fonksiyonlarının çok hızlı olması gerekebilirken, diğer pek çok fonksiyonunun kesin zamanlama gereksinimi olması gerekmemektedir. Bu tip sistemler, gerçek zaman kısıtlarını sistem gereksinimlerine bağlanmış özel amaçlı yazılım ve donanım kombinasyonları ile karşılarlar.

Gömülü sistemleri hız ve maliyet ile karakterize etmek zor olmakla beraber, maliyeti azaltan yüksek hacimli sistemler öncelikli tasarım hedefidir. Genellikle gömülü sistemlerin düşük performans gereksinimleri olur. Bu sistem donanımını, maliyeti düşürmek için basitleştirme olanağı tanır. Mühendisler gerekli fonksiyonları sağlayan "yeteri kadar iyi" olan donanımları seçerler.

Örneğin, uydu televizyon için dijital bir set-top box'ın saniyede onlarca mega bit veri işlemesi gerekir ama bu işlemlerin büyük bölümü çok kanallı dijital videoyu ayrıştıran, yönlendiren ve çözen özel tümleşik devreler tarafından gerçekleştirilir. Gömülü CPU bu işlemi kurar ve set-top box'ın kullanıcı ara yüzü gibi görsel öğelerini ekrana basar. Gömülü işlemcilerin hızları artıp, fiyatları ucuzlarken, yüksek hızlı veri işlemede öncülüğü daha fazla ele almaktadırlar.

Düşük hacimli gömülü sistemler için kişisel bilgisayarlar, programları sınırlayarak ya da işletim sistemini gerçek zamanlı işletim sistemiyle değiştirerek kullanılabilir. Bu durumda özel amaçlı donanım bir ya da daha fazla yüksek performanslı CPU ile değiştirilebilir. Bazı gömülü sistemler istenen görevleri yerine getirmek için, yüksek performanslı CPU, özel donanım ya da büyük belleklere ihtiyaç duyabilir.

Taşınabilir müzik çalıcılar ve cep telefonları gibi büyük hacimli gömülü sistemlerde, maliyeti düşürme yönünde yapılan çalışmalar öncelik kazanır. Bu sistemler genellikle birkaç tümleşik devre, bütün fonksiyonları kontrol eden tümleşik

CPU ve tek bir bellek çipi içerirler. Bu tasarımlarda her bir eleman bütün sistem maliyetini düşürecek şekilde seçilir ve tasarımları bu şart altında yapılır.

Bir disk sürücüsü olmayan gömülü sistemler için yapılan yazılıma firmware adı verilir. Firmware yazılımları cihazın içinde bir ya da daha fazla ROM ya da flash bellek yongalarında gömülü olarak bulunur. Gömülü sistemlerdeki programlar genellikle pek çok donanım kaynağından mahrum bir şekilde çalışırlar. Genellikle bu tip sistemlerde disket sürücü, işletim sistemi, klavye ya da ekran yoktur. Eğer bir kullanıcı ara yüzü var ise küçük bir klavye ya da likid kristal bir ekran gerekebilir.

Gömülü sistemler, yıllarca hatasız bir şekilde çalışacağı varsayılan donanımların içinde bulunur. Bu yüzden gömülü sistem içindeki yazılımlar, kişisel bilgisayar içinde olanlardan daha dikkatli bir şekilde geliştirilip test edilir. Pek çok gömülü sistemde, disket sürücü, anahtar ya da düğme gibi mekanik parçaların kullanımından sakınılır çünkü bu tip parçalar flash bellekler gibi statik parçalara kıyasla daha az güvenilirdir.

Bütün bunlara ek olarak, gömülü sistemler insanların kolaylıkla ulaşabileceği yerlere uzak olabilirler (bir petrol kuyusunun dibinde ya da uzayda bulunan bir uydu üzerinde). Dolayısıyla gömülü sistemler çok büyük veri kayıpları yaşandığı durumlarda bile kendilerini yeni baştan başlatabilmelidirler. Bu tip bir işlev watchdog timer adı verilen standart elektronik bir parça tarafından yerine getirilir.

3.4 Gömülü Sistemlerin Tasarımı

Elektronik cihazlar genellikle mikroişlemci ya da mikrodenetleyici kullanırlar. Bazı büyük ya da eski sistemler ise genel maksatlı büyük bilgisayarlar ya da mini bilgisayarlar kullanırlar.

3.4.1 Kullanıcı arayüzleri

PARC, Apple Computer, Boeing ve HP' de bulunan arayüz tasarımcıları, kullanıcı eylemlerinin tipinin sayısının azaltılması gerektiği prensibini keşfettiler.

Gömülü sistemlerde bu prensip daha düşük maliyetlere yönelik çalışmalarla birleştirilmektedir.

Gömülü sistemlerde yaygın olarak kullanılan standart bir arayüz iki tuştan oluşmaktadır; bunlardan biri menü sistemini kontrol etmek için diğeri ise istenilen seçimi gerçekleştirmek için kullanılmaktadır. Menüler kendilerini belgeleyebildikleri ve çok basit kullanıcı eylemleri ile seçilebildikleri için oldukça popülerdir.

Bir diğer yöntem ise çıktı tipini ufaltmak ve basitleştirmektir. Tasarım bazı durumlarda her bir arayüz çıkışı ya da sistem hatasını bilgilendirmek amacı ile ışık veren LED içerir. Ucuz bir seçenek olması açısından üzerine kullanıcının yerel dilini içeren hata çıkıntılarını gösteren baskılı matris etiketleri yapıştırılmış ışık barları kullanmaktır. Örneğin, pek çok küçük bilgisayar yazıcısı üzerinde herhangi bir dilde yazılar olan etiketlerle etiketlenmiş ışıklar kullanmaktadır. Bazı marketlerde bu tip ürünler müşterinin seçeceği dilde hazırlanmış etiketlerle satışa sunulmaktadır.

Kullanılan bir başka yöntem modların kullanıcı ekranında açıkça görünür hale getirilmesidir. Eğer bir arayüzün modları varsa, her zaman bir yöntemle ya da ara yüzün kendisi tarafından otomatik olarak tersine çevrilir. Örneğin, Boeing'in standart test ara yüzü bir düğme ve birkaç ışıktan oluşur. Düğmeye basıldığı zaman ışıklar yanar ve bırakılması ile birlikte hata mesajı veren ışıklar yanar. Kullanılan etiketler yalın bir İngilizce ile hazırlanmıştır.

Tasarımcılar sıklıkla farklı renkleri kullanırlar. Kırmızı tehlike anlamına gelir ya da tüm sistemi etkileyecek bir hatanın varlığını belirtir. Sarı bir takım problemlerin olduğunu yeşil ise sistem durumunda herhangi bir aksaklık olmadığını belirtir. Seçilen renkler birçok insanın anlaması açısından trafik işaretlerindeki renkler olarak belirlenir.

Eğer yapılan tasarım bir ekran gerektiriyorsa tasarımcılar genelde düz metin kullanımını tercih ederler. Eğer ürün görsel eğlence öğeleri üzerinde kurulmuş bir tasarımdan ibaretse, görselliği zengin metinler, resimler ve menüler ürün için tercih edilmektedir.

3.4.2 Platform

Gömülü tasarımlarda kullanılabilecek ARM, MIPS, Coldfire/68k, PowerPC, X86, PIC, 8051, Atmel AVR, Renesas H8, SH, V850, FR-V, M32R vb. gibi pek çok işlemci mimarisi bulunmaktadır.

Standart PC/104 küçük hacimli gömülü sistem tasarımları için sıkılıkla tercih edilmektedir. Bu tasarımlar genellikle DOS, Linux, NetBSD veya QNX ya da Inferno gibi gerçek zamanlı işletim sistemleri kullanmaktadır.

Büyük hacimli gömülü sistem tasarımlarında ise genellikle tek bir çip üzerinde toplanmış sistemler tercih edilir ve bunun içinde uygulamaya yönelik tümleşik devre tasarımları kullanılır. Bu tasarımlar genelde CPU dâhil tüm lojik tasarımlar FPGA kullanarak gerçeklenebilir.

3.4.3 Araçlar

Gömülü sistem tasarımcıları, bilgisayar programcıları gibi derleyici, çevirici ve hata ayıklayıcı gibi araçları gömülü sistem geliştirmekte kullanırlar. Bununla birlikte, pek çok programcıya yabancı gelebilecek bir takım araçlarda geliştirme sürecinde kullanılır.

Yazılım araçları birkaç kaynaktan elde edilebilir:

- Gömülü pazarda uzmanlaşmış yazılım şirketleri
- GNU yazılım geliştirme araçlarından çalışılacak ortama taşınarak
- Bazen de işlemci mimarisi kullanılacak gömülü işlemci mimarisine yakın olan kişisel bilgisayarlarda kullanılan yazılımlardan faydalanılabilir.

Bilgisayar programcılarının tercih etmediği ama gömülü sistem tasarımcıları tarafından kullanılan birkaç yazılım aracı bulunmaktadır;

- Sıklıkla kullanılan bir araç “in-circuit emulator” (ICE) daha modern tasarımlarda ise gömülü bir hata ayıklayıcı. Bu hata ayıklama aracı gömülü kod geliştirmede kullanılan temel yapılardan biridir. Mikroişlemci’ ye yapılan bağlantı ile sistemde geliştirilen kodların çabuk ve hızlı bir şekilde yüklenmesini ve ayıklanmasını sağlar.

- Gömülü bağlayıcıların (linker) kodun boyutunu ufaltmak ve çalışma zamanını azaltmak için pek çok optimizasyon özellikleri vardır. Bunlara ek olarak data overlays, ve bank switching gibi optimizasyon teknikleri de sağlayabilmektedirler.
- Kullanılan bir diğer araç ise gömülü uygulamaya bir CRC programı eklenmesi böylece gömülü sistem program verisini çalıştırmadan önce kontrol edebilir.

Bunların dışında kullanılan bazı programlama dilleri kullanıcıya gömülü sistemler tasarlamak amacıyla çeşitli kolaylıklar sağlar. C dili için;

- İsimlendirilmiş adres alanları
- İsimlendirilmiş depolama sınıfları
- Temel I/O donanım adresleme

3.4.4 Hata ayıklama (Debug)

Hata ayıklama genellikle bir devre emülatör ya da mikrokontrolör tarafından çalışan mikrokodu kesebilecek bir çeşit hata ayıklayıcı tarafından gerçekleştirilebilir. Mikrokod kesme hata ayıklayıcıya üzerinde sadece CPU'nun çalıştığı donanım üzerinde çalışma imkânı verir. CPU tabanlı hata ayıklayıcılar bilgisayar donanımlarını CPU açısından test etmek ya da hata ayıklamak amacı ile kullanılabilir. Bunun yanında geliştiriciler, yüksek seviyeli dillerle kesme noktası ve tek adımlama kullanarak hata ayıklamalıdır çünkü bu özellikler oldukça yaygın olarak kullanılmaktadır. Bunlara ek olarak basit log kayıtlarının tutulması gerçek zamanlı eylemlerin hata ayıklamasında faydalı olur.

Gömülü sistemin karmaşıklığı arttıkça üst seviye araçlar ve işletim sistemleri tasarımların içine girmeye başlar. Cep telefonları, PDA'lar ve diğer tüketici bilgisayarları ileri düzey yazılım ve işletim sistemleri gerektirir. Bu tip sistemlerde, Linux, NetBSD, OSG'i ya da Embedded Java gibi açık programlama ortamları, geniş bir markete satış yapabilmek amacı ile kullanılmalıdır.

3.4.5 İşletim sistemi

Gömülü sistemlerin genellikle bir işletim sistemleri yoktur ya da özelleştirilmiş gömülü işletim sistemleri kullanılmaktadır.

3.4.6 Başlangıç

Tüm gömülü sistemlerin bir başlangıç kodu vardır. Kesmeleri iptal eder, aygıtları ayağa kaldırır, bilgisayarı test eder (RAM, CPU ve yazılım) ve ardından uygulama kodunu başlatır. Pek çok gömülü sistem kısa süreli güç kayıplarından, en son yapılan testler tekrarlanmadan kurtarılabilir.

Tasarımcılar hataları işaret etmesi açısından genellikle LED kullanımını tercih ederler. Genel bir tasarım olarak tekrar başlatma sırasında tüm cihazlara bağlı LED'ler yanmaya başlayacaktır. Daha sonra açılış testi sırasında yazılım, LED'lerin durumunu hata ya da normal haller için değiştirir.

3.4.7 Yerleşik Self-Test

Kendini test eden gömülü sistemlerin temel şekilleri şöyle açıklanabilir;

- CPU, RAM ve program belleğinin test edilmesi. Bu genellikle sisteme enerji verildiğinde bir kereliğine gerçekleştirilir. Güvenliğin kritik olduğu sistemlerde periyodik olarak ya da belli zaman aşımlarında gerçekleştirilir.
- Giriş, çıkış ve çevre aygıtlarının test edilmesi. Komünikasyon, analog ve kontrol sistemleri bu tip testler gerçekleştirebilir.
- Batarya ya da ona karşılık gelen enerji kaynağının test edilmesi.
- Bağlantı üniteleri arasında gerçekleştirilen iletişim testleri. Üniteler arasında yollanan ve alınan basit mesajlarla sağlanır.
- Kablo testleri. Bağlantı pinlerinin doğruluğunu test etmek için yapılır ve özellikle senkronize iletişim sistemlerinde ön plandadır.

- Donanım arama testi. Eklenen bir donanım ile ilgili bilgiler kullanıcıya döndürülerek sistem yükleme işlemine yönelik yol gösterme sağlanır.
- Tüketilebilinenlerin testi. Sistemde kullanılanların miktarı seviyesi ve pozisyonu hakkında bilgiler döndürür. Yakıt ya da çeşitli kimyasal maddelerin kontrolü buna örnek verilebilir.
- Operasyonel testler. Bu testler sistemin çalışması esnasında yürütülen fonksiyonlara yönelik bir testtir ve sistem çalışırken iken gerçekleştirilir.
- Güvenlik testi. Sistem güvenliği belli bir aralık değerine göre test edilir.

3.4.8 Güvenilirlik rejimleri

İnsanların talep etme nedenlerine göre güvenirliğin farklı tanımları vardır. Ama güvenilirlik tipleri farklı sistemler için temel olup birbirleri arasında büyük değişiklik göstermemektedir;

- Sistem oldukça güvensiz ya da onarım maksadı ile ulaşılması imkânsız. (Uzay sistemleri, denizaltı kabloları… vb.).
- Sistem güvenli bir şekilde kapatılamaz.
- Sistem güvensiz durumdayken çalışamaz.

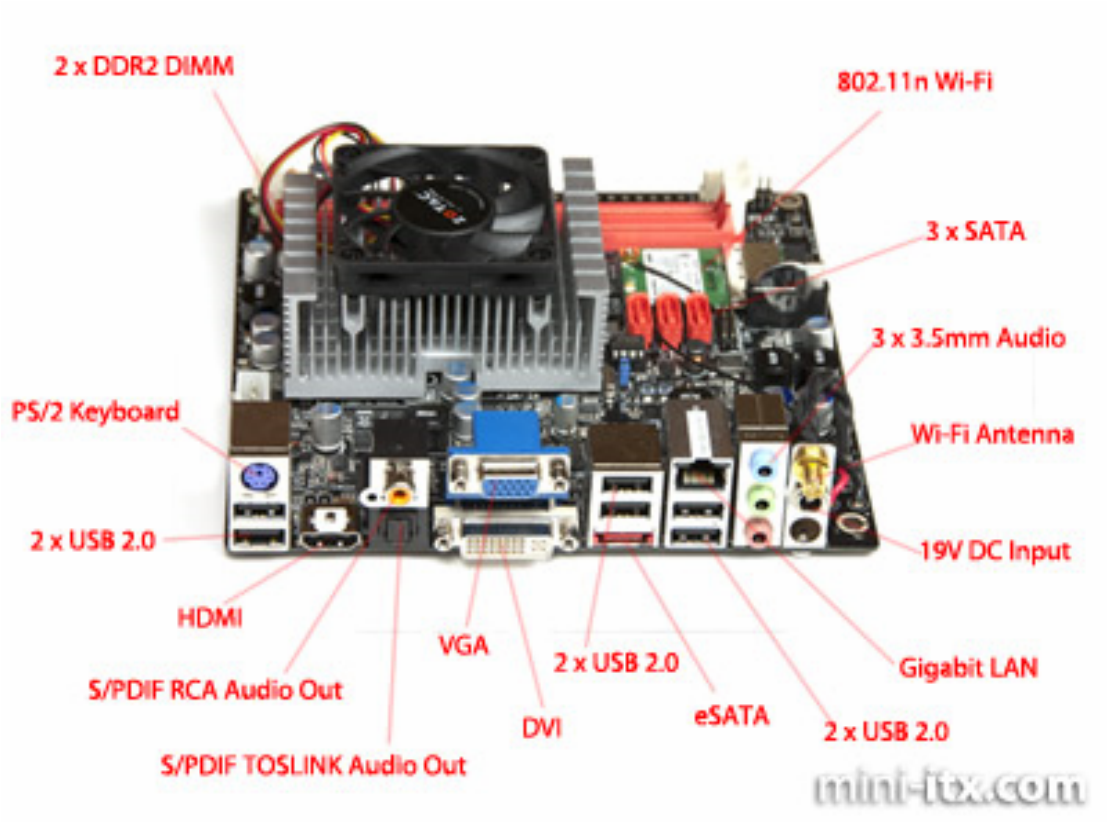
Bu tanımlamalar güvenilirlik rejimi için kriter olarak seçilebilir.

3.5 Gömülü Bilgisayar Sistemi Kartları

Çalışmanın bu bölümünde, gömülü bilgisayar sistemlerinde gelinen son noktada, popüler olan birkaç kartın özelliklerinden bahsedilecektir. Bölüm 8.1 donanım bölümünde, tez çalışmasında kullanılan Lego NXT gömülü bilgisayar sitemi detaylı olarak anlatılacaktır.

3.5.1 Zotac ION-ITX-A gömülü sistemi

Şekil 3.3'de resmi verilen Zotac ION-ITX-A gömülü sistemi, 17 x 17 cm boyutlarında ve bir çok gömülü sistemde tek seferde bulunmayan hemen hemen bütün donanımları bulunduran gelişmiş bir karttır.



Şekil 3.3 Zotac ION-ITX-A gömülü sistemi

Zotac ION-ITX-A gömülü sistemin teknik özellikleri Çizelge 3.1'de verilmiştir.

Çizelge 3.1 Zotac ION-ITX-A gömülü sistemi teknik özellikleri

İşlemci	Intel® Atom™ İşlemci 330 1M Cache, 1.60 GHz, 533 MHz FSB
Chipset	MCP7A-ION
Hafıza	2x DDR2 800 DIMM slotlar (8 GB'a kadar)
Ekran	Onboard NVIDIA® ION (NVIDIA® GeForce 9400M) with NVIDIA® PureVideo™ HD, CUDA™, PhysX™ and nView™ support (2 monitöre kadar)
Ses	5.1 Channel HD Audio
Ağ	1x Gigabit LAN Integrated 802.11n Wi-Fi (Sadece ION-ITX-A için)
Arka Panel Bağlantıları	1 adet PS/2 Klavye portu 6 adet USB 2.0 portları 1 adet HDMI portu 1 adet S/PDIF RCA ses çıkışı 1 adet S/PDIF TOSLINK ses çıkışı 1 adet VGA 1 adet DVI 1 adet eSATA 1 adet Gigabit LAN 1 adet Wi-Fi Antenna (Sadece ION-ITX-A için) 1 adet 19VDC giriş (Sadece ION-ITX-A and C için)
Kart Giriş/Çıkış Bağlantıları	2 adet DDR2 800 DIMM slotlar (Her DIMM için 4GB'a kadar) 3 adet Serial ATA 3Gb/s bağlantıları (RAID 0, 1 and 0+1) 1 adet 10-pin Seri pin bağlantı 1 adet Ön panel bağlantı 1 adet 3-pin CMOS Clear Jumper (Sadece ION-ITX-A ve ION-ITX-C) 1 adet Molex dişi güç bağlantısı (Molex 3x SATA kablosu destekler)

3.5.2 Nano 8050 ITX gömülü sistemi

Şekil 3.4'de görülen Nano 8050 ITX gömülü sistemi, 12 x 12cm boyutunda ve özellikle 3 boyutlu grafik uygulamalarında 2048 x 1536 piksel çözünürlükle yüksek performans sağlayan gelişmiş bir karttır.

Nano 8050 ITX gömülü sistemin teknik özellikleri Çizelge 3.2'de verilmiştir.



Şekil 3.4 Nano 8050 ITX gömülü sistemi

Şekil 3.5'de de Nano 8050 ITX gömülü sistemin arka kart bağlantıları gösterilmiştir.



Şekil 3.5 Nano 8050 ITX gömülü sistemi arka bağlantıları

Çizelge 3.2 Nano 8050 ITX gömülü sistemi teknik özellikleri

İşlemci	Intel® Latest ULV Mobile SFF 45nm Core™ 2 Duo / Celeron® M İşlemci 800/1066Mhz
Chipset	Intel® ICH9M SFF Chipset
Hafıza	1 adet 200 pin SO-DIMM destekli DDR2 800/ 667Mhz (4GB kadar)
Ekran	GMCH Integrated Intel® Graphics Media Accelerator (GMA) 4500 MHD
Ses	Ses çıkışı ve Mikrofon girişi
Ağ	1 adet 10/100/1000BASE-T Ethernet portu
Arka Panel Bağlantıları	1 adet Seçilebilir RS232/422/485 portu 4 adet USB 2.0 portları 1 adet RJ45 portu 1 adet VGA 1 adet ses çıkışı 1 adet mikrofon girişi
Kart Giriş/Çıkış Bağlantıları	2 adet Serial ATA 3Gb/s bağlantıları 2 adet USB 2.0 portları 1 adet 24 bit LVDS bağlantısı 8 bit GPIO pin header

3.5.3 WADE-8656 ITX gömülü sistemi

Şekil 3.6'da görülen WADE–8656 ITX gömülü sistemi, 17 x 17 cm boyutunda ve özellikle simülasyon oyun tarzı uygulama uygulamalarında 2048 x 1536 piksel çözünürlükle yüksek performans sağlayan gelişmiş bir karttır.

Şekil 3.7'de de WADE 8656 ITX gömülü sistemin arka kart bağlantıları gösterilmiştir.



Şekil 3.6 WADE-8656 ITX gömülü sistemi



Şekil 3.7 WADE 8656 ITX gömülü sistemi arka bağlantıları

WADE 8656 ITX gömülü sistemin teknik özellikleri Çizelge 3.3'de verilmiştir.

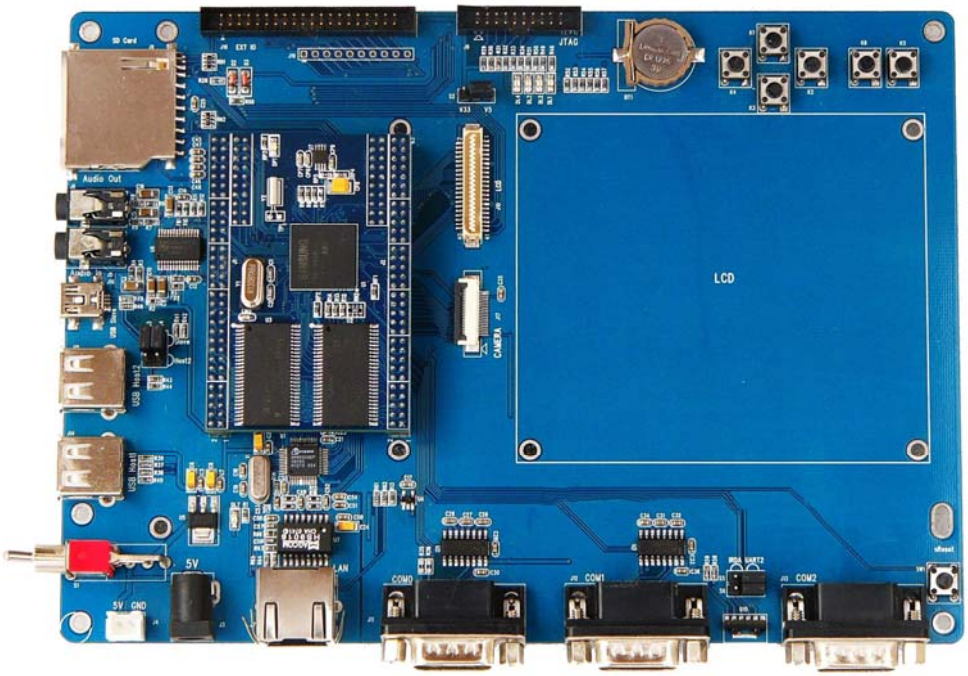
Çizelge 3.3 WADE 8656 ITX gömülü sistemi teknik özellikleri

İşlemci	Intel® Core 2 Quad / Core 2 Duo ve Pentium® 4/Celeron D işlemci
Chipset	Intel® Q965 GMCH & 82801HB ICH8DO
Hafıza	2 adet 240 pin çift kanal DDR2 SDRAMM DIMM 533/ 667/800 Mhz (4GB kadar)
Ekran	Intel® Q965GMCH Integrated Graphics Media Accelerator (GMA) 3000 grafik kartı
Ses	Ses çıkışı ve Mikrofon girişi
Ağ	IEEE 802.3 10/100/1000BASE-T Gigabit Ethernet portu
Arka Panel Bağlantıları	1 adet RS232 portu 1 adet Seçilebilir RS232/422/485 portu 1 adet Klavye portu 1 adet Mouse portu 4 adet USB 2.0 portları 2 adet RJ45 portu 1 adet VGA 1 adet ses çıkışı 1 adet mikrofon girişi
Kart Giriş/Çıkış Bağlantıları	6 adet Serial ATA 3Gb/s bağlantıları 4 adet USB 2.0 portları

3.5.4 SBC2440-IV gömülü sistemi

Şekil 3.8'de görülen SBC2440-IV gömülü sistemi, 6,5 x 4,5 cm boyutunda ve özellikle Windows CE işletim sistemi yüklü olmasından dolayı bir çok Windows uygulamasını destekleyebilmesi sayesinde hiçbir program yazmadan hazır yapılmış uygulamaları da çalıştırabilme yeteneğine sahip gelişmiş bir karttır.

Şekil 3.9'da da görüldüğü gibi SBC2440-IV gömülü sistemin grafik ekranı istenirse kartla beraber gelmektedir.



Şekil 3.8 SBC2440-IV gömülü sistemi



Şekil 3.9 Grafik ekranlı SBC2440-IV gömülü sistemi

SBC2440-IV gömülü sistemin teknik özellikleri Çizelge 3.4'de verilmiştir.

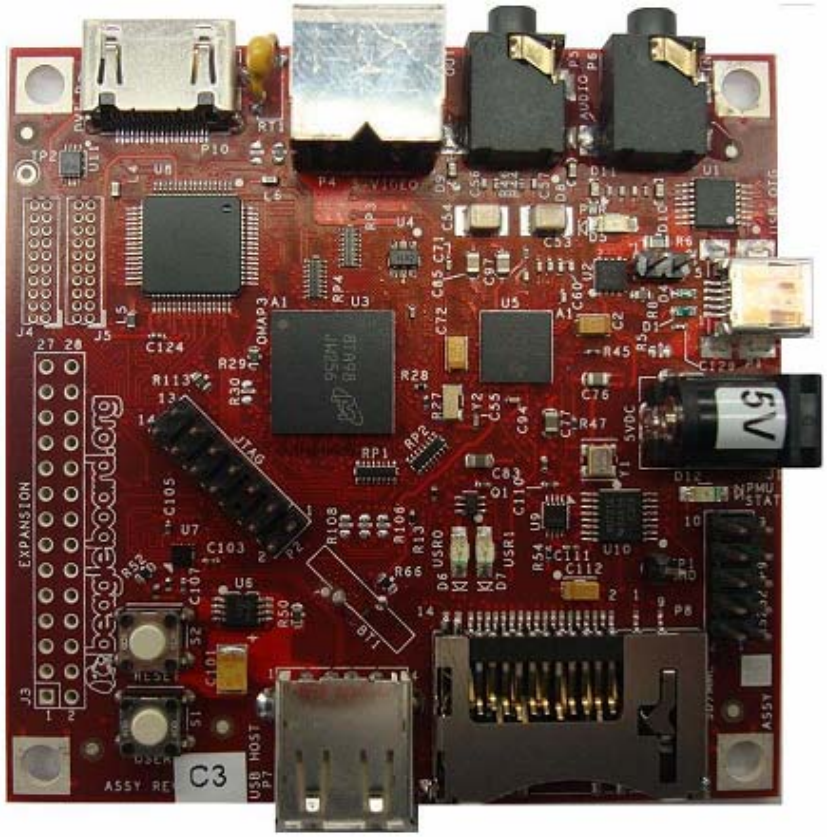
Çizelge 3.4 SBC2440-IV gömülü sistemi teknik özellikleri

İşlemci	Samsung S3C2440, İşlem Frekansı 400MHz @1.33V
Hafıza	64MB SDRAM, 64MB NandFlash
Ekran	7-inch TFT-LCD display module (Optional)
Ses	Ses girişi ve çıkışı slotlar
Ağ	10/100Mbps, RJ45 portu
TFT-LCD	1024 x 768/16-bit, DF9-41S-1V (hirose)
Touch ADC	4 kanal 10-bit ADC
USB portu	2 adet USB portu
Seri port	1 adet beş telli RS232 seri portu, 2 adet üç telli RS232 seri portları
JTAG	20-pin Jtag arayüzü
LED	4 adet programlanabilir led
Buton	6 adet programlanabilir buton
SD kart	1 adet Standard SD kart slotu
IrDA	On-Board kızılötesi
Kamera	8-bit sinyal, FPC bağlantı, 0.5mm pitch
İşletim Sistemi	Windows CE 5.0(BSP), Eboot (tüm kaynak kodları ile), Linux 2.6, uboot, Yaffs dosya sistemi (tüm kaynak kodları ile) işletim sistemlerini destekler.

3.5.5 Beagleboard OMAP 3530 gömülü sistemi

Şekil 3.10'da görülen Beagleboard OMAP 3530 gömülü sistemi, 7,8 x 7,6 cm boyutunda bir karttır. En büyük avantajı kendi özelliklerine yakın diğer gömülü sistemlere göre fiyatının oldukça ucuz olması bu kartı oldukça popüler yapmaktadır.

Beagleboard OMAP 3530 sistemin teknik özellikleri Çizelge 3.5'de verilmiştir.



Şekil 3.10 Beagleboard OMAP 3530 gömülü sistemi

Çizelge 3.5 Beagleboard OMAP 3530 sistemin teknik özellikleri

İşlemci	Texas Instruments OMAP3530 ES3.0
Hafıza	Micron 256MB MDDR ve SDRAM (2GB kadar)
Ekran	DVI-D ve S-Video çıkışları
Ses	3,5 mm L+R Streo çıkışı ve girişi
USB portu	Mini HS USB 2.0 OTG portu
Seri port	UART, McSPI, McBSP, I2C
SD kart	1 adet SD/MMC/SDIO kart slotu
Buton	1 adet ayarlanabilir buton, 1 adet reset butonu

4 ROBOTLARA GENEL BİR BAKIŞ VE MICROSOFT ROBOTICS STUDIO

Robot deyince akla genellikle televizyonlarda veya sinemalarda gördüğümüz tip robotlar gelir. Hatta “Star Wars” serileri veya “Star Trek” gibi bilim - kurgu filmlerindeki robotlar hayalimizde canlanır. Bu örneklerin dışında, robotların, yaşamımıza direk olarak ve pratik bir etkisi olabileceğini düşünmezsiniz ama emin olun ki bütün düşünceleriniz yakın bir gelecekte değişebilir.

Bill Gates bu konuyla ilgili olarak, “Her evde bir robotun olacağı günler pek de uzak değil.” der. Bill Gates “Scientific American” adlı derginin Ocak 2007 sayısındaki bir makalede “1980’lerde PC’lerin önünü nasıl açtıysak, Robot endüstrisini de öyle açacağız.” demiştir. Ve şu şekilde yazar;

“Robotların, günlük hayatımızın birer parçası haline geleceğini görebiliyorum. İnanıyorum ki geliştirilmiş bilgi, ses, görüntü işleme ve kablosuz veri iletimleri gibi teknolojiler sayesinde bizim yerimize kararlar verebilecek yeni nesil otonom cihazlara kapı açılmış olacaktır.”

Bu bölümde sizlere robotların kullanım alanları, hangi aşamalardan geçtiği ve tarihi hakkında yüzeysel bir ön bilgi vereceğim. Ardından, gömülü bilgisayar tabanlı mobil robotumun yazılımını geliştirdiğim Microsoft Robotics Studio programına bir giriş yapacağım.

4.1 Günümüz Robotları

Robotlar, uzun yıllardır üzerinde çalışılan bir alandır. Yapay zeka ile büyük bir bağlantısı vardır ama çoğu insanın gözünde robotlar, otomasyon için kullanılan pratik iş yapabilen makinelerdir.

Robotların kullanımı son birkaç yıldır, insan gücünün çok üzerinde bir çalışma kapasitesine sahip olmaları gerekçesiyle endüstri alanında yaygın olarak

kullanılmıştır. Örneğin otomotiv sektöründe bir robot 10 işçinin iş gücüne eşit iş yapabilmektedir. Dünya pazarında endüstriyel robot piyasası 2005 yılında sivrilmiş ve sonraki yıllarda %30'luk bir artış göstermiştir.

Endüstriyel robotlardan sonra, yeni bir pazar olan servis robotları çıkmıştır. Bu robotlar kişisel ve profesyonel kullanımlar için uygundur. "Roomba iRobot", yani diğer bir deyişle zemin temizleyen robot. Roomba'nın üreticisi aynı robotun değişik versiyonları ile yer temizlemekten bomba tespit ve imha etmeye kadar çok çeşitli tipte ürün üretmektedir.

Profesyonel amaçlı servis robotları, su altı keşif aracı olarak veya koruma, güvenlik, inşaat ve medikal gibi pek çok alanı kapsar. Kişisel kullanımda ise en çok kullanılanlar özellikle çim biçme ve vakum robotlarıdır. Ve bu pazarın önümüzdeki yıllarda çok daha gelişeceği görülmektedir.

Diğer bir gelişen pazar ise eğlence robotları pazarıdır. Şekil 4.1'de "Robosapien" adlı eğlence amaçlı bir robot gösterilmektedir. Robosapien, uzaktan kumandalı olup ayrıca yürüme, hırıldama, geğirme, dans etme gibi 100 farklı program fonksiyonu ile kullanıma sunulmuş bir eğlence robotudur.



Şekil 4.1 Popüler bir oyuncak robot olan "Robosapien"

Uluslararası Robot Federasyonu'ndan elde edilen bilgiye göre, kişisel kullanım için üretilen servis robotlarının sayısı 2006 – 2009 yılları arasında 5,6 milyona ulaşmıştır.

Bunun gibi kişisel kullanım amaçlı robot sektörü gün geçtikçe gelişmeye devam edecektir. Bu durumda üreticiler için robot programlama bilgisine olan ihtiyaç kaçınılmaz olacaktır.

4.2 Robotlardaki Aşamalar

Araştırmacıların en yoğun şekilde üzerinde durmaları gereken konu, sonsuz olasılıklar içerisinde bulunduğu ortama göre uyum sağlayabilen, işlemsel robotlar yapabilmektir. Engellerle dolu bir odadayken, uygun yolu bulup o yönde hareket etmek için beynimizde yaptığımız işlemleri inceleyecek olursak, yüzlerce hatta binlerce spesifik işlemler sayesinde oluştuğunu fark edebiliriz.

Örneğin, bir robotun bir odada karşıdan karşıya düzgün bir şekilde gidebilmesi için öncelikle odada ona engel teşkil edecek nesnelerin farkına varıp bunlara takılmadan hangi yoldan gidebileceğine karar vereceği analizler yapması gerekmektedir. Sandalye ve masa gibi yerinde sabit duran engeller bile fark edilmeleri açısından bir robot işlemcisi için yeterince zor bir işlemdir. Ama bir de robotun karşısında hareketli bir insan olduğunu düşünün. Robot bu kişiye çarptığı zaman anlık bir tepki vermeli ve gidebileceği yeni bir yol seçimini hesaplamalıdır ki bu da robotlar için pek de kolay bir iş değildir. Sensörleriyle yeniden veri toplamalı, işlemeli ve karar vermelidir. Bu durum biz insanlar için oldukça basit ve doğal bir iş gibi görünse de, robotlar için işlemci gücünü en üst düzeyde kullanarak yerine getirebildikleri bir iştir. Her hareket, her işlem ayrı ayrı özel olarak programlanmalıdır. Çünkü robotların kendi kendine düşünme yetenekleri yoktur.

Robotlar, insana benzer vasıflara sahip olmasının güçlüğüne rağmen, yine de bazı işleri oldukça basit bir şekilde yapabilirler. Roomba vakum gibi tekerlekli robotlar kolaylıkla bir odayı gözlemleyip, sensörleri vasıtasıyla tüm engellerin yerini tespit edip, buna göre bir rota izleyebilmektedir. Böcek robotlar ise, engebeli zeminde ilerleme konusunda oldukça ustadır.

4.3 Microsoft'ta Robotlar

Birkaç yıl önce, günümüzün Microsoft Robotics grup başkanı Tandy Trower, robot endüstrisinde neye en çok ihtiyaç olduğunu araştırmak için birçok insanla tanıştı. Akademik ve ticari faaliyetlerde görev yapan araştırmacılarla tanıştı. Bu işi hobi olarak yapan kişilerden ve öğrencilerden elde ettiği bilgiler ışığında, neye en çok ihtiyaç olduğuna dair araştırması sonuçlanmış oldu. Araştırmanın sonucunda şu aşağıdaki maddeler listelendi.

- Sensörler ve eyleyicilerinin kendi aralarında asenkron bir şekilde kolayca çalışabilmesi.
- Dinamik "Başla" ve "Bitir" yazılımı.
- İnteraktif olarak simüle bir şekilde robotun işlemlerini izleyebilme.
- Birden çok insanın tek bir robotu kontrol edebilmesini ve tek bir insanın birden çok robotu kontrol edebilmesini sağlamak
- Aynı yazılımı, değişik robotlar üzerinde yeniden kullanabilmek.

Trower, bu ihtiyaçlara yönelik bir takım düzenlemeler yapıp, robotları pek çok yönüyle inceleyebilecekleri bir yazılım geliştirebileceğini düşündü. Bu sayede robot dünyasının kapılarını pek çok insana açabileceğini düşündü.

4.4 Microsoft Robotics Studio'ya Giriş

Haziran 2006'da Trower ve küçük bir araştırmacı grup bir araya gelerek Microsoft Robotics Studio (MSRS) adında bir programın mümkün olduğunu düşündüler. MSRS, robot camiasında kullanılabilecek doğrudan bir çözümdü. Potansiyel meraklılar ve araştırmacılar için çeşitli engelleri ortadan kaldırabilecek iyi bir platform olacağı umut ediliyordu.

MSRS, Microsoft Windows tabanlı, servise yönelik işletim aşamaları incelenebilen, görsel kullanıcı araçları bulunan, örnekler ve dokümanlarla desteklenmiş bir yazılımdır. MSRS araç kutusu, ticari ve akademik araştırmacılara üzerinde çalışabilecekleri bir platform sunar. MSRS aynı zamanda öğrenciler ve meraklılar için Görsel Programlama Dili (Visual Programming Language) denilen bir

programlama dilini de kullanabilme imkânı sunar. 2006'nın sonlarında 1.0 versiyonu ileri sürüldüğünden beri, Microsoft Robotics grubu üyeleri, asıl büyük versiyon olan MSRS V.1.5'i piyasaya sürdüler. Versiyon 1.5'te özellikle Visual Simulation (Görsel Simülasyon) ve Visual Programming Language (VPL) üzerinde durdular ve geliştirdiler.

MSRS geliştirme grubu oldukça küçük bir grup (Yalnızca 11 araştırmacı) olmasına rağmen, yine de işleri oldukça çabuk çözebiliyorlardı. Ama bu çabukluk da bir yere kadardı. Çünkü MSRS CCR (Concurrency & Coordination Runtime "Eş zamanlı & Koordineli İşlem Takibi") tabanlıydı ve CCR, asenkron programlamayı daha kolay hale getirebilmek için .NET kütüphanesini kullanmaktaydı. Bütün bunlar Trower'ın "Robotlar için daha kullanışlı ne yapabiliriz?" araştırmasının sonucu olarak tesadüfen gelişti.

MSRS'nin son versiyonu olan V.1,5 kullanım açısından kullanıcıya büyük kolaylıklar sunan bir versiyondur. MSRS, şimdiki robot donanımına entegre olabilmek için üçüncü parti donanım üreticileriyle çalışmaktadır.

4.5 Bir Geliştirme Dili Seçmek

MSRS, .NET Framework üzerinde çalıştığından, robot uygulamalarınızı dizayn edebilmeniz için Microsoft Visual Studio kullanmanız gerekmektedir. Burada .NET'in desteklediği herhangi bir programlama dilini kullanabilirsiniz. MSRS ile birlikte gelen örneklerin çoğu C# veya Visual Basic .NET ile yazılmıştır.

MSRS aynı zamanda size açık kaynak kodlu ve objeye yönelik olan Python Scripting Language'ı kullanma imkânı da sunar. Microsoft'un önerdiği .NET tabanlı çalışan Python'lardan biri "Iron Python" dır.

MSRS aynı zamanda size görsel bir programlama dili olan Visual Programming Language'ı (VPL) de sunar. VPL, programlamaya yeni başlayan öğrencilere ve meraklılara, sürükle – bırak teknolojisiyle görselliği sayesinde kolay bir programlama imkânı da sunar. Ayrıca gelişmiş programcılar için de uygulamanın bir taslağını çabucak oluşturmak için bir kolaylık sağlar. VPL ile yapılan bir program, kolaylıkla C# diline çevrilebilir. Bununla birlikte, VPL ile başlanmış bir

program, Visual Studio ve .NET Framework üzerinde değiştirilebilir ve geliştirilebilir.

4.6 Microsoft Robotics Studio'nun Anahtar Bileşenleri

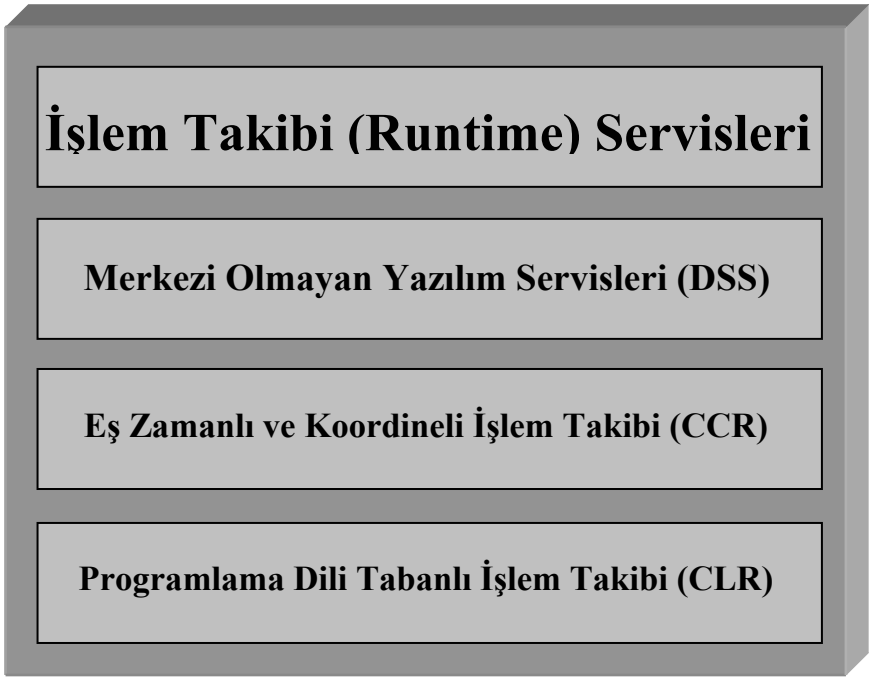
Robot uygulamalarını geliştirmek, düzenlemek ve uygulamak için MSRS'de birçok servis ve araçlar bulunmaktadır. MSRS ile oluşturulmuş uygulamalar bir web ağı üzerinden veya yerel ağ üzerinden çalışabilmektedir. Bu özellik, araştırmacılara web tabanlı robot uygulamaları yapmalarına imkân sağlamaktadır.

4.6.1 Runtime (İşlem / İşletim Takibi)

MSRS'deki işlem takibi (Runtime), Şekil 4.2'de görüldüğü gibi 3 alt dala ayrılır. Bunlar;

- Eş Zamanlı ve Koordineli İşlem Takibi ("Concurrency and Coordination Runtime (CCR)")
- Merkezi Olmayan Yazılım Servisleri ("Decentralized Software Services (DSS)")
- .NET Programlama Dili Tabanlı İşlem Takibi 2.0 ("The .NET Common Language Runtime 2.0 (CLR)")

CCR, asenkron işlemlerden oluşan uygulamaları koordine etmeyi sağlayan mesaja yönelik bir modeldir. DSS, World Wide Web ile Web servisleri mimarisine kombine bir şekilde çalışan, servise yönelik bir modeldir. CLR ise her iki işlem takibi modelini de destekler, ayrıca .NET Framework imkanlarını sunar.



Şekil 4.2 MSRS işlem takibi (Runtime) servisleri

4.6.2 Eş Zamanlı ve Koordineli İşlem Takibi (CCR)

CCR, MSRS işlem takibini servise bağlayan bir arabulucudur. MSRS'de de tanıtıldığı gibi, asenkron işlemli uygulamalar yapabilmemizi sağlar. CCR, asenkron işlemlerde kullanılacak her türlü uygulamayı içeren bir .NET kütüphanesini de kullanıma sunmuş olur. Böylece programcıyı, her bir uygulama için farklı kod yazma zahmetinden kurtarır.

Robot uygulamalarında asenkron işlemlerin niçin bu kadar önem taşıdığı söylenecek olursa, eğer işlemler asenkron olmazsa, robotumuz birden çok, değişik işlemleri aynı anda yapamaz. Sadece bir tekerleğini ya da bir kolunu hareket ettirebilir. Yaptığı bir servise ait işi bitirmeden bir sonrakine geçemez, aynı anda birden fazla servisin işini yerine getiremez.

Robotlar, bulundukları ortamın şartlarını algılayabilecekleri birçok sensörle donatılmışlardır. Aynı zamanda, hareketlerini veya amaçlarını gerçekleştirebilmek için eyleyicileri bulunmaktadır. Bütün bu sensörler ve eyleyicileri, robotu çalıştıracak

olan asıl amaç ile koordineli ve uyumlu olmak zorundadır. İşte CCR, bunu yapabilmek için gereken birbirinden bağımsız işleri yönetir.

Burada kısa bir şekilde sensörlerin robotlarda ne işe yaradıklarını söyleyecek olursak. Robotlar bulundukları ortama ait bilgiyi alabilmek için sensörleri kullanırlar. Bu sensörler adeta robotunuzun gözü, kulağı, teni ve burnu gibidir. Robotunuz için kullanılabilecek düzinelerce çeşitlilikte sensör bulunmaktadır. Çoğu robotta en azından bir adet kontak sensörü bulunur. Bu sensör dokunmaya veya ışığa duyarlı bir modeldir. Ayrıca, robotta bir sıcaklık kayıt sensörü veya bulunduğu yerin tespiti için bir GPS alıcısı gibi kompleks sensörlerde bulunabilir.

CCR'ye geri dönersek, CCR olmasaydı programcılar servisleri koordine edebilmek için çeşitli geri bildirim metotları geliştirmek zorunda kalırlardı, asenkron işlemlerde oluşabilecek aksaklıkları engellemek için karmaşık kodlara gereksinim duyarlardı. Bu durum kesinlikle kullanışsız ve zahmetli olurdu. Burada ise programcı, asenkron işlemleri yönetirken birçok zahmetten kurtulmaktadır.

4.6.3 Merkezi Olmayan Yazılım Servisleri (DSS)

DSS (Decentralized Software Services), programcıların etkileşimli ortamda servisleri izleyebilmelerini sağlar. DSS, Temsili Durum Transfer (Representational State Transfer [REST]) protokolleri üzerine kuruludur. REST aynı zamanda World Wide Web (www) protokollerini de kapsamaktadır. Web ortamında her yeni bir sayfa için gereken bilgiler, istenmeyen ve en son kullanılan başvuru üzerinden gelir. Her yeni istek, bir öncekinden bağımsızdır. Ayrıca .NET Web programcıları değişken oturumlardaki gibi, birbirlerinden bağımsız olan istekler arasındaki bilginin depolanması için servis tabanlı depolar kullanabilirler.

REST, alışılagelmiş bir model değildir ama amaçlarını yerine getirebilmek için standartlar kullanır. REST kaynakların nasıl kullanılacağına ilişkin kurallar içerir ve istenilen bilgiler Web yoluyla iletilir. Bu ve bunun gibi basit kurallar, milyonlarca insanın, interneti her gün basit ve kolayca kullanabilmelerini sağlamaktadır.

Bir robotu uzaktan kumanda edebilmek için, MSRS oldukça karmaşık ve geliştirilmiş bir çözüme ihtiyaç duyar. REST prensiplerinin önemi işte burada ortaya

çıkıyor. Çünkü Web'deki karmaşık transfer işleri, bu prensipler sayesinde basite indirgenebiliyor. REST, oldukça geniş bir kullanım alanı olan HTTP iletişim protokolünü ve beraberinde XML dilini kullanmaktadır. Yapısal XML'e (Structured XML) kolayca dönüştürülebilir.

MSRS aynı zamanda esnek ve geliştirilmiş senaryolara uyum sağlayabilecek bir iletim protokolüne de ihtiyaç duymaktadır. Basit Obje Erişim Protokolü (Simple Object Access Protocol [SOAP]) bunun için en iyi seçimlerden biridir.

Sürekli adı geçen servisin ne olduğu hakkında kısa bir bilgi verecek olursak. Web ortamında ve yerel ağ üzerinde, Tek Biçimli Kaynak Belirleyici (Uniform Resource Identifier [URI]) ile kullanılabilir olan servis, MSRS robot uygulamalarının anahtar objesidir. Bir veya birden çok obje üzerindeki işlemleri çalıştıran, bir çeşit kod topluluğudur. Servisler, robot uygulamalarının temel taşları, bir inşaatın tuğlaları gibidir. Robottan gelen bilgileri alıp, bu bilgilere göre robota ne yapacağını öğreten bilgiler yollayan araçlardır.

MSRS ekibi, kurallarına Web servislerinin kurallarını dâhil etmişlerdir. Bu durum onlara yapısal veri manipülasyonlarında ve olay bildirilerinde esnek çözümler sunar. Web servisleriyle birlikte REST modelini kullanmakla MSRS, farklı makineler arası etkileşimi destekleyen, kaynağı açılabilir ve etkili bir platform sağlar. Her bir servisi diğerinden izole edebilmek ve işin normal gidişatından ayrık durumlar sağlayabilmek için bunu DSS ile kombine ettiğinizde ise, robot uygulamalarında kullanabileceğiniz çok etkili ve güçlü bir araç ortaya çıkmış olur.

Burada bir durumu bildirmek isterim. MSRS internet protokollerini sırf işlem takibini (Runtime) yürütmek amacıyla kullanır. Yoksa bu robot uygulamalarınızı illa ki internet veya yerel ağ üzerinde çalıştırmak zorunda olduğunuz anlamına gelmez. Yani, robot uygulamalarınızın tüm elemanları aynı makine üzerinde yer alabilir. Bir Web Dizayn uzmanı olmanıza veya web siteleri tasarımına başlamanıza gerek yoktur. Burada yalnızca DSS işlem takibinin temelinin bu protokollere dayandığını daha iyi anlayabilmek için verilmiştir.

4.6.4 Merkezi Olmayan Yazılım Servisleri Protokolü (DSSP)

MSRS, DSSP adlı, SOAP tabanlı protokolü önermektedir. MSRS için özel olarak tasarlanmış olan bu DSSP protokolü, HTTP üzerinden erişim sağlar ve mesaj tabanlı işlemlerde alternatif bir metot olarak önerilir. Örneğin MSRS üzerinde çalışan bir HTTP işlemi olan GET'i örnek alalım. Bu işlem, XML sütunundaki gibi, istenilen bilginin çağrılmasını sağlar. Bir servisin, başka bir servisten gelen veriyi elde tutması için, bu verinin SOAP mesajı gibi gönderilmesi gerekir. Bunu sağlayabilmek için MSRS, HTTP-GET işlemini destekler.

Servisleri kullanmanın öncelikli faydalarından biri, birlikte çalışarak ana görevi başarabilmeleridir. Bir servis, bilgiyi başka bir servise iletmek zorundadır. Bunun olabilmesi için, SOAP mesajındaki gibi, her servisin gelen mesajı makine dilinde onaylaması gereklidir.

DSSP, tüm HTTP tabanlı işlemler için, birbirine yakın, karşılaştırılabilir konumlar önerir. Örneğin GET, POST, PUT ve QUERY gibi. DSSP aynı zamanda MSRS'nin ihtiyaç duyduğu fonksiyonları destekleyen özel ek işlemler de sunmaktadır. Örneğin MSRS, üye servise iletilmiş olan olay bildirimlerini isteyen bir işleme ihtiyaç duymaktadır.

HTTP, ABONE (SUBSCRIBE) işlemini desteklemez ancak DSSP destekler. ABONE işlemi MSRS'de desteklenmek üzere özellikle eklenmiştir.

DSSP, HTTP'nin veya Web servislerinin yerini alacak değildir, bu sadece HTTP'nin bir uzantısı, ekidir ve Web servislerinin önerdiği avantajlar sunar.

4.6.5 İşlem Takibi (RunTime) Servisleri

DSS, işlem takibi servisleri diye bilinen altyapı servislerini teklif etmeye olanak sağlayan açılabilir bir taban sağlar. Bu servisler MSRS tarafından oluşturulan diğer servisleri yönetmek için kullanılır. Servisleri yöneten servisler kulağa şaşırtıcı geliyor olabilir ama aslında gerçekleşen de tam olarak budur. Örneğin, HTML tabanlı bir kullanıcı ara yüzünün Kontrol Paneli haline gelmesine olanak sağlayan işlem takibi servisleri vardır. Üzerinde çalıştığınız makinede çalışan tüm servisleri

listeleyen kontrol paneli diğer servislerin yerini tespit edip çalıştırmak için kullanılır. Kısacası her şeyi izleyebildiğimiz ve kontrol edebildiğimiz Web tabanlı bir ortam oluşturur.

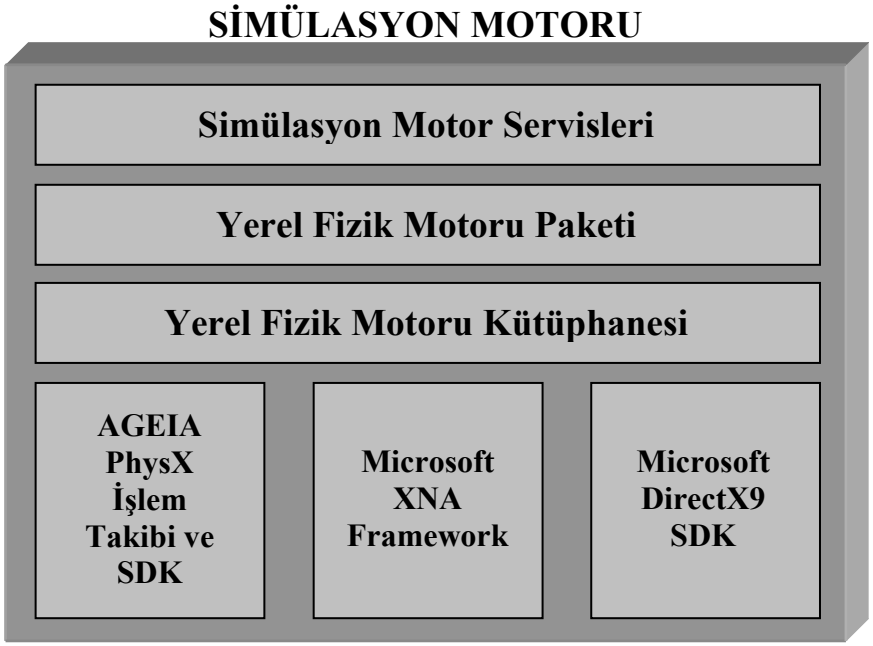
4.7 Görsel Simülasyon Ortamı (Visual Simulation Environment)

Görsel Simülasyon Ortamı (VSE) aracını içinde bulunduran MSRS, programcılara hiçbir robot donanımına gerek olmaksızın robotlar dünyasına bir giriş yolu sağlar. AGEIA PhsyX aracı tarafından güçlendirilmiş olan VSE, marketlerdeki en iyi bilgisayar oyunlarının sahip olduğu grafiklere yakın kalitede gelişmiş fiziksel grafikler sunar. VSE aynı zamanda 3 boyutlu işlemler için Microsoft DirectX 9 işlem takibi bileşenlerini ve Microsoft XNA'yı kullanır. Bu durumda kullandığınız makinenin DirectX 9'un standartlarını desteklemesi gerekir, ancak zaten günümüz bilgisayarlarında bu durumla ilgili bir problem yaşanmamaktadır.

VSE, çok pahalı robotlara para yatıramayacak durumda olanlar, öğrenciler ve meraklılar için robotlar hakkında birçok şeyi öğrenebilecekleri ve geliştirebilecekleri harika bir ortamdır. Genellikle, gelişmiş pahalı donanımlar, bir takımın öğrencilerini ve programcıları yalnızca tek bir robot üzerinde zaman planlaması yaparak çalışmaya zorlar. VSE ile takımın her bir üyesi istedikleri zaman, istedikleri kadar simülasyon aracı vasıtasıyla çalışabilmektedirler. Aynı zamanda istedikleri işlemi birbirlerinden bağımsız bir şekilde doğrudan robot üzerinde uygulayabilmektedirler.

VSE aynı zamanda robot projeleriyle ilgili örnek bir prototip oluşturmak isteyen geliştiriciler için de kullanışlıdır. Programcılar robotu oluşturmak ve yapılandırmak için çokça zaman harcamak yerine, pahalı ekipmanlarına herhangi bir zarar vermeden VSE sayesinde sanal ortamda kodlarını test etme olanağına sahiptirler. Bu sayede testlerin simülasyon ortamında yapılabilmesi maliyeti en aza inmektedir. Kuşkusuz gerçek ortamda robotlar, programcıların hiç ummadığı, beklenmedik engellerle karşılaşır. Bu yüzden projenin başlarında simülasyonun çok büyük faydası vardır. Ama gerçek şu ki, eninde sonunda robotlar gerçek ortam şartlarına boyun eğmek zorundadırlar.

Simülasyon aracı da kendi içinde başlı başına bir servistir. Şekil 4.3'te içerisinde bulunanları gösterilen bu servis, AGEIA PhsyX aracı içindeki yerel fizik araç paketini ve kütüphanesini kullanır. İsteğe bağlı olarak, bilgisayar oyunlarını geliştirmek için özel olarak tasarlanmış olan PhsyX işlemcisi, bazı simülasyonlar sırasında bilgisayarınızın işlemcisi çok yoğun bir şekilde çalıştığı durumlarda lazım olduğunda donanımın ivmelenmesine destek olur. PhsyX işlemcisi olmadan birkaç robot üzerinde simülasyon yapılabilir ancak yüzlerce robotla çalışılması durumunda PhsyX işlemcisine ihtiyaç duyulmaktadır.

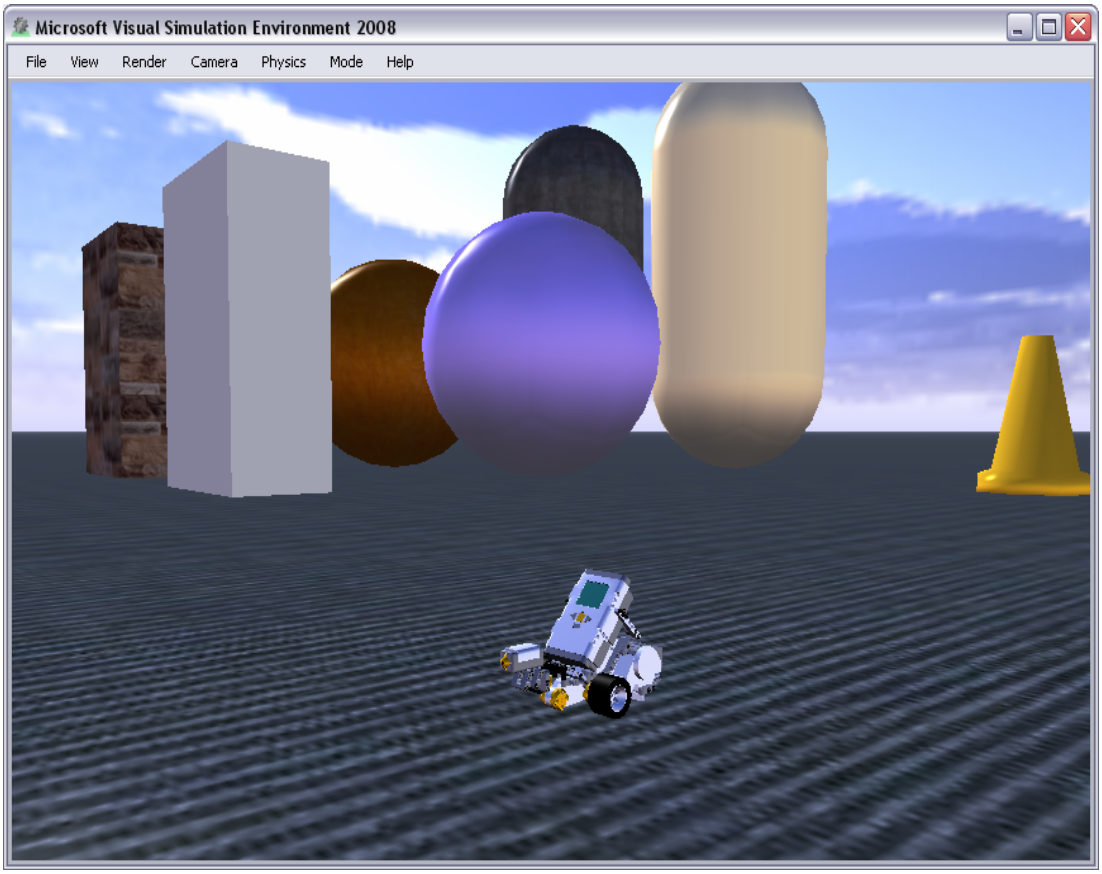


Şekil 4.3 VSE simülasyonu servisleri

Simülasyon takımı aynı zamanda Microsoft XNA Framework ve DirectX 9 Yazılım Geliştirme Kiti (Software Development Kit [SDK]) tabanlıdır. XNA Framework, .NET Framework'e karşı oluşturulmuş, oyun programcılarının C# dilini kullanarak oyunlar üretebilmelerine olanak sağlayan bir kategori kütüphanesi setidir. XNA Framework tarafından kullanılan DirectX 9 işlem takibi geliştiricilere VSE gibi başarılı çoklu ortam uygulamalarını yapabilmelerine olanak sağlar. Eğer üzerinde çalıştığınız makinede bu bahsedilen kütüphaneler yoksa MSRS ilk yükleme sırasında size bu kütüphaneleri yüklemek isteyip istemediğinizi sorar.

VSE bir oluşumu, örneğin bir robot, bir engel, bir kamera, hatta gökyüzü şartları gibi fiziksel objeleri yönetmek amaçlı kullanılabilir. Servisler robotu hareket ettirmek ve sensörlerden gelen verileri anlamlandırmak amaçlı kullanılır. Yönetici (Orchestration) servisler bir grup servisi koordine etmek için kullanılır. Simüle edilmiş ortamda, MSRS işlem takibinde olduğu gibi, aynı servisleri kullanabilirsiniz. Bu özellikler VSE'yi prototip robot uygulamaları için oldukça kullanışlı kılar.

Grafik sahneleri şu dört modelden biri olarak gerçekleştirilebilir; görsel, wireframe, fiziksel veya bu üç modelin tamamının birleşimi. Şekil 4.4'de gösterilen görsel modelde robotunuzun gerçekte nasıl bir görüntüsü olduğunu izleyebilirsiniz. Wireframe modeli sadece objeleri ana hatlarıyla gösterir. Fiziksel model, fizik aracı tarafından modellenmiş objeleri görüntüler.



Şekil 4.4 Görsel model simülasyon sahnesi

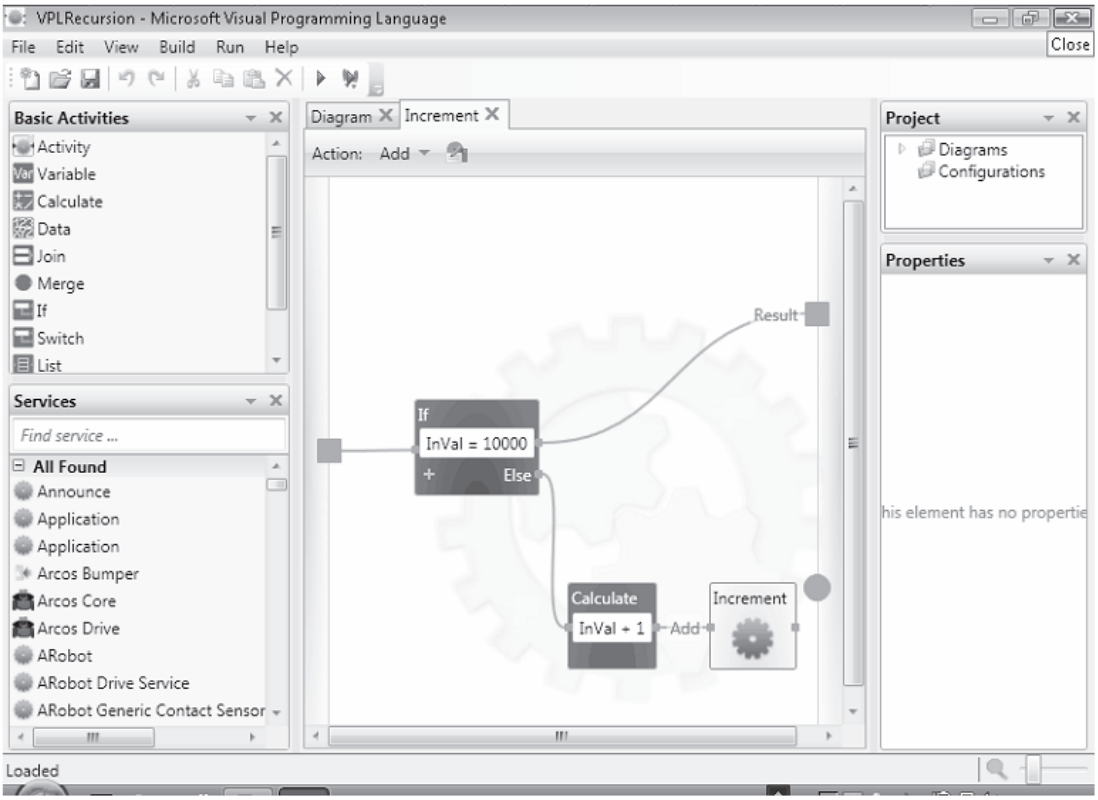
4.8 Görsel Programlama Dili (Visual Programming Language [VPL])

MSRS içerisinde VPL diye kısaltılmış bir görsel kod üreticisi mevcuttur. Visual Studio (Görsel Stüdyo) ile tümleşik olarak gelen VPL, yeni başlayanından uzmanına kadar tüm programcılar tarafından kullanılabilir. VPL'de Kullanıcıların sadece ilgili blokları tasarım yüzeyine sürükleme yöntemiyle istedikleri programı tasarlamaları mümkündür. VPL'de iki çeşit aktivite vardır. Bunlardan ilki, giriş verileri, hesaplamaları ve değişkenleri içeren "Temel Aktiviteleri"dir. İkincisi de, MSRS içinde gömülü olan servisleri içeren "Servis Aktiviteleri"dir. VPL aynı zamanda MSRS'yi çalıştıran işlem takibi servislerini de içerir.

Bağlantı noktaları bir bloğun çıkışının, diğer bir bloğun girişi olarak kullanılmasını belirtmek için kullanılır. Şekil 4.5'te iki blok ve bir bağlantıdan oluşan basit bir program gösterilmiştir. Bu program bir diyalog kutusunda "Merhaba Dünya" (Hello World) yazısının belirmesini sağlar. Bu durumda "Merhaba Dünya" (Hello World) yazısı birinci bloğa yazılmıştır ve birinci bloğun çıkışından ikinci bloğun girişine gönderilir. İkinci blok da bunu karşılayarak sonucu ekrana getirir.

VPL, birçok robot sensöründen gelen verileri işleyebilecek karmaşık ve gelişmiş programlar üretebilmenizi de sağlar. Ayrıca uzaklık ölçüm cihazı ve diferansiyel sürüş sistemine sahip robotlar için otonom sistemler oluşturma amacıyla da kullanılabilir. Deneyimli tüm programcılar MSRS içine gömülü servisler ve VPL ile bir uygulama yapmanın ne kadar pratik ve çabuk olduğunu görebilirler.

Diferansiyel Sürüş Sistemi hakkında kısaca bilgi verecek olursak. Günümüzdeki pek çok robot (özellikle MSRS'nin destekledikleri) tekerlekli yani mobil robotlardır. Tekerlekli robotları işletebilmek için, robotların diferansiyel sürüş sistemini kullanmalısınız. Bu sistem motorun herhangi bir yönde seyrini sağlar. Robotlardaki her teker bir motor tarafından kontrol edilir ve bu motorlar hangi tekere ne kadar döndürme gücü uygulayacaklarını belirleyecek olan değerleri karşılar. Robotunuzun hareketleri iki teker arasındaki dönüş farkına göre belirlenir. İki tekeri motorlu ve üçünü tekeri serbest olan bir mobil robot sistemi için her iki tekere aynı oranda döndürme uygulanırsa robot ileri yönde gidecektir. Eğer motorlardan birisine diğerine oranla daha az güç uygulanırsa robot o yönde dönme hareketi yapacaktır.



Şekil 4.5 VPL'de tasarım ortamı

4.9 Tipik Bir Robot Uygulaması Neye Benzer?

Birçok robot uygulaması, robotun bir veya birden çok sensöründen gelen bilgileri toplayıp, bunları anlamlı bir şekilde yorumlayıp, bu datalara uygun sonuçlar ortaya çıkarabilme temeli üzerinde çalışmaktadır. Örneğin bir ses sensörü ile çalışan bir robot uygulaması yapabilirsiniz. Ses sensörleri farklı desibel değerlerindeki sesleri yakalayıp, kaynaktan gelen yüksek veya alçak sesleri ayırt edebilmektedir. MSRS ile robotunuzun hareketlerini el çırpma veya komutu yüksek sesle söyleme şeklinde yönlendirebilirsiniz. Aslında robotunuz, bu seslere göre ne yapması gerektiğini anlayamaz, ancak belirli desibel değerleri aralığında ne yapması gerektiğini daha önceden robota bildirmişseniz, robotunuz o aralıklardaki seslere karşı ilgili tepkileri verebilir.

MSRS programları ihtiyaç duyulduğu ölçüde basit veya karmaşık programlar olabilirler. Aslında bu programlar bir robot donanımına gereksinim duymak zorunda

değildir. Öyle bir servis oluşturabilirsiniz ki, sadece giriş bilgileri üzerinde işlem yapıp, bir çıkış datası üretebilir. Örneğin "Merhaba Dünya" yazısı gösteren örnekte "Merhaba Dünya" yazılı bir satır, ilgili servisin girişine gelir, bu serviste işlenir, web tarayıcınızın ekranında görüntülenebilecek şekilde XML mesajı olarak çıkacaktır. Bu örnekte de olduğu gibi bir robot donanımı olmadan da MSRS'de program yazılabilir.

Bir robot uygulaması hazırlamak için gereken zaman pek çok şeye bağlıdır. Seçtiğiniz robotun yapısının karmaşıklığı, kullandığınız robot sayısı, robotunuzun fonksiyonel yapısı bunlara etkendir. Örneğin komutu aldığı zaman sadece ileri yönde hareket edecek basit bir robotunuz varsa, bununla ilgili komutları ve uygulamayı hazırlamanız, bir grup futbol maçı yapacak olan robotlarla ilgili uygulamayı hazırlamaktan çok daha kısa sürecektir.

4.10 MSRS'nin Desteklediği Robotlar

4.10.1 Desteklenen robotların ve çözüm ortaklarının listesi

Bu konu içerisinde MSRS desteklediği ve MSRS programına destek veren tüm profesyonel robot üreticilerinin bir listesini verilmiştir. Bu bilginin tez çalışması içerisinde verilmesinin en büyük nedeni hem bu programa destek veren dünyaca ünlü profesyonel robot üreticilerinin ne kadar çok olduğunu ve MSRS'ye ne kadar önem verdiklerini hem de MSRS programının ne kadar güçlü ve kullanılan bir program olduğunu göstermektir. Çizelge 4.1'de MSRS'nin desteklediği tüm profesyonel robot üreticilerini ve ulaşabileceğiniz güncel Web sayfalarını bulabilirsiniz. MSRS programına destek veren robot firmalarının ve çözüm ortaklarının sayısı her geçen gün hızla artmaktadır. Bu da MSRS'nin ilerleyen zamanlarda tüm robotik uygulama alanlarında vazgeçilmez olduğunu göstermektedir.

Çizelge 4.1 MSRS'nin Desteklediği Tüm Robot Üreticileri

aldebaran-robotics.com	arobotineveryhome.com	allmotion.com	barrett.com
braintech.com	cerebellum-automation.com	conscious-robots.com	coroware.com
diversity.co.uk	drrobot.com	ed.co.kr	elementdirect.com
energid.com	fischertechnik.de	graupner-robotics.de	robotics.co.kr
hanulkid.com	hokuyo-aut.jp	icoptech.com	irobot.com

 jadi.us	 itkmc.com	 kuka.com	 mindstorms.lego.com
 lynxmotion.com	 mostitech.com	 parallax.com	 phidgets.com
 resquared.com	 ridgesoft.com	 robai.com	 roborealm.com
 robodynamics.com	 robosoft.fr	 robotever.com	 robo3.com
 roboticsconnection.com	 roboware.co.kr	 scatterweb.net	 segway.com
 senseta.com	 simplysim.net	 skilligent.com	 sri.com

surveyor.com	toradex.com	triindia.co.in	via.com.tw
visualcomponents.com	wanyrobotics.com	whiteboxrobotics.com	yujinrobot.com

Ayrıca MSRS programının geliştirilmesinde çözüm ortağı olan birkaç firmanın bilgilerine ve güncel Web sayfalarına da Çizelge 4.2'den ulaşılabilir.

Çizelge 4.2 MSRS'nin Çözüm Ortakları

camelot.dk	lgcns.com	lntinfotech.com	mobotsoft.com
ologicinc.com	rchordata.com	sharplogic.com	

Bu başlık içerisinde MSRS'nin desteklediği üç tip robot ile ilgili biraz bilgi verilecektir. Bunlardan birincisi Şekil 4.6'da gösterilen iRobot tarafından üretilmiş olan "Create" robotudur. Create robot, hiçbir montaja ihtiyaç duymayan bir robottur.

MSRS tarafından desteklenen bazı robotlar iRobot Create gibi ön düzenlemeleri yapılmış şekilde gelir ve çok az bir düzenleme gerektirir.



Şekil 4.6 iRobot firmasının ürettiği “Create” isimli robot

MSRS’nin desteklediği robotlardan ikincisi ise Şekil 4.7’de gösterilen Parallax firması tarafından üretilmiş olan “Boe-Bot” robotudur. Boe-Bot robot, çok az montaj gerektirir. Büyüme ve açılım için geniş bir imkân sahası vardır.



Şekil 4.7 Parallax firmasının ürettiği Boe-Bot robotu

Ve son olarak bahsedilecek robot ise tez çalışmamda da kullanılan, Şekil 4.8'de de gösterilen LEGO Mindstorms NXT robotudur. Mindstorms NXT robotu, şaşırtıcı derecede güçlü ve kullanışlı robotlardır. Gömülü bilgisayar sistemi, çok kullanışlı sensörleri ve Bluetooth teknolojisi içerir. Diğer robotlara göre gerekli düzenlemeler için daha fazla zaman ayırmak gerekir ama robot donanımının nasıl çalıştığını daha iyi öğrenmek için mükemmel bir robottur.



Şekil 4.8 LEGO Mindstorms NXT robotu

Bu üç robota, küçük, kullanışlı ve çalıştırması kolay robot tipleri oldukları için bilgi verme adına tez içerisinde yer verilmiştir. Bu, MSRS'yi endüstriyel robot sektöründe veya karmaşık senaryolar gerektiren donanımlarda kullanamayacağımız anlamına gelmez. Burada amaç robotik dünyasına ilk adımı atacaklar için veya MSRS programı ile ilk kez tanışacaklar için pratik yapma ve öğrenme aşamalarında yardımcı olması için bu tez çalışmasında yer verilmiştir. Eğer henüz robotlar konusunda yeniyseniz, Create sizin için ideal bir robot olabilir. Küçük elektronik uygulamalarda az da olsa deneyim sahibiyseniz, Boe-Bot, LEGO Mindstorms NXT, Fishertechnik sizin için daha iyi bir seçim olabilir. Ciddi uygulamalar için, Kondo KHR-1 veya Pioneer 3DX kullanılabilir. Endüstriyel robot yapımcısı KUKA

tarafından geliştirilmiş robot kollar da vardır. 6 eklemli kola sahip bu robot ile ilgili pek çok örnek KUKA'nın web sitesinde bulunabilir.

Önemli bir konu da şudur ki, MSRS sadece meraklılar ve öğrenciler için kullanılan bir program değildir. Dünya üzerinde kullanılmakta olan oldukça karmaşık ve profesyonel robotlar için bile MSRS kullanılabilir.

4.10.2 Desteklenmeyen donanımlarla çalışmak

MSRS bize, desteklediği robotlarla ilgili ön düzenlemeleri yapılmış servislerle birlikte gelir. Ancak diyelim ki biz MSRS'nin desteklemediği bir donanım üzerinde çalışmak ve kendi servislerimizi kendiniz oluşturmak istiyoruz. Desteklenmeyen robotlar veya cihazlar, sensörlerden gelen dataları ve motoru kontrol edebilmek için uzaktan yönetilen bir iletişim ara yüzüne ihtiyaç duyarlar. Eğer donanımımızda bu ara yüz yoksa robotumuz veya cihazımızla birlikte çalışan bir kod platformunda bir ara yüz oluşturmak zorundayız. Bunun içinde iyi derecede C# veya Visual Basic .NET programlarını bilmek zorundayız.

4.11 Robot Programlama Konusunda Kısa Bir Bilgi

Visual Studio ve .NET Framework'e yatkın olan .NET geliştiricileri, en hızlı kullanımı MSRS'de bulabilirler. Aslında robot programlaması, diğer normal programlama çeşitlerinden ve özellikle Web programlamasından oldukça farklıdır. Eşzamansız işlemcilerle biraz da olsa çalışmış olan programcılar bile, robot uygulamalarında gereken birçok eşzamansız görevi kontrol etmeleri olanaksızdır.

.NET Framework ile gelişen Web teknolojisi bir veya birden çok kullanıcının aynı anda bir web sayfasına girebilmelerini ve bilgi alabilmelerini sağlar. Bilgi her ne kadar dinamik ve ilişkisel veritabanından beslense de, işlemler genellikle ardışıktır ve bir işlem tamamlanmadan diğer işlem başlamaz. Robot programlarında ise, birçok sensörün bilgiyi aynı anda alıp işlemesi gereklidir. Serbestçe dolaşabilecek bir robotta, otonom davranışları tanımlayabilmek için, sensörlerden gelen bilgileri

okuyup robota ne yapacağını söyleyen kodların yazılmış olması gereklidir. Tüm bu aktiviteler aynı zamanda gerçekleşmelidir.

5 SERVİSLER

Servisler, MSRS ile yapılan robot uygulamalarının temel yapı taşlarıdır. Tezimin bu bölümünde özellikle servislerin ne işe yaradığını ve robot sensörlerinin kontrolünün servisler tarafından nasıl gerçekleştirildiği anlatılacaktır. Alınan ve gönderilen mesajların kullanımının servisler tarafından nasıl kontrol edildiğini, ayrıca servislerin diğer servislerle nasıl iletişim kurdukları anlatılacaktır. Bölümün sonunda servislerin, harici Dinamik-Link Kütüphane (Dynamic-Link Library [DLL]) dosyası gibi gömülü kaynakları ve çıkışları nasıl kontrol ettiği anlatılacaktır.

5.1 Robot Uygulamalarının Tanımı

Üzerinde çalışılacak robot, bulunduğu ortama ait çevre bilgilerini algılayabilen sensörlere sahip olan fiziksel bir cihazdır. Sensörler, ışığın tonu veya robotun başka bir cisme olan uzaklığı gibi ölçülebilir şeyleri algılayabilmek için kullanılan cihazlardır. Sensörler, herhangi bir programlama yapılmadığı sürece kullanışsızdırlar. Alınan verinin işlenip bir sonuca ulaşabileceği küçük bir yazılım mutlaka olmalıdır.

Sensörlere ek olarak, robotlar aynı zamanda eyleyici (actuators) diye bilinen çıkış cihazlarına da sahiptir. Küçük bir hoparlör veya bir LED diyot “eyleyici (actuators)” olarak düşünülebilir. Eyleyicileri, robotların sensörler tarafından algıladıkları ortama bağlı bilgilere göre cevap vermelerini sağlar.

Örneğin, küçük ve tekerlekli bir robotun tampon sensörü olsun ve bu sensörden gelen bilgiyi işleyebilen bir program oluşturulur. Bu durumda robot ilerlerken karşısına çıkan bir objeye çarptığı zaman tampon sensöründen gelen bilgiyi işleme alarak, robotun bir cisme çarptığını robota bu yolla anlatmış oluruz. Programa yapılacak ufak tefek ilavelerle, robotun böyle bir durumla karşılaşınca dışarıya ses veya ışık olarak cevap vermesi de sağlanabilir. Verilen bu örnekte, tampon sensöründen gelen bilgi “giriş”, objeye çarptıktan sonra dışarıya cevap verme niteliğindeki ses veya ışık da “çıkış” olarak düşünülebilir.

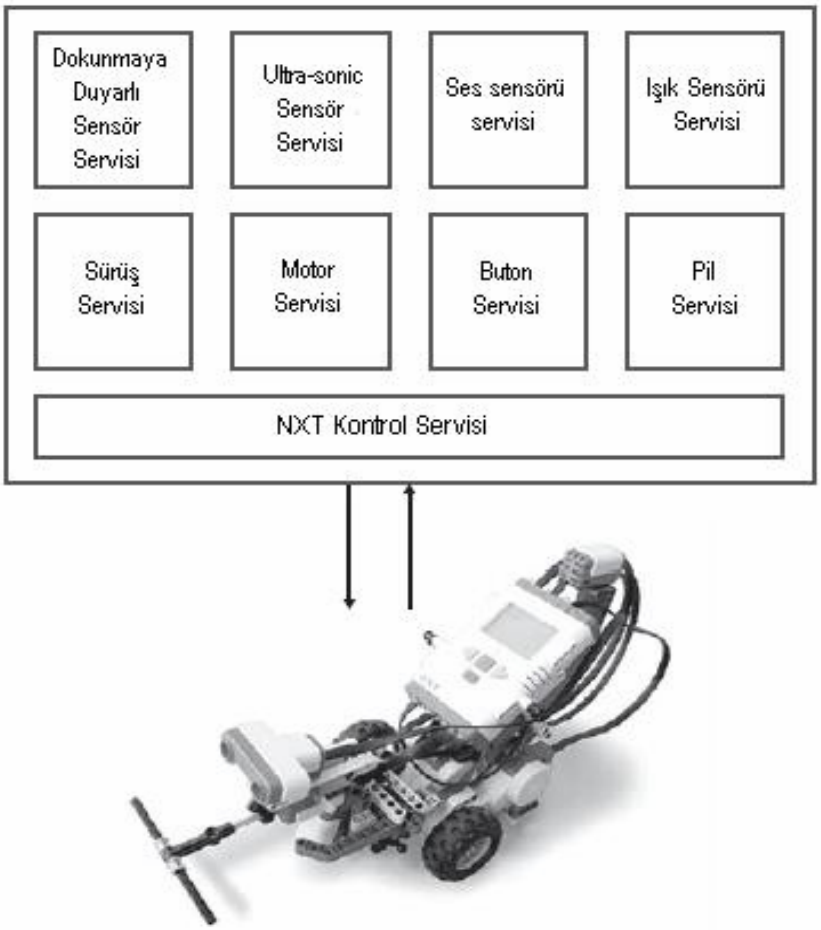
Robot uygulamaları, giriş bilgisini alıp, işleyip, bir çıkış veren diğer başka uygulamalarla benzerlik gösterir. Ancak robot uygulamalarını benzersiz kılan özellik de şudur ki, robotlar birden çok giriş bilgisini aynı anda alıp, işleyip, çeşitli çıkışlar verebilir. Birçok sensör ve eyleyici tek bir robotta bulunabilir. Bir sensör bilgileri toplarken aynı zamanda eyleyiciler çıkış verebilmektedir. Ancak robotun böylesine bir cevap verme kapasitesine sahip olabilmesi için bütün bunların eş zamanlı olarak yani aynı anda gerçekleşmesi gereklidir.

5.2 Servis Nedir

Robot uygulamalarında gereken özel ihtiyaçları karşılayabilmek için, MSRS anahtar obje olarak servisleri kullanır. Her web tabanlı servis, tek bir sensörden gelen veriyi okumak veya eyleyiciye bir çıkış sinyali göndermek gibi bir veya birden fazla fonksiyonu yerine getirebilmek için kodlara ihtiyaç duyar. Servisler, diğer servislerle veya harici yazılımlarla iletişim kurmak amacıyla da kullanılmaktadır. Bir robot uygulaması genel olarak robotun, misyonunu yerine getirmesini sağlamak için birlikte çalışan birçok servisten oluşur. Örneğin Şekil 5.1'de tezimizde kullandığımız LEGO Mindstorms NXT'yi çalıştırmak için kullanılan servisleri gösterilmiştir.

5.3 Yeni Bir Servis Oluşturmak

MSRS, DSS (Merkezi Olmayan Yazılım Servisleri) işlem takibi servisleri arasındaki etkileşimi sağlayan pek çok komut satırı aracını sizin kullanımınıza sunar. Çizelge 5.1'de DSS ile çalışan komutlar verilmiştir. MSRS V.1.5 ve daha üzeri versiyonlar piyasaya çıkmadan önce, geliştiriciler manüel olarak "DSSNewServise" komut satırı aracını kullanarak yeni bir servis oluşturmak zorundaydılar. Şimdi Şekil 5.2'de gösterildiği gibi MSRS bizlere Visual Studio modelini sunuyor. Bu model bize Visual Studio 2005 ve daha üstü versiyonlar ile yeni servisler oluşturma imkânı sağlıyor.

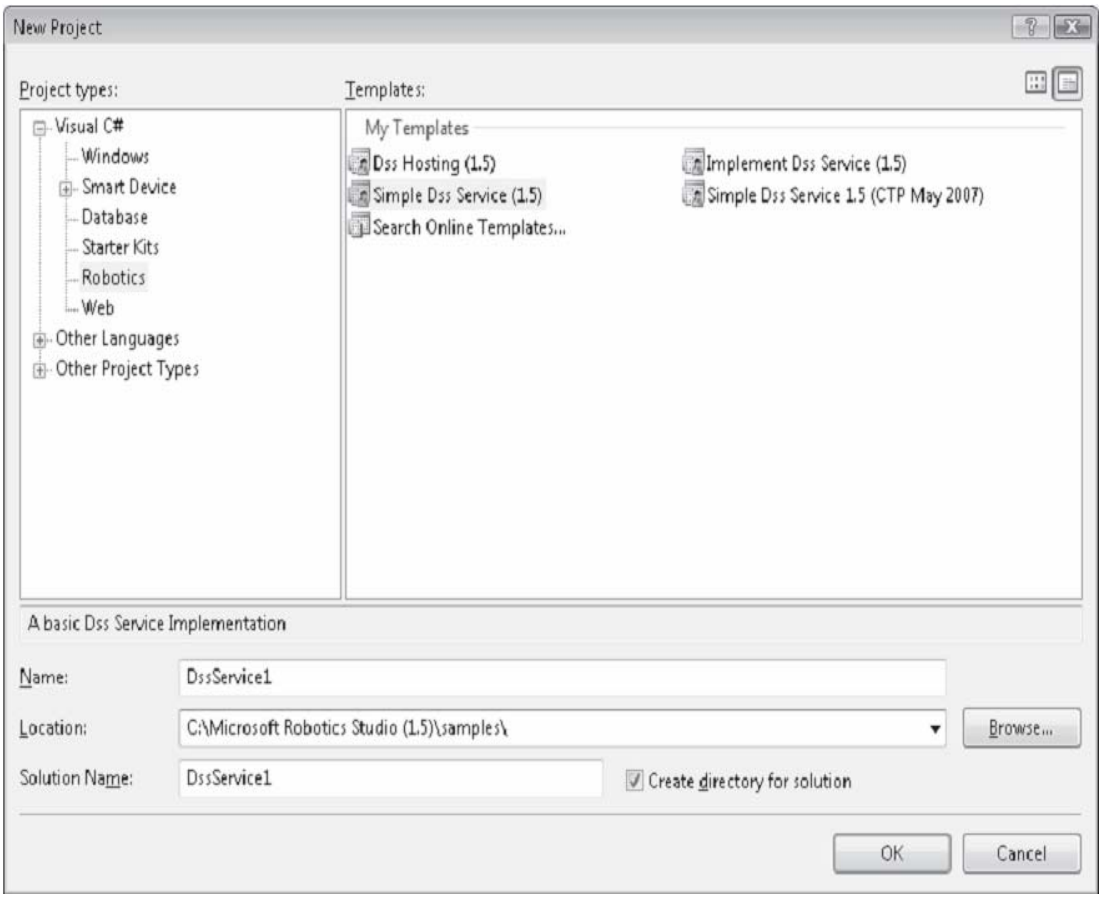


Şekil 5.1 LEGO Mindstorms NXT'yi çalıştıran servisler

Çizelge 5.1 DSS ile çalışan komut satırı araçları

ADI	AÇIKLAMA
DssHost	DSS modülünü port üzerinde başlatmak ve bir veya birden fazla servisi çalıştırmak için kullanılır.
DssNewService	Yeni bir servisin iskeletini oluşturmak için kullanılır. Visual Studio çözümlerini ve proje dosyalarını içerir. Proje oluşturma aşamasında programlama yolunda çeşitli alternatifler sunar. Yeni bir DSS servisi oluşturmanın diğer bir yolu da MSRS'nin sağladığı Visual Studio modelini kullanmaktır.
DssInfo	Servis hakkında bilgi almak için kullanılır.
DssProxy	Proxy'leri ve dönüşümleri üretmek için kullanılır. Proxy, ürünün kontrat bilgisini açığa çıkarmaya yarar. Sıklıkla doğrudan çağrılır. Yeni bir servis oluştururken, işlem aşamalarının bir parçası olarak

	kullanılır.
DssProjectMigration	Projenin versiyonlar arası geçişinde kullanılır.
DssDeploy	Tüm dosyaları bir servisle giden, yayılım için kullanılabilir olan, işlem yapılabilir şekilde takımlar oluşturmada kullanılır.



Şekil 5.2 DSS Servislerin tasarlandığı Visual Studio programı

5.4 Bir Servise Başlamak

Yeni bir servis oluşturmak için gereken kodları gözden geçirmeden önce bir servise nasıl başlanacağını anlatacağım. "DSSHost.exe" komut satırı aracı bir port üzerinde yeni bir DSS modülüne başlamak ve bir veya birden çok servisi çalıştırmak için kullanılır. Çizelge 5.2'de DSSHost komut satırı aracında kullanılan parametre

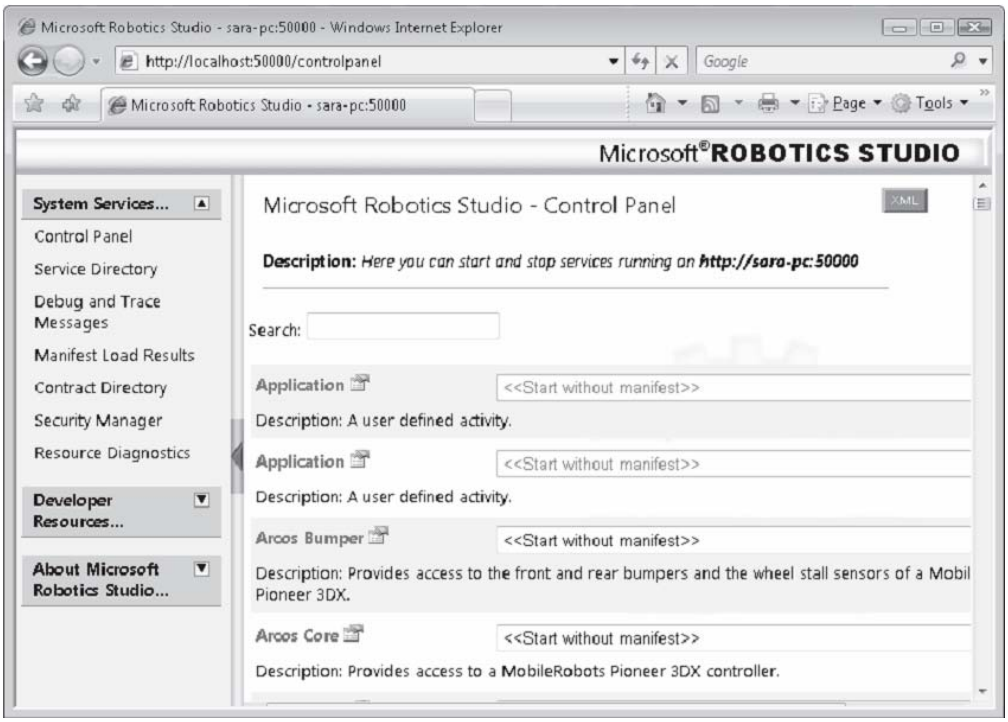
listesi verilmiştir. DSS modülünü kullanarak bir port üzerinden çalışacak olan başka bir servisin çalışmasından önce DSS modülü başlatılmalıdır. DSS nodülünü başlatmanın en kolay yolu BAŞLAT'tan Microsoft Robotics Studio altındaki "Run DSS Node" öğesini tıklayarak çalıştırılabilir.

Çizelge 5.2 İşletilebilir DSSHost parametre listeleri

UZUN ADI	KISA ADI	AÇIKLAMASI
/manifest	/m	En son kullanılan manifest(ler)i yüklemeyi işaret eden URI veya yol.
/dll	/d	En son kullanılan DSS Servis DLL'yi yüklemeyi işaret eden URI veya yol.
/contract	/c	En son kullanılan DSS contract'ı yüklemeyi işaret eden URI veya yol.
/port	/p	HTTP isteklerini karşılayan İletişim protokolü (Transmission Control Protocol "TCP") portunu içeren tam sayı değeri. Bu portun numarası yaygın olarak 50000'dir, böylece bilinen diğer portlarla çakışma olmaz.
/tcpport	/t	SOAP isteklerine karşılayan İletişim protokolü (Transmission Control Protocol "TCP") portunu içeren tam sayı değeri. Bu portun numarası yaygın olarak 50001'dir, böylece bilinen diğer portlarla çakışma olmaz.
/hostname	/h	Host ile özdeşleşmiş olan host adını içeren dizi (string) değer.
/security	/s	Güvenlik ayar dosyasına işaret eden URI veya yol.
/verbosity	/v	Sözcük seviyesi. Off/Error/Warning/Info/Verbose olabilir. Info fabrika ayarı değeridir.
@file	@file	Argümanların listesini içeren metin dosyasının adıdır. Aynı argümanları tekrar tekrar yazmak gerektiğinde, iyi bir alternatiftir.

DSSNode'u çalıştırdıktan sonra karşımıza komut istemcisi ekranı gelir ve Internet tarayıcımız yandaki URL ile açılır: http://localhost:50000. Internet tarayıcımız Şekil 5.3'de görülen MSRS'nin Sistem Servisi Web Sayfasını (System Services Web Page) açacaktır. Buradan robotta kullanılan servislerin bir listesini görmek için ekranın solundaki panelde görülen Kontrol Panel'e tıklanabilir. Çalışan servisleri görmek için yine sol paneldeki Servis Dizini'ne (Service Directory)

tıklanarak ulaşılabilir. Ekrana gelen komut istemcisi penceresini kapatılmaz. Çünkü eğer kapatılırsa DSS modülü durur ve bir daha açana kadar servisler yönetilmez.



Şekil 5.3 MSRS Sistem Servisi Web Sayfası

HTTP'nin 50000 portunu, SOAP'ın da 50001 portunu kullanarak, aşağıdaki komutu yazarak MSRS dosyalarınızın kayıtlı olduğu ana dizinin yerini belirterek, DSS modülünü manüel olarak başlatmak da alternatif bir yoldur.

```
Dsshost  -p:50000 -t:50001
```

Bu komut, manifest veya servisler yüklenmeden DSS modülünü başlatır. Manifest, başlatılacak servislerin listesini içeren bir Açılabilir Biçimleme Dili (Extansible Markup Language "XML") dosyasıdır. Yaygın olarak DSS modülü başlatılırken kullanılır. Manifest, Tek Biçimli Kaynak Belirleyiciye (Uniform Resource Identifier "URI") geçiş veya DSSHost'un çalıştırılabilmesi için gerekli kaynak yolu olarak belirtilebilir. Contract, DSS Servislerin başka bir servisi kullanmak için ihtiyaç duyduğu bilgileri içerir. Contract, HTML dosyasıyla

birleşiktir ama bu dosyaya doğrudan erişilmez. Bunun yerine, contract'ın içeriğini listelemek için DssInfo komut satırı aracını kullanılabilir.

5.5 Servis Projesi Oluşturmak

Yeni bir servis oluşturmak için Visual Studio 2005 veya daha üst bir versiyon açılır. Dosya menüsünden Yeni'ye ve ardından Proje'ye tıklanır. Şekil 5.2'deki "Yeni Proje" diyalog kutusunda tercih ettiğiniz lisan için ilgili modülü genişletilir, Robotics modülü seçilir ve Örnek DSS Servis 1.5 Proje Şablonu'nu (Simple DSS Service) seçilir. Daha sonra isim kutusuna yeni servise vermek istediğiniz adı yazılır (veya otomatik olarak yazılmış olan DssService1 adı onaylanabilir). Kod dosyaları için kayıt yapılacak hedef belirlenir ve Tamam'a tıklanır. Bu işlem örnek bir DSS Servisi oluşturmak için ihtiyaç duyulacak dosya ve çözümleri sağlar. Bu dosyalar sınıf dosyaları ve XML manifest dosyası içerir.

Servis adı olarak otomatik belirlenen adı seçtiğinizi kabul edecek olursak, sınıf dosyalarınız "DssService1.cs" ve "DssService1Types.cs" olarak adlandırılmış olacaktır. "DssService1.cs", yürütme sınıfıdır ve sensörlerden gelen bilgiyi okuma ve robota komut gönderme işlevlerini yürütür. DssService1Types.cs ise, contract sınıfıdır ve servisin durum bilgisinin geri bildirimini sağlar. Contract sınıfı, aynı zamanda bir servisin oluşturulmasındaki istekleri de ele alır.

Durum servislerini, kullanışlı bilgiler içeren aktif dokümanlar olarak düşünebiliriz. Bu dokümanları açabilir ve ne gibi bilgiler içerdiğini görebiliriz. Bunu yaptığımız zaman, istenen servisin sunumuna ilişkin servis durum bilgisine erişmiş olursunuz. Örneğin herhangi bir sensörden veri dönüşü sağlayan bir servisimiz varsa, bu servisin durum bilgisi, olay anında sensörden gelen veriyi okuyor durumudur.

Hiçbir ilave kod gerekmeksizin, Visual Studio şablonu ile bir DssServis Projesi oluşturabilir. Varsayılan ayarlardaki bir servisin durum bilgisi boştur. Dolayısıyla hiçbir cihazla haberleşmez veya durum bilgisi dönüşü olmaz.

Eğer basit bir şeyler yapabilen bir DssServis Projesi oluşturmak istersek, "Merhaba Dünya" dönüşünü sağlayan servisi düşünebiliriz. Burada "Merhaba Dünya" yazısı, servisin durum bilgisidir ve bu bilgi servise erişildiği zaman

görüntülenir. Bunu daha iyi anlamak ve uygulamak için "DssService1Types.cs" dosyasının içine aşağıdaki kodu yazmanız gerekir.

```
private string _outputmsg = "Merhaba Dunya";
[DataMember]
public string OutputMsg
{
   get { return _outputmsg; }
   set { _outputmsg = value; }
}
```

"DataMember" tanımı kamuya aitliğe işaret eder (XML gibi sıralanmış olan). Sıralama (Serialization), bir .NET objesinin sunum SOAP mesajında olduğu gibi, byte akışını sağlayan bir işlemdir. Buna ek olarak, .NET objesini örneklemek için SOAP mesajından bir veri almak "deserialization" olarak bilinir.

DssService1State için sınıf tanımlaması DataContract'a işaret eder. Bu da bir servis için üretilen, DSS Proxy dll içinde bulunan DSSService1State ile birlikte olan objelere işaret eder.

Bunu daha iyi anlamak için, "Merhaba Dünya" DssServis Projesiyle ilgili SOAP cevap mesajına bakalım. Bunu görmek için başarılı bir şekilde DssServis Projesini F5'e basarak çalıştırmak gereklidir. Bu işlem projemizi inşa etmemizi ve MSRS bin klasörü içerisinde assembly oluşumunu sağlar. Varsayılan olarak, Visual Studio şablon proje özellikleri aynı zamanda DSS Proxy assembly oluşturmak için DssProxy'yi çalıştırır. Sonuç olarak projenin onarım özellikleri modülü başlatmak ve DssServis'i yüklemek için DssHost'u çalıştırır. DSS Proxy, bir servisin sunumu için assembly kullanır. Servisler kendi aralarında doğrudan haberleşmek yerine, bu Proxy üzerinden haberleşirler. Proxy, servis derlendiği zaman üretilir.

Servisi çalıştırmak için F5'e bastıktan sonra, komut istemcisinden çıkıp, internet tarayıcınızı açıp, servisiniz için gerekli URL girilir. URL şu şekilde olmalıdır:

```
http://localhost:50000/DssService1
```

Açılan SOAP mesajı aşağıdakine benzer olacaktır:

```
<s:Envelope xmlns:s=http://www.w3.org/2003/05/soap-envelope
xmlns:wsa=http://schemas.xmlsoap.org/ws/2004/08/addressing
```

```
xmlns:d="http://schemas.microsoft.com/xw/2004/10/dssp.html">
<s:Header>
   <wsa:To>http://127.0.0.1:1153/</wsa:To>
   <wsa:Action>
http://schemas.microsoft.com/xw/2004/10/dssp.html:GetRespons
e
   </wsa:Action>
   <d:TimeStamp>
   <d:Value>2007-07-10T13:16:28.0938784-05:00</d:Value>
   </d:TimeStamp>
   <wsa:RelatesTo>uuid:99cf78b1-dabf-4b7d-8c6b-
14124cbfc023</wsa:Relates>
   </s:Header>
<s:Body>
<DssService1State
xmlns="http://schemas.tempuri.org/2007/07/dssservice1.html">
   <OutputMsg>Merhaba Dunya</OutputMsg>
</DssService1State>
</s:Body>
</s:Envelope>
```

Bu "DssService1State" adlı SOAP mesaj elemanı gövdesine bakarsak. Bu elemanla XML isim-alanı (namespace) birleşimi, servisin contract ismi olarak nitelendirilebilir. Bu isim-alanı (namespace) basit anlamda, contract'ın eşsiz kimliğidir. Geçerli bir URL'deki gibidir ama aktif bir dosyanın hedef yolu olmayı gerektirmez.

5.6 Mesajları Göndermek ve Karşılamak

Tezde buraya kadar geçen konuların en sonunda "Merhaba Dünya" satır durum bilgisine sahip olan basit bir servis oluşturuldu. Bu işin sonunda ilgili satır DSSPort'una aktarıldı. Bir port üzerindeki DSS işlemini yerine getiren mesajlar, diğer servislerin servis durumunu güncellemelerini ve okumalarını olanaklı kılar. Son örnekte "Merhaba Dünya" servisinin durum bilgisi sunuldu. Bu satır porta gönderildi ve sonuç bir SOAP mesajı olarak internet tarayıcısında görüntülendi.

DSS portu, servislerin ileri veya geri yönlü haberleşmesine olanak sağlar. Bir servis, bir veya birden çok portu belirtebilir ve her port, o porta gönderilmiş mesajları

tutan verilere sahiptir. Özel bir tutma pozisyonu belirleyebilmek için dâhili bir port belirlemek mümkündür.

5.6.1 Bir mesajın anatomisi

Her servisin temel fonksiyonu mesaj göndermek ve almaktır. Servislerin birbirleriyle iletişim kurmasını sağlayan da budur. Mesajlar güçlükle yazılır ve mesaj tipleri servisin yapısına göre değişiklik gösterir. DSS için, bir mesaj aşağıdaki üç etkeni içermelidir:

- **Aksiyon (Action)**: Yaygın olarak, işlem takibi esnasında gerçekleşen işlem hakkında bilgi veren bir terimdir. Örneğin "Get" terimi, servisin en son durumunu çağırır. "Update" terimi, servisin en son durumunu günceller.
- **Gövde (Body)**: İsteğin gövdesini oluşturur. Bir yazının gönderilmesini ifade etmez. Örneğin aşağıdaki XML GetRequest teriminin gövdesidir:

```
[DataContract]
[XmlType(Namespace =
"http://schemas.microsoft.com/xw/2004/10/dssp.html")]
[XmlRoot("GetRequest", Namespace =
"http://schemas.microsoft.com/xw/2004/10/dssp.html" ,
IsNullable = false)]
```

- **Yanıtlama Portu (Response Port)**: Gönderilene cevap verme portudur. Eğer belirtilmişse, cevap aksiyon sonucu başarılı veya hatalı, ne olursa olsun, rasgele gönderilebilir.

"Merhaba Dünya" satırı DSS transform dosyası vasıtasıyla getirilir. Transform dosyası servis proxy'nin bir parçasıdır ve servisin assembly kodu derlenince üretilir. Bir servisin durum öğeleri servise geri dönmeden önce çağrılara gider. Transform, DSS Proxy üretildiğinde oluşturulan bir assembly'dir. Proxy, servis contract'larını meydana çıkarır ve transform da servis ile Proxy arasında adres bulmayı sağlar. Servis ve Proxy assembly'leri arasında bir köprü vazifesi görür.

Visual Studio şablonunu kullanıldığında, Proxy ve transform dosyaları kendiliğinden oluşturulur. Nereden çağrıldığına bakmak için "Project" menüsü

içerisinden “Project” > “name” seçenekleri seçilir. “Build” > “Events” sekmesine tıklanır ve ardından “DssProxy” ile komut satırı görüntülenir. Varsayılan olarak, oluşturulan assembly’ler, MSRS yüklemesi için var olan bin klasörü içerisinde depolanır.

Bir servisi derlerken, MSRS üç assembly üretir: Bunlardan biri servis için, biri Proxy için, ve biri de transform içindir. Bunlar aşağıdaki kurala göre isimlendirilir:

<service name>.Y<year>.M<month>.< Proxy, Transform veya blank>.dll

Örneğin aşağıdaki üç dosya DssService1 adlı servis derlendiği zaman üretilir.

- C:\Microsoft Robotics Studio (1.5)\bin\DssService1.Y2009M06.dll
- C:\Microsoft Robotics Studio (1.5)\bin\DssService1.Y2009M06Proxy.dll
- C:\Microsoft Robotics Studio (1.5)\bin\DssService1.Y2009M06Transform.dll

5.6.2 PortSet kullanımı

MSRS’nin mesajları DSSPort üzerinden nasıl alıp gönderdiğini anlamak için, Visual Studio şablonu ile oluşturulmuş bir servisin ürettiği aşağıdaki kodlara bakalım. DssService1.cs dosyası içinde ana port ifade edilmiştir.

```
[ServicePort("/DSSService1", AllowMultipleInstances = true)]
DSSService1Operations _mainPort = new
DSSService1Operations();
```

Bu kodlarda gösterilmeye çalışılan şey, ana port birçok örneğe izin vermez ve servis de “dssservice1” adlı dizin içerisinde yer edinmiştir. Buna göre, servisin URL’sini “http://localhost:50000/dssservice1” şeklinde tanımlayabiliriz. Eğer port birçok örneğe izin veriyorsa, her örnek Globally Unique Identifier (GUID) ile birbiri ardına eklenecektir.

Bu kodlardan anlaşılan bir diğer husus da, bu port üzerinde DssService1Operations sınıfında belirtilen DSSP işlemleri yerine getirebilir. DSSP’nin SOAP tabanlı bir protokol olduğunu ve DSS tarafından mesajları ele

almak için kullanıldığını hatırlarsak. “DssService1Operations” sınıfı “DssService1Types.cs” dosyasının içindedir ve aşağıdaki gibidir:

```
[ServicePort]
public class DSSService1Operations :
PortSet<DsspDefaultLookup, DsspDefaultDrop, Get>
{}
```

DssService1Operations sınıfı PortSet’i tanımlar ve bu PortSet ne tür DSSP ve http mesajlara izin verildiğini anlamak için kullanılır. Tüm servisler DsspDefaultLookup işlemine ihtiyaç duyar. “Lookup” mesajı, bir servisle nasıl haberleşileceği ile ilgili bilgiyi içeren servis context’ini geri çağırmak için kullanılır. “DsspDefaultDrop” mesajı buna gerek duymaz ama servisin Drop mesajları desteklemesine izin verir. Sonuç olarak, “Get” mesajı bir servisin kendisine ait o anki durumuyla cevap vermesine izin verir. Get, birçok DSSP işleminden yalnızca biridir. Durum, DSSP’nin de ihtiyaç duyduğu gibi, SOAP mesajı biçiminde geri döner. Durumu, XML olarak geri döndürmek için PortSet listesine, “HttpGet” şeklinde bir http işlemini eklemeniz gerekir. Çizelge 5.3’de DSSP işlemleri verilmiştir.

Çizelge 5.3 MSRS’nin sunduğu DSSP işlemleri

İŞLEM	**VARSAYILAN UYGULAMA**	**AÇIKLAMASI**
Create	DsspDefaultCreate	Yeni bir servis oluşturur. Direkt olarak çağırmak durumunda değilsiniz. Çünkü varsayılan servis yapıcı DsspServiceCreationPort tarafından ele alınmıştır.
Delete	DsspDefaultDelete	Delete işleminin gerçekleştiği durum parçasını siler. Sadece durumu siler, servisin kendisini değil.
Drop	DsspDefaultDrop	Servisi kapatır. Servise gönderilen en son mesaj olmalıdır.
Get	DsspDefaultGet	Servisin tüm durumlarını almak için kullanılır. Eğer durum, pek çok öğe içeriyorsa, hepsi geri döner. Özel öğeler için Query işlemini kullanmalısınız.
Insert	DsspDefaultInsert	Request’in içerdiği durum servis durumunun sonuna eklenir.
Lookup	DsspDefaultLookup	Servisi geri çağırır. Tüm mesajlar için bu işlem gereklidir.

Query	No default provided	Özel parametre tabanlı istekleri baz alan durumlara erişir.
Replace	DsspDefaultReplace	Tüm elementleri servis durumu içinde yerleştirir.
Submit	No default provided	Talimatları işletmekle aynıdır, servis durumunu değiştirmeden işlemleri yerine getirir.
Subscribe	No default provided	Bir serisin, servis durumu değişiklikleri hakkında sonuç bilgilerini karşılamasına izin verir.
Update	DsspDefaultUpdate	Durumun güncellenecek bir parçasını belirlemek için kullanılır.Update isteği, delete'i ve insert'i aktif hale getirir.
Upsert	No default provided	Insert ve Update'in birleşimidir. Eğer o anda durum içeriyorsa, durum güncellenir. Aksi halde, durum insert'lenir.

DsspOperation sınıfı tarafından üretilen kendi mesajlarımızı da ekleyebiliriz. Yazdığınız servislerin çoğu en azından bir tane özel işlem uygulaması içerecektir. Çizelge 5.4'te gösterildiği gibi DSSP işlemlerine yapabileceğiniz bazı ilavelerle http tabanlı işlemleri de kullanabiliriz.

Çizelge 5.4 HTTP işlemleri

İŞLEM	**DURUM DEĞİŞİKLİĞİ? (YES / NO)**	**AÇIKLAMA**
httpDelete	Yes	İstenilen durum öğesini silmek için kullanılır.
httpGet	No	URL kullanan durum geri döner. Çalışması, bir web tarayıcısının herhangi bir dokümana ulaşması gibidir.
httpOptions	No	Haberleşme seçenekleri hakkındaki bilgiler için istek sunar.
HttpPost	Yes	Genellikle HTML form data işlerinde kullanılır.
httpPut	Yes	İsteğin içinde bulunan data ile yerleştirilen servis durumunun isteğidir.
httpQuery	No	HttpGet gibidir ama Query'de parametreler URI olarak geçer.

5.6.3 Servis işleyicileri kullanmak

Servis işleyici, port üzerindeki gelen mesajları işlemekle yükümlüdür. Bir servis, birden çok servis işleyiciye sahip olabilir ve her biri farklı DP, HTTP, veya özel işlemlere mahsustur. Yalnızca DsspDefaultLookup ve DsspDefaultDrop işlemleri farklılık gösterir çünkü bunlar DSS Runtime ile işlenir. Ama gerektiğinde servis tarafında üzerine yazma işlemi yapılabilir.

Yeni bir DSS servis oluşturmak için kullanılan Visual Studio şablonu, sadece tek bir servis işleyici sağlar. Aşağıda gösterilen servis işleyici DSSP Get mesajlarını işlemek için kullanılır:

```
[ServiceHandler(ServiceHandlerBehavior.Concurrent)]
public virtual IEnumerator<ITask> GetHandler(Get get)
{
   get.ResponsePort.Post(_state);
   yield break;
}
```

"GetHandler" durumu SOAP mesajı olarak yanıt portuna kaydeder. Bu işleyicideki kod, hesaplamaları veya diğer servislere mesaj gönderimini işletebilir.

Her servis işleyici, bunun nasıl işleneceğini belirlemeye olanak sağlar. Örneğin, "GetHandler" eşzamanlı çalışmaya kurulmuştur. Bu çalışma şekli, durum bilgisini düzenlemeyen işleyiciler için de kurulabilir. Durumu düzenleyen işleyiciler için, "Exclusive" servis işleyicisini kullanabilir. Diğer nitelikler olarak, derlendikten sonra servisi kapatan "Teardown" ve işlem tipine bağlı durumları seçen DsspOperationDefault verilebilir.

5.7 Durum Yönetimi

Tezimin "Mesajları göndermek ve almak" başlıklı bölümünde geçen, mesajları işlemek için gereken temel kod dizilimi verildi. Mesajlar istekler için veya durumla iş alışverişi için kullanılır. Bazen servisle tümleşik olan durumu değiştirmek gerekebilir. Tezimin bu konusu altında durum yönetme yollarını anlatacağım.

5.7.1 Dâhili port'a veri nakli

Tez çalışmasının son bölümünde, servisin durumuna ulaşmak için "Get" işlemi kullanıldı. Internet tarayıcısında SOAP mesajı olarak ortaya çıkan servis durumu ana porta aktarılmıştı. Buna alternatif olarak durum bilgisi, dâhili port'a da aktarılabilir. İstek olana kadar, verilerin biriktirilmesi veya kaydedilmesi gereken durumlarda kullanışlı bir alternatiftir. Her veri, durumun özel tutuş versiyonunda depolanabilir. Duruma, servis çalıştığı sürece bakılır.

Bunun nasıl çalıştığını anlamak için, gelen her http isteği ile ilgili olan portu görüntülemekle sorumlu olan devamlı işlem servisleri dikkate alınır. Bir istek karşılandığında, bu istekle ilgili bilgi mesaj dâhili port'ta geçici olarak saklanır. Durum bilgisi, kullanıcının veya başka bir servisin izni olmadan ana port'a aktarılmaz.

Veriyi dâhili port'a aktarmak için, durum değişkenleri belirlenir. Aynı zamanda ilave port da belirlenir. Aşağıda port adı MyInternalPort olan bir portun oluşturulması için gereken kod verilmiştir.

```
Port<string> MyInternalPort = new Port<string>();
```

Daha sonra, "Update Generic" sınıfı tarafından elde edilecek bir mesaj oluşturulur. Aynı zamanda oluşturulan işlem tipi için, yeni bir işlem tipi ve servis işleyici oluşturulur. Bu yeni işlem tipi, ana PortSet tarafından desteklenen işlemler listesine eklenecektir. Örneğin GetURL adlı işlem tarafından desteklenen bir servis aşağıdaki gibi bir ana işlem portuna sahip olabilir:

```
[ServicePort()]
public class MonitorRequestsOperations : PortSet
      <DssDefaultLookup,DsspDefaultDrop, Get, GetURL>
{}
```

Aynı zamanda, yeni bir mesaj için eşsiz bir mesaj işleyici eklemenize ihtiyaç vardır. Bu kodu http portundaki istekleri kontrol etmek için eklemelisiniz. Veriyi dâhili porta aktarmak için, CCR Arbiter sınıfından yararlanmanız gerekir. Gelen istekler ele alınırken oluşabilecek çakışmaların önüne geçmek için CCR gereklidir.

Arbiter-Recieve metodu tek öğe alıcı oluşturmak için kullanılır. Alıcı kontrolü için CCR, "DpServiceBase.Activate()" metodunu çağırmaya ihtiyaç duyar. Örneğin aşağıdaki kod, bir satırı MyInternalPort isimli port'a aktarmak için kullanılır:

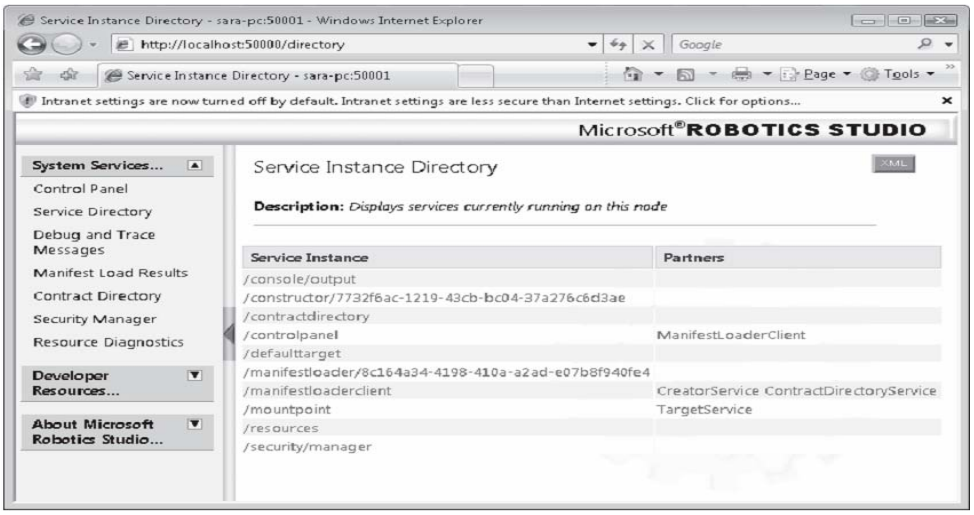
```
MyInternalPort.Post(message);
Activate(Arbiter.Receive(true, MyInternalPort,
                                    MyInternalPortHandler));
```

5.7.2 Durumu Bir Dosyaya Kaydetmek

Bazı durumlarda, servisin durumunu bir dosyaya kaydetmek faydalı olabilir. Böylece daha sonra istenildiğinde tekrar geri dönülebilir. Ayarları bir konfigürasyon dosyası olarak kaydetmek istediğinizde kullanışlı olabilir. Durum, ısrarla Initial State Partner kullanmak isteyebilir. Partner, Partner Attribute kullanılarak bildirilir. Servisler bir veya birden çok partnere sahip olabilir.

Partner, başka bir servise bağlanan bir servis sunar. Partnerlikler servisler arasındaki ilişki ile ilgili MSRS'yi bilgilendirir. ConstructorService gibi bazı partnerlikler servisinizi derlediğiniz zaman otomatik olarak oluşturulurlar.

Şekil 5.4'de System Services Web sayfasındaki Service dizininden hangi servisin hangi partnerle bağlantılı olduğunu gösterilmiştir.



Şekil 5.4 Service Instance Directory'de çalışan servis ve ilişkili partnerlerin listesi

- **ConstructorService:** Create işlemini kullanan servisler de yeni bir servisi ilişkilendirmek için kullanılır.
- **PartnerListService:** Partner Liste yönetici servisi geçerli partnerleri geri çağırmak için kullanılır. Bu partner tüm servisler için listelenecektir.
- **DurumServisi:** Durumu yönetmek gerektiğinde bu servis kullanılır. Eğer durum sadece geri çağrılan ve kaydedilmeyen bir durumsa, bu servisin partnerler listesinde bulunması gerekmez.

DSS, özel bir tip olan Initial State Partner'ı kullanıma sunar. Bu partner tipi, durum dokümanından erişilen değerleri baz alan durumun başlangıç değerini ayarlamak için kullanılır. Örneğin aşağıdaki kod, durumu MyState.Config.xml adlı bir dosyaya kaydedecek olan partneri belirlemek üzere InitialStatePartner'ı kullanır.

```
[InitialStatePartner(Optional = true, ServiceUri =
                                        "MyState.Config.xml")]
private MyServiceState _state = new MyServiceState()
```

Bu kodda servis, MyState.Config.xml dosyasını içermese bile yine de çalışabilir. Çünkü partner isteğe bağlı bir değişkenle belirlendi. Durum dokümanı, MSRS "Installation" klasörünün altındaki "Store" dizini içinde yerleşmiştir.

Eğer durum hala içermiyorsa, "Start()" metot kodu eklenir. Bu metot, servis her çalıştırıldığında işleme alınır. Eğer durum dokümanı yoksa durum değişkenleri için başlangıç değerleri ayarlanmalıdır. Örneğin aşağıdaki "Start()" metodu, önceden oluşturulan "DssService1" için durumu sıfırlamada kullanılabilir:

```
protected override void Start()
{
   //Check to see if the state already exists.
   //If not, then we will initialize it
   if (_state == null)
   {
      _state = new DssService1State();
      _state.Message = "";
   }

   //Start the service
   base.Start();
}
```

Yukarıdaki örnekte tek bir durum değişkenine değinilir. Çünkü yalnızca tek bir değişken "DataMember" olarak işaretlendi. Normalde bir servis pek çok değişken içerebilir.

Durumu kaydetmek için, durumu MyService1.Config.xml dosyasına kaydeden kodu eklemek gerekir. "SaveState()" metodu DSS servis modelinin bir parçasıdır ve aşağıdaki .NET ile birlikte servis uygulamaları ile uyumludur.

```
using Microsoft.Dss.ServiceModel.DssServiceBase;
```

DspServiceBase'de servis uygulamaları için base class'ı sunan taban referansını kullanarak metotlara başvurur. Durumu kaydeden kod, durumun değerlerini değiştiren kodların bulunduğu alana eklenebilir. Böylece aşağıdaki örnekte olduğu gibi; SaveState() metodu, taban referansının içinden çağırabilir:

```
base.SaveState(_state);
```

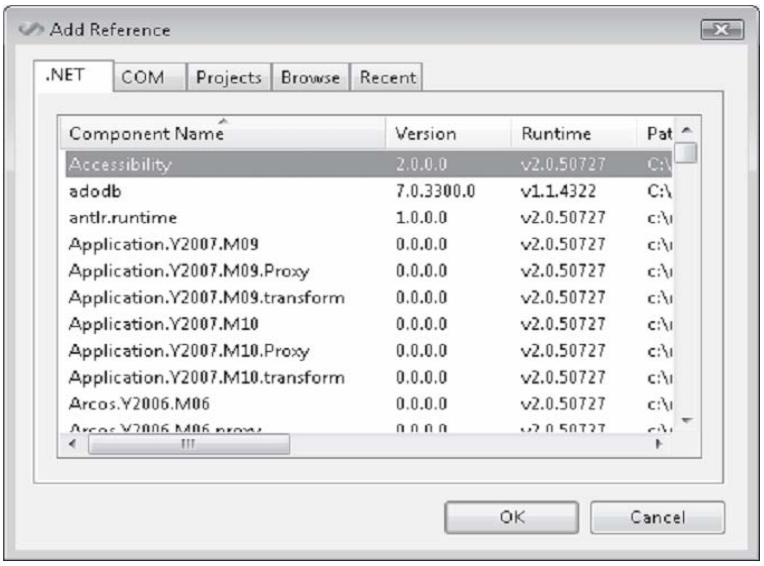
Durum bilgisini kaydetmek, servise gelen çağrılar arasında durum bilgisine erişmek ihtiyacı doğduğunda kullanışlı olmaktadır.

5.8 Diğer Servislerle Haberleşme

Partnerlikleri kullanırken, bir servis başka bir servisten metot çağırabilir veya durum geri dönüşümü sağlayabilir. Bir servis örneğin DssService2 diğer bir servisle örneğin DssService1 arasındaki ilişkiyi yorumlayabilmek üzere partner kullanabilir. Bu işi yerine getirebilmek için referans'ın DssService2 projesine eklenmesi gereklidir.

Referansın başka bir servise eklenmesiyle, Proxy asembly'nin yerini Şekil 5.5'de gösterildiği gibi araştırılmalıdır. DssService2 uygulama sınıfının diğer bir adı olarak .NET adı eklenebilir. Örneğin aşağıdaki kod, DssService1'in adını belirlemek için kullanılır:

```
using svc1 = Robotics.DssService1.Proxy;
```



Şekil 5.5 Referans ekleme diyalog kutusu.

Referansın ilk eklenişinde, partnerliği tanımlamak için bir kod ekleyebilirsiniz. Örneğin aşağıdaki kod DssService1 ile partnerliği belirler:

```
[Partner("Svc1", Contract = svc1.Contract.Identifier,
CreationPolicy = PartnerCreationPolicy.UseExistingOrCreate)]
private scv1. DssService1Operations _svc1Port = new svc1.
                                        DssService1Operations();
```

Diğer servislerle haberleşirken, iki servisin etkileşimine dikkat etmek gerekir. Aksi halde, servisin çalışması bloke olur. CCR, C#'daki yineleyicilerle, eşzamansız görevleri yönetmekte size yardımcı olabilir. Yineleyicileri (Iterators) kullanarak eşzamansız çalışma şekline ihtiyaç duyan kodlar bile sıralı bir biçimde yazılabilir.

Asenkronluğu devam ettirmek için, C#'ın yeni bir özelliği olan yield return kelimesini kullanmanız gerekecek. Yield return size metot çağrısı tarafından geri döndürülen objeleri yineleme imkânı sunar.

Buna ek olarak, çağrı Arbiter.Choice metoduyla bütünleşik olabilir. Örneğin aşağıdaki kod işlemin başarılı veya hatalı olduğunu geri bildirmek için UpdateRequest'i bekleme metodu olan Choice() metodunu kullanır:

```
yield return Arbiter.Choice(
             updateRequest,
             delegate(string response){result = response;},
```

```
                delegate(Exception ex) {Console.WriteLine(ex);}
);
```

CCR'nin bir parçası olan Arbiter sınıfı Çizelge 5.5'de de gösterildiği gibi istekleri ve cevapları koordine etmeye yarayan birçok yardımcı metot sunar.

Çizelge 5.5 Arbiter sınıfındaki yardımcı metotlar

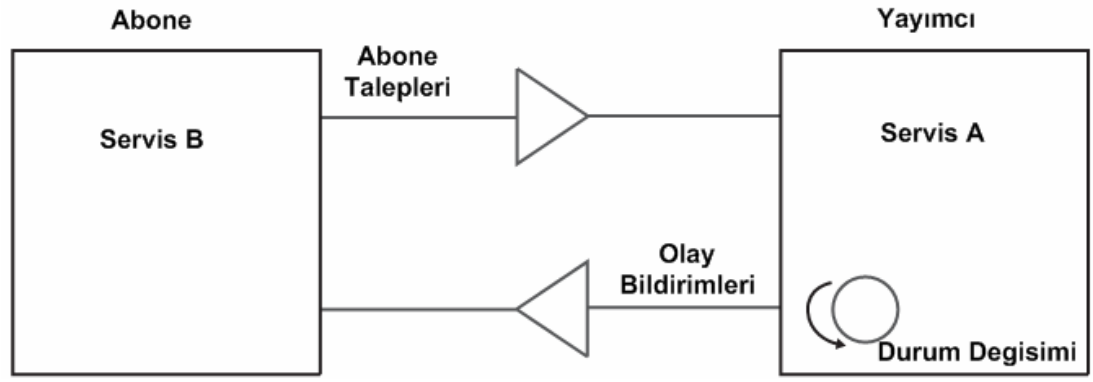
METOT ADI	AÇIKLAMA
Activate	Bir porta aktarılan öğeleri her daim işleyen tek öğe alcısını çağırır.
Choice	Başarılı veya hatalı gibi çoklu çıkışları işlemek için yineleyiciyi etkin kılar.
ExecuteToCompletion	Görevi aktive etmek için kullanılır.
Interleave	Kaynağı eş zamanlı işlemlerden korumak için kullanılır.
JoinedRecieve	Her görevi işleyebilen çoklu port alıcısını aktif eder.
JoinedRecieveWithIterator	Hem çoklu port alıcısını hem de yineleme işlemini aktif eder.
MultipletemReciever	Bir veya birden çok porttan gelen öğeleri toplar. Sonuç type-based toplama şeklindedir.
MultiplePortReciever	Belirlenen kadar mesajın çoklu porta gelmesini beklemek için kullanılır.
Recieve	Tek öğe alıcısını işleme almak için kullanılır.
RecieveFromPortSet	Generic olmayan Portset için tek öğe alıcısı oluşturur.
RecieveWithIterator	Manüel yineleyici için tek öğe alıcısı oluşturur.
RecieveWithIteratorFrom - PortSet	Generic olmayan PortSet'e bağlanılırken manüel yineleyici için tek öğe alıcısı oluşturur.

Aralara giren işlemler için asenkronluğu korumanın diğer bir yolu, Arbiter.ExecuteToCompletion() metodunu kullanmaktır. Bu yardımcı metot, mesaj işleyici tabanlı yineleyiciyi çağırmak için kullanılan SpawnIterator() metoduyla birleştirilebilir.

5.9 Abonelikleri İşlemek

Robotlar her zaman sensörlerinden bilgi almayı ister. Örneğin bir robotun tampon sensörü bir engele çarptığında bununla ilgili bilgiyi almaya gereksinim duyar. Bu ihtiyacı karşılamak için, DSSP diğer servislerden gelen bilgiler için alıcı servisler sunar. Tampon sensöründen bilgi okuyan servis, tamponun konumu değiştiğinde, bu değişikliği diğer bir servise bildirmelidir. Abonelikleri işleme yeteneği, MSRS'nin önemli bir bileşenidir. Bu özellik olmadan, gerçek ortamda robot uygulamalarını yürütmek ve izlemek olanaksız olurdu.

İki servis arasında gerekli bir abonelik vardır. Örneğin Service A ve Service B olsun. Servis A durumu sunar ve yayımcı gibi çalışır. Abone olan Servis B, Servis A'daki değişiklikleri karşılar ve burada not edilir. Her iki servise de abone kodu eklenmelidir. Servis A aboneliği desteklemelidir, Servis B de abone isteklerini kabul etmelidir. Servis B, Servis A'nın durumu değiştiğinde bilgiyi almalıdır. Bu durum Şekil 5.6'da gösterilmiştir.



Şekil 5.6 Abone servis ile yayımcı servis arasındaki ilişki.

5.9.1 Yayımcıyı Yapılandırmak

MSRS, Abone Yöneticisi (Subscription Manager) adlı bir sistem servisi sunar. Kontrol panelinde bu servisi görülebilir. Abone Yöneticisi servisi, bilgileri alıcı abonelere iletmekle sorumludur. Abone Yöneticisi sınıfını, yayımcıdan uygun bir şekilde çağırmak için isim-alanını aşağıdaki kodda belirtildiği şekilde kullanılır:

```
using submgr = Microsoft.Dss.Services.SubscriptionManager;
```

Abone yöneticisi, aboneliği desteklemesi gereken her servis için bir servis partneri olarak servis fonksiyonları sunar. Abone yönetici hizmeti partner niteliği kullanılarak beyan edilir.

Örneğin aşağıdaki kod, abone yöneticisini, yayımcı hizmetine bir ortak olarak beyan etmek için kullanılmaktadır.

```
[Partner("SubMgr",Contract = submgr.Contract.Identifier,
CreationPolicy = PartnerCreationPolicy.CreatAlways)]
private submgr.SubscriptionManagerPort _submgrPort = new
submgr.SubscriptionManagerPort();
```

Bir servisin aboneliğe izin vermesi için, aşağıda belirtildiği şekilde abone talep tipi ve abone yanıt tipini destekleyen abonelik mesajını sağlaması gerekmektedir.

```
public class Subscribe : Subscribe<SubscribeRequestType,
                    PortSet<SubscribeResponseType, Fault>>
{
}
```

Bir kereliğine mesaj oluşturulunca, ana işlem portu için, mesaj PortSet'e eklenmelidir. Ayrıca abonelik işlemi için bir servis işleyicinin olması gerekmektedir. Bu işleyici abone servisini kullanarak aboneleri listeye eklemekle sorumludur. İşleyicinizin alt DSSP Servis Tabanı parametresini kabul ettiğini varsayacak olursak, Abone yardımcısı metodu, abone işleyicinin içine yerleştirilebilir.

```
SubscribeHelper(_submgrPort, sub.Body, sub.ResponsePort);
```

Yayımcının yapması gereken son şey de her ne zaman durum bilgisi değişirse, bu değişiklerle ilgili bilgileri abonelere göndermektir. Güncellenen durumun herhangi bir yerine bilgi gönderme kodu eklenmelidir. Örneğin aşağıdaki kod, ReplaceHandler'a eklenebilir:

```
base.SendNotification(_submgrPort, replace);
```

Bilgi gönderimi için bu kodu eklediğiniz gibi, tüm işlemlerin bir sonuç üretemeyeceği de aklınızda olsun. Örneğin Get ve Query sonuç üretmez. Sonuç üreten işlemler aşağıdakilerdir:

- Delete
- Drop
- Insert
- Replace
- Update
- Upsert

5.9.2 Aboneyi Yapılandırmak

Yayımcı servis biçimlendirildikten sonra, abone servise de kodlar ilave edilmesi gerekir. Yapılacak ilk şey, yayımcı servis için Proxy assembly dosyasını referans almaktır. Proxy assembly, Solution Explorer'da sağ tıklayarak References'ten "Add Reference" seçilerek referans alınır. Daha sonra MSRS yüklemesi için "bin" dizini bulunmalı ve yayımcı servis için Proxy.dll'nin yeri belirtilmelidir.

Referans eklendikten sonra, abone servis için çıkış klasörüne kopyalanmasını engellemek için bazı özellikleri değiştirilmelidir. Bu yapılarak daima doğru assembly versiyonunu referans alınmış olunur. Özellikleri değiştirmek için "References" klasöründeki assembly'i bulunur, sağ tıklanır ve "Özellikler" seçilir. Copy Local ve Specific Version özellikleri "false" olarak değiştirilir.

Yayımcı sınıfı kolayca işleyebilmek için, abonenin "implementation" sınıfına bir isim belirlenebilir. Abone serviste oluşturulana benzer bir Partner tanımlaması eklenmesi gerekir. Bu, yayımcı servise eklenen Subscribe işlemini abone servisin işleyebilmesine izin verir. Bu izin, yayımcı servisin durum değişiklikleri bilgilerini abone servisin karşılayabilmesi demektir. Yapılacak son şey de, abone servise, Subscribe işlemini çağıracak kodu eklemektir.

5.10 Çıkışı İşlemek

MSRS, Extensible Stylesheet Language Transformations (XSLT) dosyası kullanan servis durumunu biçimlendirmesini sağlar. Bu olay durum datasını

görüntüleyen bir servis geliştiricisi sunar. XSLT dosyası, XML servisten döndüğü durumlarda, XML dosyasını dönüştürmek için kullanılır. Sonuç, biçimlendirilmiş HTML, XML veya TEXT dokümanıdır.

XML'i dönüştürmek için, bir XSLT dosyası oluşturulur. Bunun için Solution Explorer'daki Visual Studio projesine sağ tıklanıp "Ekle (Add)" ve "Yeni Öğe (New Item)" tıklanır. HTML olarak biçimlendirmek için, XSLT dosyası, "html" değeriyle kurulmuş bir metot içeren çıkış elementine sahip olmalıdır. Örneğin aşağıdaki XSLT dosyası DssService1'deki tek durum öğesini biçimlendirmek için kullanılabilir:

```
<?xml version="1.0" encoding="UTF-8" ?>
<xsl:stylesheet version="1.0"
     xmlns:xsl="http//www.w3.org/1999/XSL/Transform"
     xmnls:svc1="http://schemas.tempuri.org/2007/07
/dssservice1.html"

 <xsl:output method="html"/>
 <xsl:template match="/svc1:DssService1State">
  <html>
    <head>
      <title>DSS Service 1</title>
      <link rel="stylesheet" type="text/css"
href="/resource/dss/Microsoft.Dss.Runtime.Home.Styles.Common
.css" />
    </head>
    <body style="magrin:10px">
       <h1>DSS Service 1 - XSLT Examples</h1>
       <table border="1">
         <tr class="odd">
           <th colspan="2">Service State formatted with
XSLT</th>
         </tr>
         <tr class="even">
           <th>Output Message:</th>
           <td>
             <xsl:value-of select="svc1:OutputMsg"/>
           </td>
           </tr>
    </body>
  </html>

 </xsl:template>
```

```
</xsl:stylesheet>
```

Stylesheet, servis contract'a bir referans içerir (http://schemas.tempuri.org/2007/07/dssservice1.html). Eğer Contract için bir URI bilinmiyorsa, servis için manifest.xml dosyasını açılır ve "dssp:Contract" elementinden erişim sağlanılabilir. Bu referans diğer adıyla svc1 olarak bilinir ve stylesheet için kullanılır.

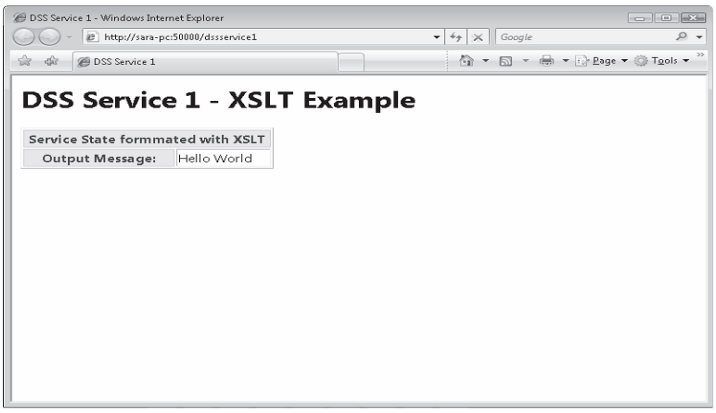
Stylesheet tamamlandığı zaman, bunu servis assembly'nin gömülü kaynağı olarak eklenir. Bunun için Solution Explorer'daki XSLT dosyasına sağ tıklanır ve Build Action'ı Embedded Resource olarak değiştirilir. Aynı zamanda aşağıdaki kodda da ilave edilir:

```
[EmbeddedResource("Robotics.DssService1.TransformOutput.
xslt")]
string _transform = null;
```

Bu noktada iken, bu HttpGet mesaj işleyiciden referans olarak alınabilir. HttpGet işlemi kullanılmak zorunda, çünkü bu sayede geri dönüş bir SOAP mesajı değil, XML mesajı olur. Aşağıdaki gibi bir kod, yeni bir gömülü kaynağı referans almak için kullanılabilir:

```
httpget.ResponsePort.Post(new HttpResponseType(System.Net.
HttpStatusCode.OK, _state, _transform));
```

Servisi derledikten sonra çıkış, Şekil 5.7'de de görüldüğü gibi Internet tarayıcıda HTML biçiminde görüntülenir.



Şekil 5.7 XSLT dosya biçimli ekran görüntüsü

5.11 Hataları İşlemek

Eğer gönderici bir isteği karşılarsa, işlemin başarıyla yerine getirilip getirilemediğine dair bir mesaj gönderilir. İşlem her ne zaman bir yanıtlama portu içerirse, bu mesaj için port set ile World Wide Web Consortium (W3C) SOAP hata gövdesi dâhil edilebilir.

DSSP, Çizelge 5.6'da gösterildiği gibi bir takım hata kodları sunar. Örneğin, bir giriş işlenirken, durum öğesinin DuplicateEntry hata kodu kullanımına sahip mi diye gözlemlenebilir. Aşağıdaki kod, böyle bir örneği gösterir:

```
if (/* the item already exists */)
{
   insert.ResponsePort.Post(Fault.FromCodeSubcodeReason(
FaultCodes.Sender, DsspFaultCodes.DuplicateEntry));
   yield break;
}
```

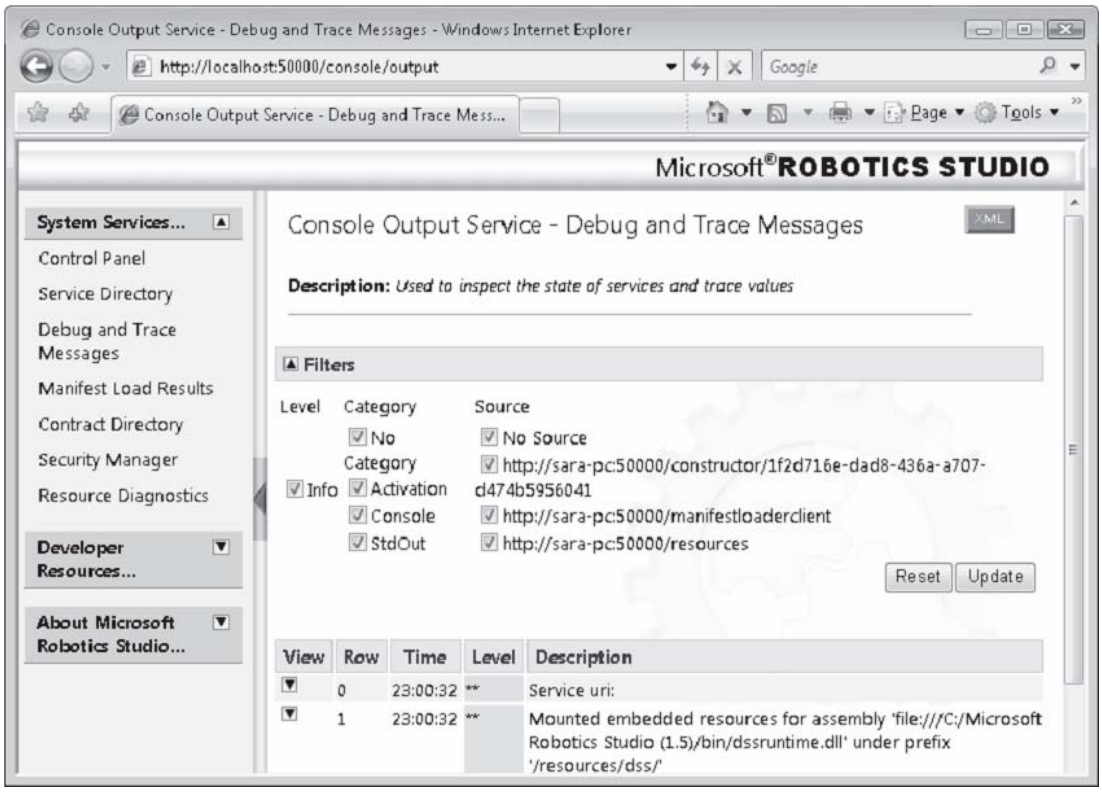
Çizelge 5.6 DSSP servis modelinin sunduğu DSSP Hata Kodları

HATA KODU	AÇIKLAMA
ActionNotSupported	İşlem servis tarafından desteklenmiyor.
DuplicateEntry	Zaten durum öğesinde var olanla aynı giriş. (Eş giriş)
InsufficientResources	Yetersiz kaynak sebebiyle işlem gerçekleştirilemedi.
MessageNotSupported	DSSP gövde tipi desteklenmiyor.
OperationCancelled	İşlem iptal edildi.
OperationFailed	İşlem gerçekleştirilemedi.
QueuingLimitExceeded	İşlem gerçekleştirilemedi çünkü hedef portun queuing (sıraya girme, kuyruk) limiti üst limiti aştı.
ResponseTimeout	Yanıt verme süresini aştı.
UnknownEntry	Query, update veya upsert işlemi gerçekleştirilemedi çünkü bir durum öğesi yok.

5.12 Mesajları Düzeltmek ve İzlemek

MSRS, Şekil 5.8'de gösterildiği gibi hataları ve servisten dönen bilgilendirme mesajlarını görebilmeniz için bir ara yüz sunar.

Filtreler gösterilen mesajların türlerini sınırlandırabilmenize imkân sağlamaktadır. Örneğin eğer hata kontrol kutusunu seçerseniz sadece, hata seviye izi ile işaretlenmiş mesajlar listede belirecektir.



Şekil 5.8 Hata ve İzleme Mesajları penceresi

Ayrıca kesin bir kaynak ya da kategoriden oluşan mesajları göstermek için de çıktı sınırlandırılabilir. Konsol Çıktı servisinde beliren mesajları yüklemek için, aşağıdaki DSSP hizmet tabanı destekli yükleme fonksiyonlarından birini kullanılması gerekmektedir.

- **Bilgi Yükle:** Bilgi izleme seviyesi ile ilgili mesajı yükler. Bu mesajlar seviye sütunu için 2 yıldız ve sarı renkle geri planda belirecektir.
- **Hata Yükle**: Hata izleme seviyesi ile ilgili mesajları yükler. Bu mesajlar seviye sütunu için 3 yıldız ve pembe renkle geri planda belirecektir.
- **Uyarı Yükle:** Uyarı izleme seviyesiyle ilgili mesajları yükler. Bu mesajlar yıldızsız ve geri planda hiç renk olmadan belirecektir.
- **Grupları yükle:** Kategorileri yükler ve mesajları belirgin, kesin kategorilere ayırmak için de kullanılabilir. Bu durum, düzeltme ve izleme penceresinde gözüken mesajları sınırlandırmaya olanak sağlar.
- **Gereksiz şeyleri Yükle**: Sadece sistem servisinden kaynaklanan mesajları yükler.

Bir izleme seviyesi, yüklenecek olan mesajın türünü belirtmek için kullanılır. DssInfo, mesajların bilgi izleme seviyesi olarak yüklenmesi için bir metot sağlamaktadır. Bilgi olarak yüklenen mesajlar, seviye sütununda sarı bir geri plan olarak belirecektir. Bir mesaj aşağıdaki kodu kullanarak bilgi olarak yüklenebilir:

```
LogInfo("This is the state for my service " +
_state.OutputMsg);
```

Gözlemleme sütununda belirmekte olan ok tıklanarak herhangi bir mesajın detaylarını gözlemlenebilir. Bu durum, mesajın ne servisinden ve hangi zamanda gönderildiği ile ilgili toplam bilgiyi gösterir. Örneğin bilgi mesajı için olan detay "Bu Merhaba Dünya servisinden gönderilen bilgidir". Şekil 5.9'daki gibi belirecektir.

Message	This is information posted from my service Hello World
Category	StdOut
Level	Info
Time	2007-07-16T12:42:48.344-05:00
Subject	This is information posted from my service Hello World
Source	*http://sara-pc:50000/servicetutorial1*
CodeSite	Void Start()() at line:72, fileC:\Microsoft Robotics Studio (1.5)\samples\Service-Tutorials\Tutorial1\CSharp\ServiceTutorial1.cs

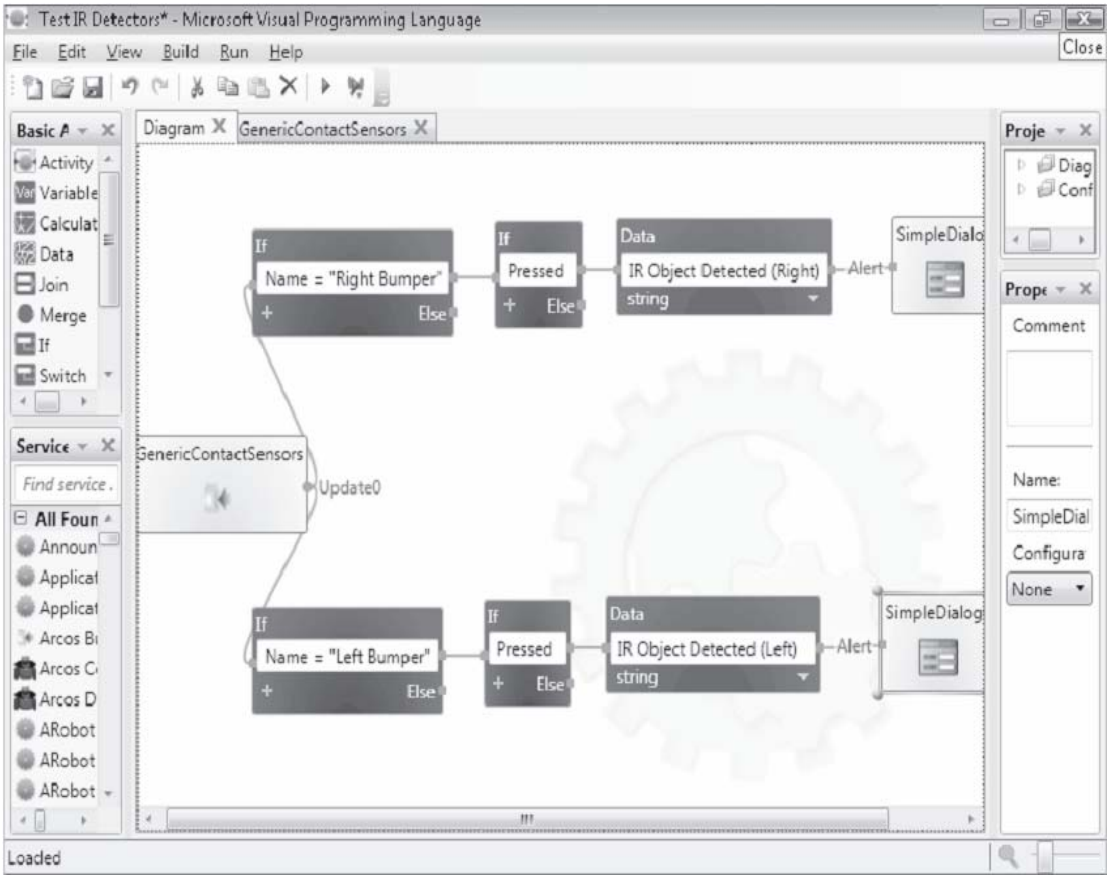
Şekil 5.9 Bilgi olarak görüntülen mesaj penceresi

Durumda değişiklik yapan fonksiyonları gerçekleştirirken, kodun başarılı bir şekilde güncellendiğinden emin olmak amacıyla kontrolleri de dâhil edilebilir. Eğer eklenmezse LogError fonksiyonunu kullanarak hata oluşması durumunda beraberinde bir hata mesajı göndermek için kod eklenebilir. Örneğin, hiç hata ile karşılaşmaz ise durumu koruyacaktır. Aksi takdirde hata mesajını yüklemeye gönderecektir.

```
yield return Arbiter.Choice(
   base.SaveState(_state),
   delegate(DefaultReplaceResponseType success) {},
   delegate(W3C.Soap.Fault fault)
   {
      LogError(null, "Unable to store state", fault);
   }
);
```

6 GÖRSEL PROGRAMLAMA DİLİ (VPL)

VPL, basit veya karmaşık robot uygulamaları üzerinde çalışılabilinen grafiksel bir uygulama dizayn aracıdır. Bu aracı kullanmak oldukça kolaydır. Dizayn yüzeyi üzerine sürükle – bırak şeklindeki programlama öğelerini kullanılır ve giriş – çıkışları belirlemek için bağlantı uçları kullanılır. Şekil 6.1'de VPL dizayn ekranı gösterilmiştir. Öğeler sadece bir bilgi içerecek kadar sade olabildiği gibi, karmaşık bir servisi de oluşturabilir. Tez çalışmasının bu bölümünde VPL ile temel aktivitelerin ve basit uygulamaların nasıl yapıldığı üzerinde durulmuştur. Nasıl bilgi gönderildiği, dâhili servislerin kullanılması ve bir VPL uygulaması derlemesi anlatılmıştır.



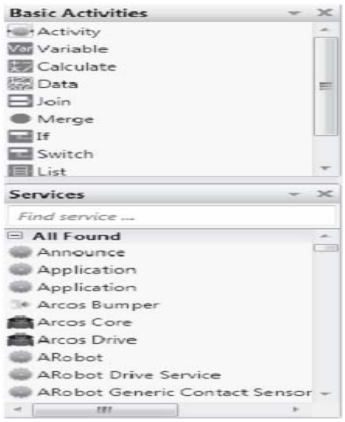
Şekil 6.1 VPL robot uygulamaları için sürükle-bırak dizayn ekranı

6.1 VPL'ye Genel Bir Bakış

VPL, kontrol tabanlı olmayıp, bilgi tabanlı olmasıyla, Web gibi başka geliştirme araçlarından farklıdır. VPL, birçok farklı işlemin bir anda veya belli bir zamanda ayrı ayrı işlemesine olanak sağlar. Örneğin bir robot çevresinden bilgi alan birçok sensöre sahip olabilir. Tampon sensörü, önüne çıkan bir engeli fark edebilir. Hareket sensörü, robotun yakınlarında hareket eden bir cismi algılayabilir. GPS cihazı robotun yerini bildirebilir. Sonuçta birçok farklı kaynaktan farklı bilgiler gelebilir ve elimizdeki uygulama tüm bu bilgileri aynı anda eş zamanlı olarak işleyebilmelidir. Ve aynı zamanda, robotun tekerleklerine hareket emrini veren komutları yorumlayabilmelidir. Böyle bir senaryoyu gerçekleştirebilmek için aynı anda birçok işleme tepki verebilen bir geliştirme aracı kullanılmalıdır.

6.2 Aktivitelerin Tanımı

VPL uygulamaları, hazır blokları dizayn yüzeyine sürükle – bırak şeklinde gerçekleştirilir. Şekil 6.2'de görülen her blok, basit bir aktivite veya bir servis sunar. Temel aktiviteler, bir değişkenin veya If – Else durumunun değerini depolamak gibi temel fonksiyonlar olarak nitelendirilebilir. Çizelge 6.1'de temel aktiviteler gösterilmiştir.



Şekil 6.2 Temel Aktiviteler ve Servisler

Çizelge 6.1 VPL'deki temel aktiviteler listesi

ADI	AÇIKLAMA
Activity	Yeni bir aktivite oluşturmak için kullanılır. Kullanıcı tanımlıdır ve programcıya kendi özel aktivite bloklarını tasarlamalarını sağlar. Her yeni aktivite giriş değerlerini karşılayıp, çıkış değerleri gönderir. Aktivite aynı zamanda bilgiyi tetikleyebilir.
Variable	Diğer aktivite bloklarında referanslaşabilen değerleri depolamak için kullanılır. Variable servisin durum bilgisi değerini depolar ve daha sonra bu değeri çoklu aktiviteler veya servisler arasında kullanır. Bu sayede, aktiviteler arası değişken bilgi paylaşımında bulunur. Değişkenin değeri .NET Framework'un desteklediği int, string veya Boolean gibi tiplerde olabilir.
Calculate	Add, subtract, multiply ve divide gibi temel dört işlem fonksiyonları için kullanılır. And için &&, Or için || ve Not için ! olacak şekilde lojik işlemleri de kullanılır.
Data	Variable veya Calculate gibi diğer aktivite blokları için data değeri üretmek için kullanılır. Data tipi .NET'in data tiplerinden biri olabilir.
Join	Birden çok aktivite bloğunun sonuçlarını birleştirmek için kullanılır. Bu aktiviteyi kullanarak, çıkış mesajı oluşturulmadan önce tüm giriş bağlantılarınki mesajları karşılama şartı belirtilebilir.
Merge	2 veya daha fazla aktivite bloğunun mesajlarını birleştirmek için kullanılır. Ve daha sonra oluşan birleşmiş sonuç mesajı bir yandaki aktivite bloğuna gönderilir. Bu akivite, mesajları senkronize edecek olmasıyla Join'den farklıdır.
If	If-Then-Else gibi koşullara bağlı ifadeleri desteklemek için kullanılır. "Eşitlik" için =, "daha küçük" için <, eşit değil için != kullanılır.
Switch	CASE ile aynıdır. Bu aktivite bloğu, mesajları düzgün biçimde dağıtmak ve gelen mesajları ayıklamak için kullanılır. Örneğin hangi fonksiyonun kullanıcı tarafından seçildiğini fark edebilmek için Switch bloğu kullanılabilir.
List	Özel data tipi için data öğelerinin listesini oluşturmakta kullanılır. Bu liste bir başka aktivite bloğu için giriş olacak şekilde kullanılır.
List Functions	Sonuç olarak üretilen yeni bir listenin ve var olan listenin öğeleri üzerinde concatenate, insert öğesi, sort, append gibi seçilen fonksiyonları yürütmek için kullanılır.
# Comment	Aktivite bloğuna açıklama notu eklemek için kullanılır.

Bir aktivitenin temel fonksiyonu giriş mesajını onaylamak ve çıkış mesajı üretmektir. Bu arada neler yapılacağı aktivitenin türüne bağlıdır. Aktiviteler, içlerinden geçen mesajlarla birlikte linkleşirler. Bir aktivite için çıkış olan mesaj, başka bir aktivite için giriş olabilir.

Activity isimli en temel aktivite, kullanıcı tanımlı yeni bir aktivite oluşturmaya olanak sağlar. Bu aktivite başka aktivite bloklarını içerebildiği gibi kendisi için referanslar da içerebilir.

VPL'nin başka bir güzel tarafı da, uygulamaları bir servismiş gibi derleyebilir. Bu sayede uygulamada kullandığınız tüm aktviteler bir servis içinde toplanmış olur. Bu servis başka uygulamalarda da kullanabilir hale gelir. Derlediğiniz aktivite servisler listesinde görünür ve başka VPL uygulamalarıyla da kullanılabilir.

VPL uygulamalarında servisler aynı zamanda aktivite blokları gibi kullanılabilir. VPL, MSRS ile oluşturulmuş DSS servis kullanımı da sunar.

MSRS'deki farklı servisleri anlamanın en iyi yolu, onları kategorilerine ayırmaktır. Çizelge 6.2'de gösterildiği gibi çeşitli robot uygulamalarında kullanılmak üzere pek çok servis türü vardır. Bunların tümünü, robotu sürmek ve sensörlerden geri dönen bilgileri karşılamak gibi görevleri yerine getirmek için VPL uygulamalarında birleştirilebilir. Şekil 6.2'de gösterildiği gibi servisler VPL panosunun sol-alt kısmına yerleştirilmiştir. Bazı platform servisleri, generic servisler olarak aktiviteleri kullanır.

Çizelge 6.2 VPL Aktiviteleri ve MSRS'nin Generic Servisleri

ADI	AÇIKLAMA
Generic Analog Sensor	LEGO NXT'deki Işık Sensörü gibi analog sensörlere erişme imkânı sunar.
Generic Analog Sensor Array	Bir dizi analog sensöre erişim imkânı sunar.
Generic Articulated Arm	Çok eklemli robot kollarının kontrolünü sağlar.
Generic Battery	LEGO NXT'deki gibi, Batarya sensörüne erişim imkânı sağlar.
Generic Contact Sensors	Robotun kontak sensöründen gelen geri bildirimleri karşılamak için kullanılır. Sıklıkla kullanılır.
Generic Differential Drive	İki tekerlekli robotlar için sıklıkla kullanılan bir servistir. İki motorunda bağımsız ve bilgileri doğru bir

	şekilde işleyerek çalışmasında kullanılır.
Generic Encoder	LEGO NXT'de de kullanıldığı gibi, Encoder'lı DC motor kontrolü için sıklıkla kullanılan bir servistir.
Generic Motor	Servo motor kullanımı içindir.
Generic Sonar	LEGO NXT'deki gibi, sonar sensörleri işletebilmek için kullanılır.
Generic SQL Store	SQL veritabanı kullanımını aktif hale getiren servistir.
Generic Stream	İki yönlü, packet-based stream erişimi sunar.
Generic WebCam	Web-Cam'den gelen görüntüleri yakalamak için kullanılır.

Generic servislere ve platform servislerine ek olarak, Çizelge 6.3'de verilen miscelaneous servisler vardır ve kullanışlı robot uygulamaları sunar. Örneğin Explorer servisi, robotu differential drive system, tampon ve laser mesafe bulucularla birlikte aktif kılar.

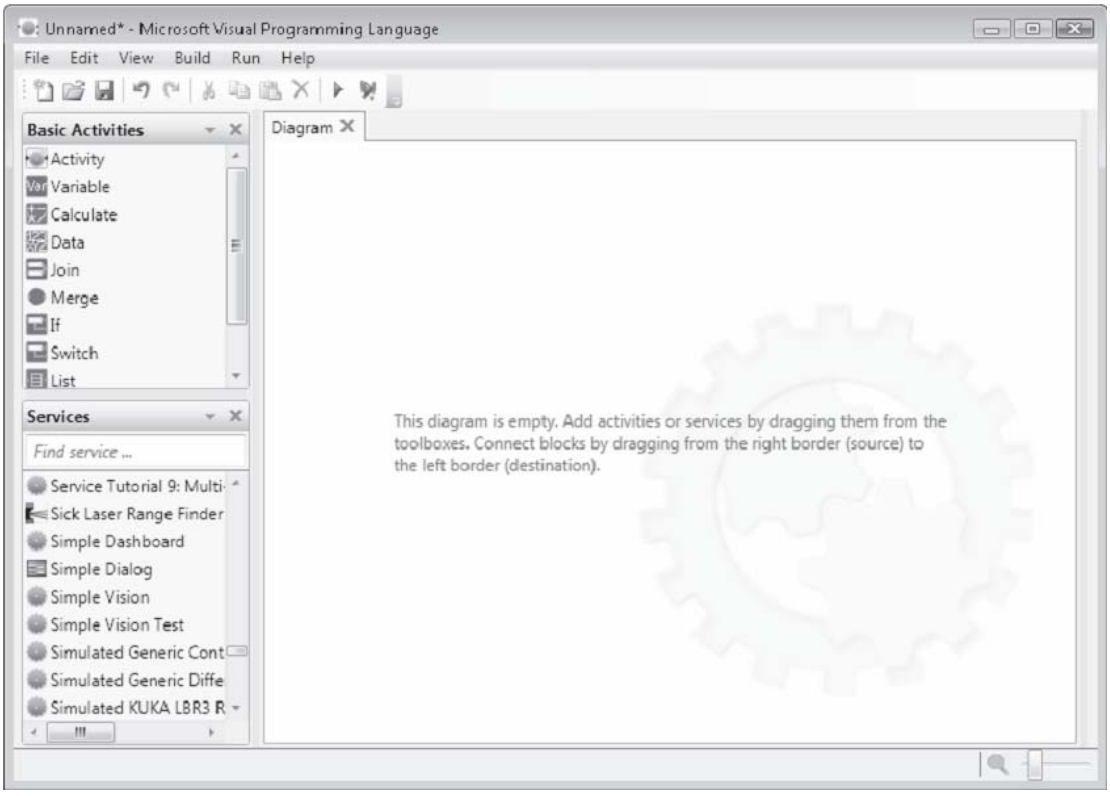
Çizelge 6.3 Miscellaneous Servisler

ADI	AÇIKLAMA
Blob Tracker	Basit bir renk taramasıyla Özel BLOB'ların veya imaj bölgelerinin yer tespiti için kullanılır. Aynı zamanda test servisi de içerir.
SQL Client for ADO.NET	ADO.NET'i kullanan SQL veritabanına erişim imkanı sunar.
Direction Dialog	Robotun ileri, geri, sağa veya sola dönüşünü kontrol edebileceğimiz Windows tabanlı bir diyalog kutusu meydana getirir.
Explorer	Robotun çevreyi keşfetmesinde kullanılır. Önüne bir engel çıktığında başka yönleri deneyerek ortamda dolaşır.
Follower	Bağımlı olduğu bir kaynağa doğru (ses, ışık gibi) giderek kaynağı takip etmesini sağlar.
Game Controller	Joystick veya Gamepad gibi doğrudan girişli oyun kumanda cihazlarına erişim için kullanılır.
Simple Dashboard	Örneğin Mouse ile robotu yönlendirmenize olanak sağlayan Windows tabanlı bir diyalog kutusu meydana getirir.
Simple Vision	Bir kamerayla, renk, yüz veya parmak izi tanıma gibi işleri yerine getirebilmesi için bir takım temel servislere erişim imkânı sunar.
Syndication	Kullanıcının bloglara erişmesini ve güncellemesini sağlayan ATOM/RSS syndication servisine erişim imkânı sunar.
Windows Messages	Mic. Windows mesajlarını gönderme ve alma imkânı sunar.
Xinput Controller	Xbox 360 gamepad'in kullanımı için bir kontrol kumandasına

	erişim imkânı sunar.

6.3 VPL Dizayn Aracının Kullanımı

VPL, öğeleri sürükle – bırak temeline dayalı olarak programlama yapılabilen bir ortamdır. Varsayılan olarak VPL ilk açıldığında Şekil 6.3'de görüldüğü gibi boş bir dizayn diyagramı çıkar. Buradaki boş dizayn bölgesine sol taraftaki panoda listelenmiş olan aktiviteleri veya servisleri sürükleyip bırakarak bir uygulama oluşturulur.



Şekil 6.3 Boş bir VPL diyagramı

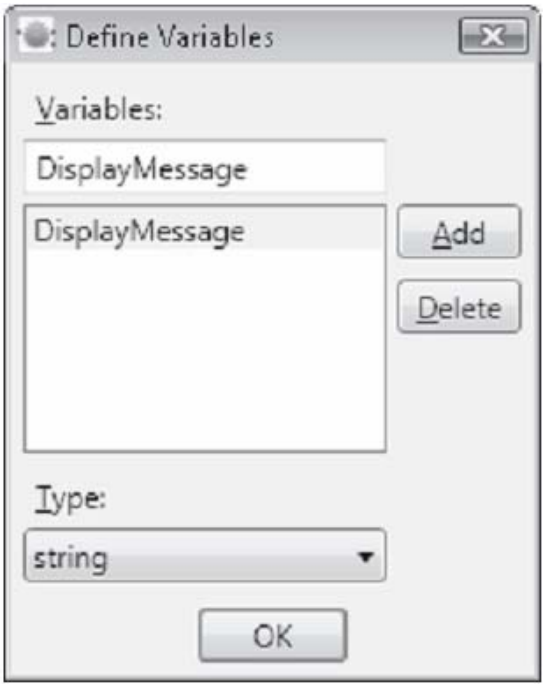
Aktiviteler mesajların takas edilmesiyle birbirlerine bağlıdırlar. Her aktivite giriş mesajına bir işlem uygular ve buna göre bir çıkış mesajı üretir. Kaynak aktivitenin çıkışı, hedef aktivitenin girişi olacaktır. Aktiviteler arası mesaj takasını

kolaylaştırmak için bağlantılar kullanılır. Bir aktivitenin bağlantı noktasından fare ile basılı tutup, diğerine kadar sürüklenirse arada bir bağlantı kurulmuş olunur. Bunu yapınca bir ekran çıkar ve burada hedef aktivitenin yapacağı iş seçilir. Böylece aktiviteye gelen mesaj üzerinde bu seçilen işlem yapılarak bir çıkış mesajı oluşturulur. Bu çıkış mesajı da bir başka aktivitenin girişi olacaktır ve aktiviteler arası ilişki bu şekilde birbirini izleyip gidecektir.

6.4 Bağlantıları Gerçekleştirmek

Bağlantıların nasıl çalıştığını daha iyi anlayabilmek için önceden bahsedilen data ve değişken aktiviteleri anlamak gerekir. Bunu VPL'de yapmak için, her iki data aktivitesini ve variable (değişken) aktiviteyi dizayn yüzeyine sürüklenir.

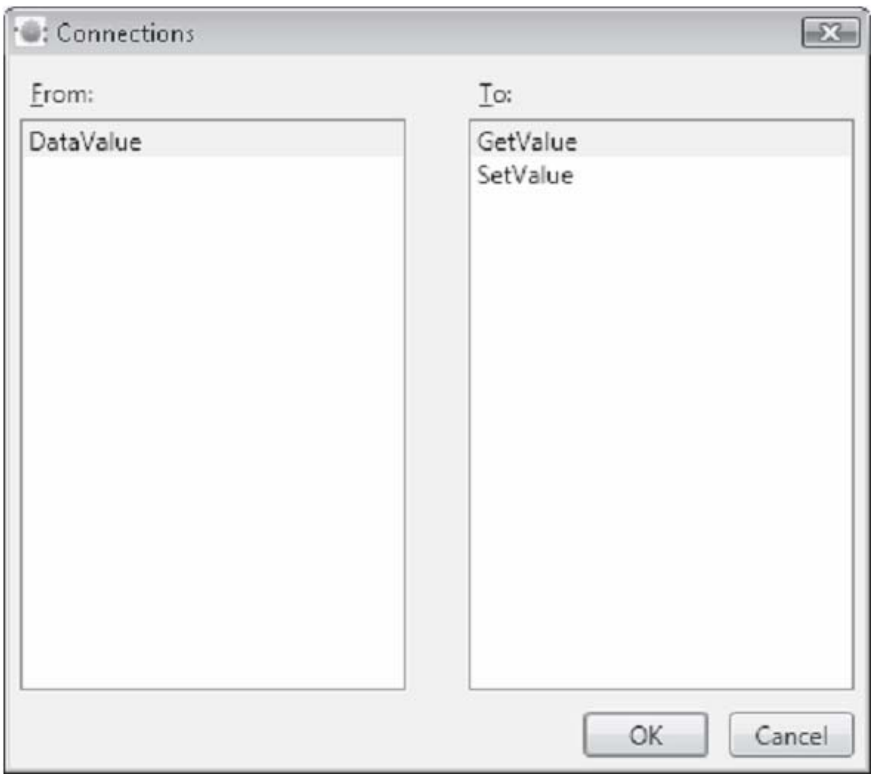
Yeni bir değişken belirleyebilmek için, elipslere tıklanır ve Define Variables ekranı açılır. Buradan değişken ismini seçilir, değişken data tipini belirlenir. Şekil 6.4'de görüldüğü gibi değişken adı "DisplayMessage" ve data tipi "string" olarak seçilir.



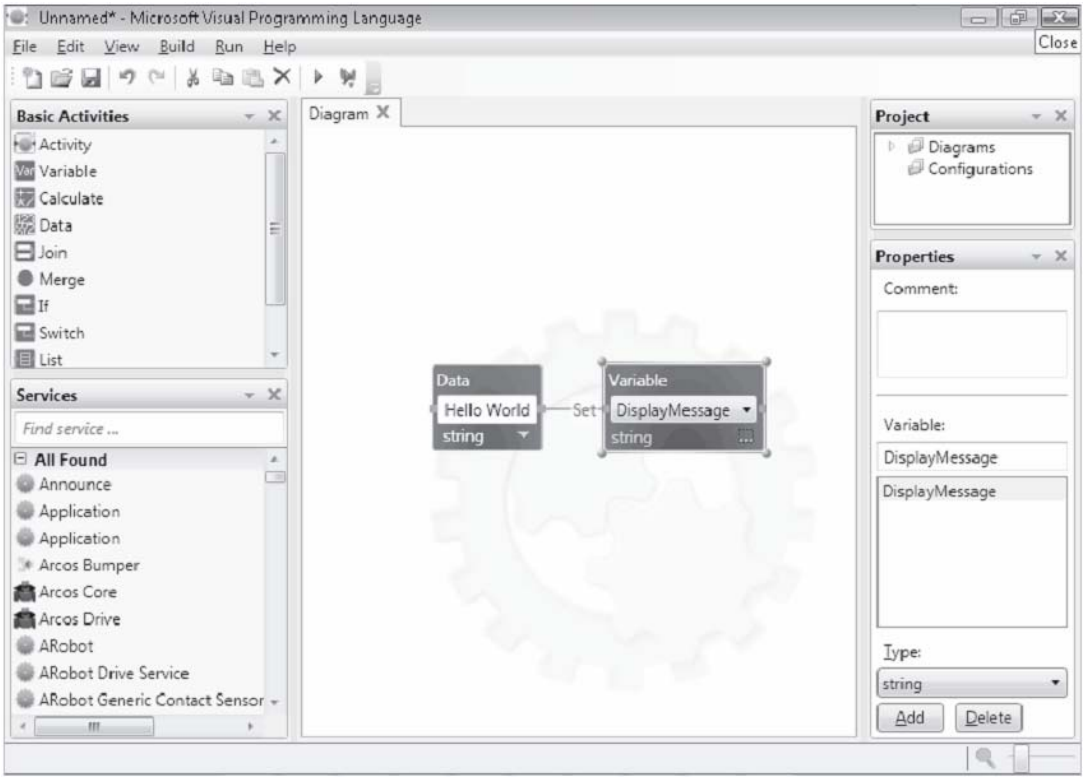
Şekil 6.4 Define Variables diyalog kutusu

Data aktivitesinin değerini ayarlamak için, istenilen değer direkt olarak data text kutusuna yazılır. Aynı zamanda data tipi de listenin aşağısındaki, data text kutusu yanındaki yerden seçilir. Son adım olarak, data ve variable aktiviteler arasında bağlantı kurulur. Bunun için data aktivitenin sağındaki çıkışı, variable aktivitenin solundaki girişe sürükle – bırak ile tutturulur. Bu yapıldığında Şekil 6.5'de görüldüğü gibi Connections diyalog kutusu belirir ve From – To değerlerini belirlenmesi istenir. Bu durumda, burada sadece bir tane From değeri vardır, çünkü burada variable değeri ayarlanacaktır. To değeri için de SetValue seçilmelidir. SetValue, giriş mesajına uygulanabilir işlem sunar.

Bu bağlantı gerçekleştirildikten sonra, Şekil 6.6'daki gibi arasında bir çizgiyle birbirine bağlı iki blokla ana şema oluşmuş olacaktır.



Şekil 6.5 Connections diyalog kutusu ve From – To değerlerinin seçimi.



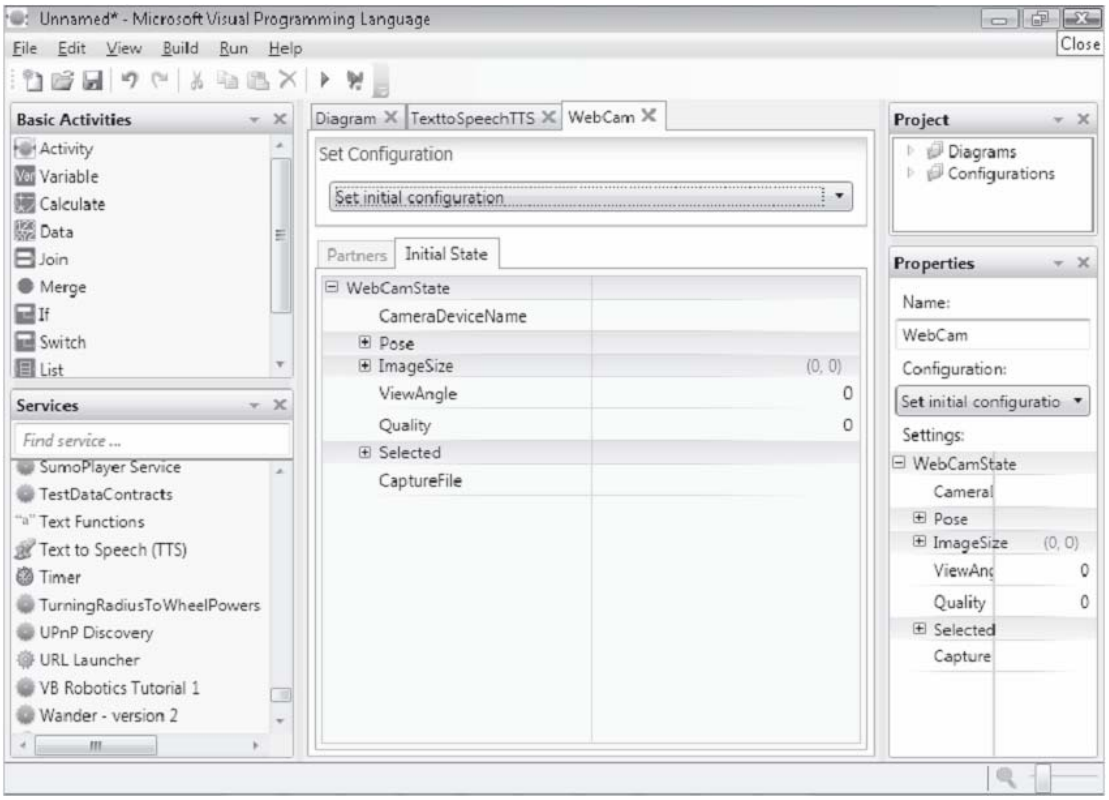
Şekil 6.6 Bağlantı oluşturulduktan sonra oluşan görüntü

6.5 Bir Servisi Yapılandırmak

Aktivite bir servisken, bu servis üzerine çift tıklayarak yapılandırma bölümüne erişilebilir. Şekil 6.7'de görülen "The Set Configuration" (Yapılandırmayı Ayarlama) sayfası servisi yapılandırmaya izin verir. Yapılandırma Ayarı sürükleme kutusu aşağıdaki 4 seçenekten birini seçmeye olanak sağlar:

- **Bir manifest kullan**: Servis yüklenirken var olan manifest dosyasını belirtilmesine olanak sağlar. Bu seçenek bir defa seçildiğinde, Import Manifest adı verilen bir ek buton belirecektir. Bu butona tıklanır ve buradan listelenmiş uygun manifest dosyalarından biri seçilir.
- **Başka bir servisi kullanmak:** Belirtilmiş bir servisin, yürütme şeklini belirlenmesine olanak sağlar. Generic servislerin kullanıldığı VPL uygulamalarında kullanışlı olabilir. Yapılandırma seçimini platform servislerini belirtmek için kullanılabilir.

- **İlk yapılandırmayı ayarlamak:** Bu seçenek, bir servisin durum değerlerinin başlangıcını belirlemek ve ilave partner servislerini yapılandırmak ve seçmek için kullanılır.
- **Boş (None):** Birçok durum için, özel bir yapılandırma yapmak gerekmez. Bu çok geçerli bir seçenektir. Örneğin bu seçenek Simple Dialog ve Text-To-Speech servisleri için çalışır.



Şekil 6.7 Servis özelliklerini ayarlama sayfası

Şekil 6.7'de WebCam servisi için ilk yapılandırma seçeneklerini gösterir. Burada kameranın yakaladığı görüntüleri depolayacağı dosya adı ve kamera adı gibi dinamik parametreleri belirlenebilir. Aynı zamanda kameranın imaj boyutu, görüntü alma açısı, çözünürlük gibi ayarları belirlenebilir. Eğer uygunsa, başlangıç yapılandırma parametreleri de ayarlanabilir. WebCam servisiyle ilgili bu örnekte yapılandırma parametreleri CameraDeviceName ve CaptureFile belirlemek için düzenlenmiştir.

Yapılandırma Kurulum sayfası iki sekmeden oluşur: Partners ve Initial State. Bir veya birden fazla partneri, bir servis olacak şekilde birleştirilebilir. Ve tüm bu partnerler de çeşitli yapılandırma dosyalarının birleşiminden oluşabilir. Buna alternatif olarak, bir servisin, birleşmiş hiçbir partneri de olmayabilir. Başlangıç yapılandırma seçeneklerini belirlendiğinde, VPL uygulamasıyla birlikte bir yapılandırma dosyası oluşacaktır.

Servisleri yapılandırmanın bir diğer yolu da manifest dosyasını kullanmaktır. Manifest; birlikte başlatılabilen servislerin bir listesini içeren XML tabanlı bir dosyadır. Bu dosya, servisin ihtiyaçlarını karşılayan bir yapılandırma dosyası içerir. XML tabanlı olan bu yapılandırma dosyası, servisin ihtiyacı olan dinamik yapılandırma parametre değerlerini içerir.

MSRS, satın alabilecek robotları kontrol edebilmek için kullanılan servisleri ve yapılandırma dosyalarını içerir. Bir servis için yüklenebilecek manifest dosyaları Şekil 6.8'deki gibi Import Manifest diyalog kutusunda listelenmiştir.



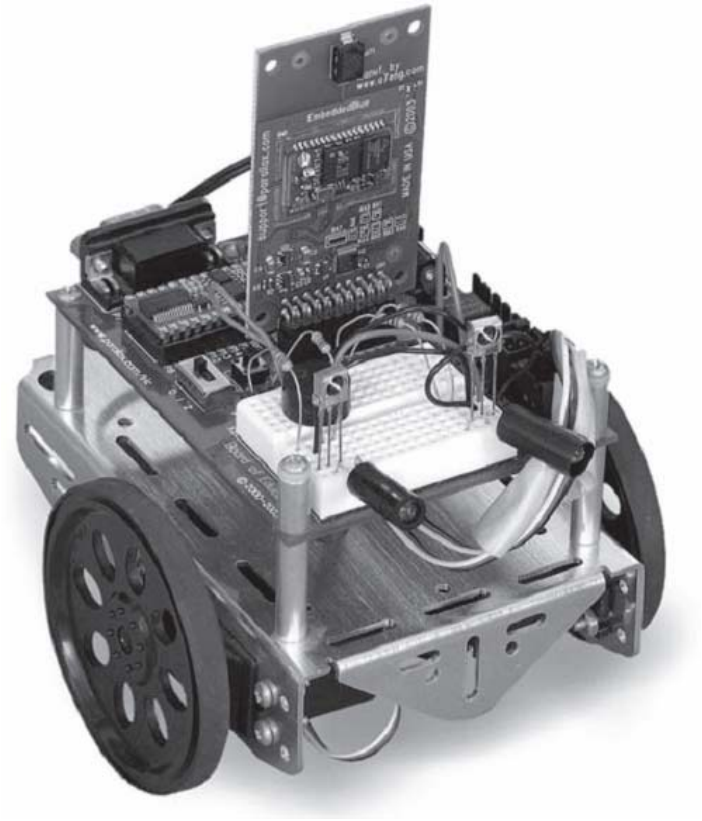
Şekil 6.8 The Import Manifest diyalog kutusu

Eğer manifest'i belirtme işlemi seçilirse, ilk olarak yapılandırma dosyası düzenlenmeli ve seri port öğesinin değeri değiştirilmelidir. Yapılandırma dosyası, durum değerlerini depolamak için kullanılan çeşitli öğeler içeriyor olabilir. Örneğin, Boe-Bot robotunu çalıştırmak için kullanılan yapılandırma dosyası aşağıdaki gibidir:

```
<Configuration>
   <Delay>0</Delay>
   <SerialPort>5</SerialPort>
</Configuration>
<AutonomousMode>false</AutonomousMode>
<Connected>false</Connected>
<FrameCounter>0</FrameCounter>
<ConnectAttempts>0</ConnectAttempts>
<MotorSpeed>
   <LeftSpeed>0</LeftSpeed>
   <RightSpeed>0</RightSpeed>
</MotorSpeed>
<Sensors>
   <IRLeft>false</IRLeft>
   <IRRight>false</IRRight>
   <WhiskerLeft>false</WhiskerLeft>
   <WhiskerRight>false</WhiskerRight>
</Sensors>
```

6.6 BOE-BOT'la Çalışmak

Boe-Bot robotu, Parallax firmasının ürettiği küçük, sağlam, dayanıklı ve sadece bazı temel el aletlerini birkaç saat kullanmayla kolayca oluşturulabilir bir robottur. Şekil 6.9'da gösterilen bu robot öğrenciler ve hobi olarak düşünenler için oldukça popülerdir. Dünya çapında 90.000'den fazla Boe-Bot kullanılmaktadır. Bir Boe-Bot araç takımı Bluetooth kablosuz modül, nesne algılama ve mesafe ölçmek için kullanılan bir takım kızılötesi infrared detektör ve çevresindeki engelleri algılayan iki takım kontak sensörleri içerir. MSRS tarafından desteklenen robotlardan biridir. Ve robotların çalışmasıyla ilgilenenlere harika bir öğrenme imkânı sunar.



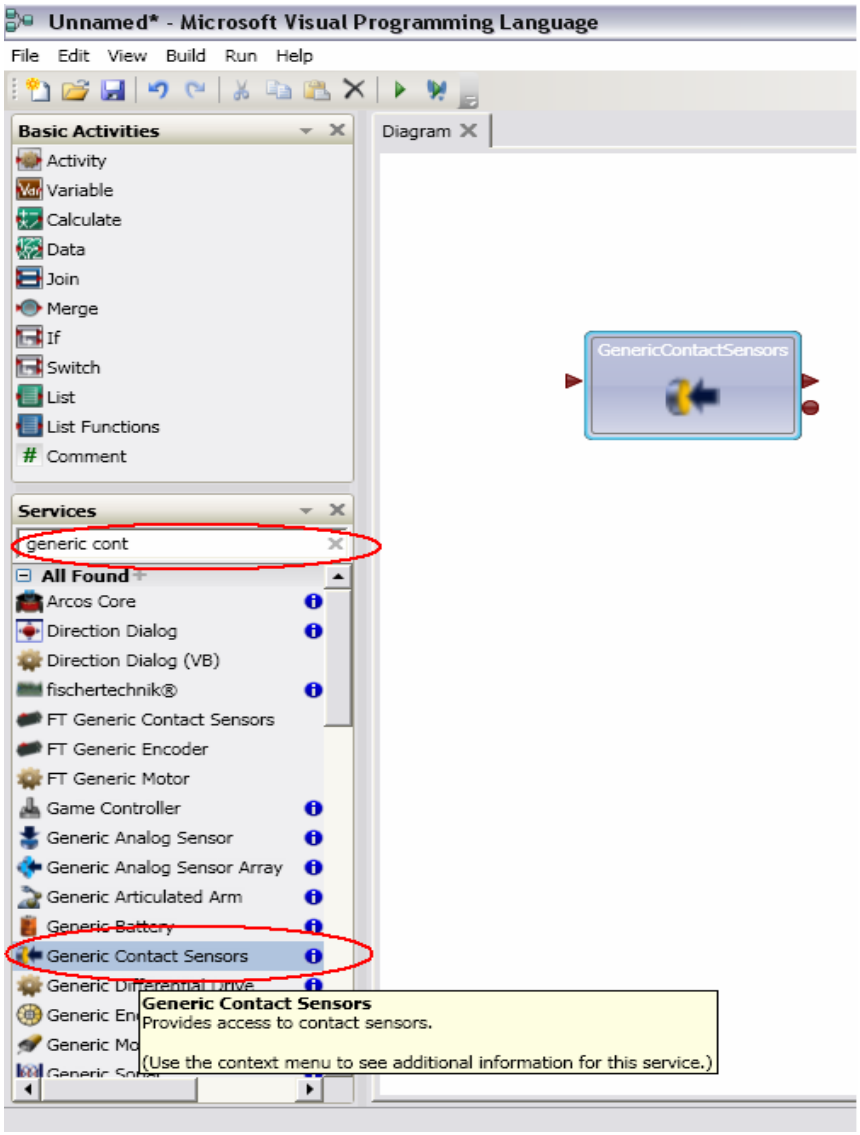
Şekil 6.9 Parallax Firmasının ürettiği BOE-BOT robottu

6.7 Basit Bir VPL Uygulaması Yazmak

Bu bölümde, Boe-Bot Tampon Test adlı basit bir VPL uygulamasını kurmak ve oluşturmak için gerekli adımlar yapılacaktır. Bu uygulama, makineye "Bir nesne bulundu" demesini sağlayacak ve aynı zamanda detektörlerin tetiklenmesini sağlamış olacaktır.

İlk önce diyagram ekranına bir servis eklenmesi gerekir. Bu işlem için Şekil 6.10'da da gösterildiği gibi ya "Services" araç kutusunun üstünde "Find service" text kutusuna aranan servisin ismi yazılır ya da eklenecek "Generic Contact Sensors" servisi bulunana kadar liste kaydırılır. Bu servis diyagram yüzeyine tutup sürükleyerek veya servis üzerine çift tıklanarak eklenebilir.

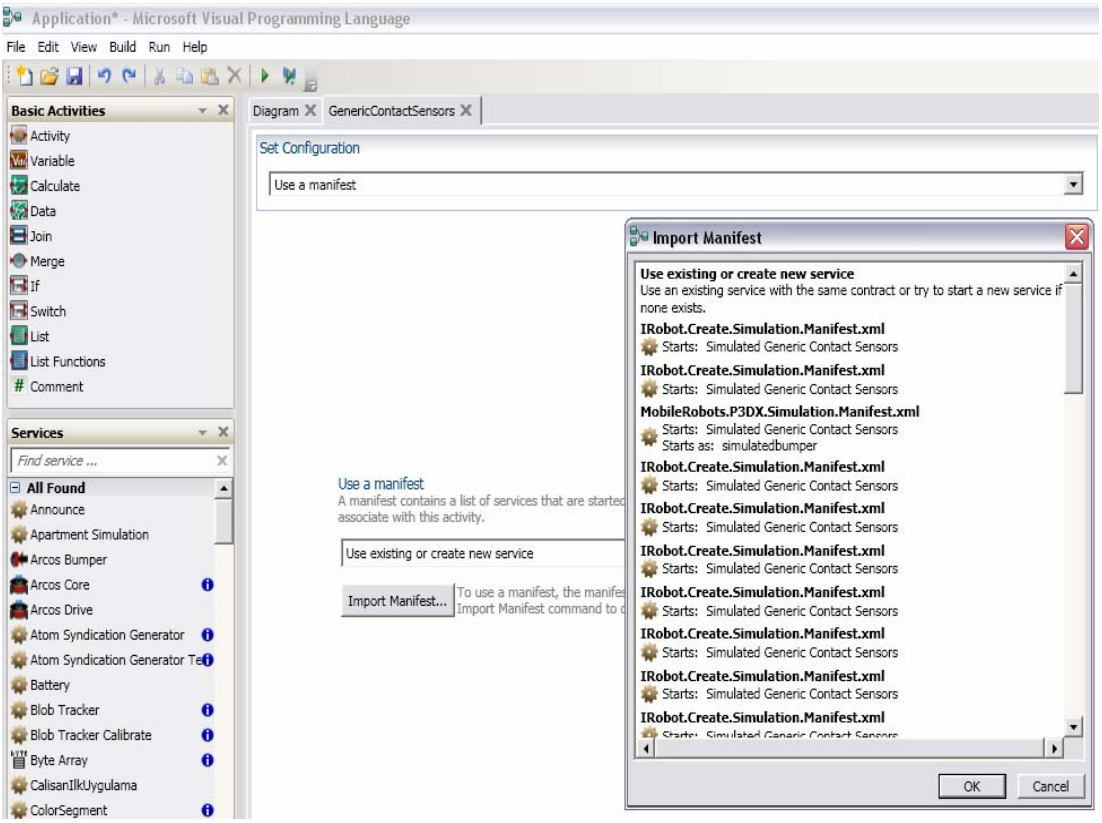
Generic contact sensors servisi veya diğer servislerin çoğu, fonksiyonlarının çok sayıda robot platformları aracılığıyla kullanılabilmesi için düzenlenmeye ihtiyaçları vardır.



Şekil 6.10 “Services” araç kutusundan bir servis eklemek

Generic contact sensors servisi ve Boe-Bot arasında bir bağ oluşturmak için sensör servisini yapılandırmak gerekir. Bunun için ana diyagrama yerleştirilen servis bloğuna çift tıklayarak düzenleme yapılacak bölüme ulaşılır. Bu aşağı doğru açılan liste kutusundan “Use a manifest”i (Bir manifest kullan) seçilir. Buradan manifest dosyasının gösterilebileceği yapılandırma sayfasını açar. Buradaki “Import Manifest” (Manifest ekle) butonuna tıklanır ve bir diyalog kutusu açılır. Bu diyalog kutusu

Şekil 6.11'deki gibi servis tarafından desteklenen bir manifest dosyaları listesini gösterir. Bu servisi Boe-Bot'a bağlamak için "Parallax.MotorIrBumper. manifest.xml" manifest dosyası seçilir ve tamam tıklanır. Hangi sensörün kullanılması gerektiğini bildiren manifest dosyası budur. Varsayılan bu manifesto dosyası MSRS'nin yüklendiği yerdeki "\samples\config" klasörü içersine yerleştirilmiştir. Bu manifest dosyaları ve isimleri MSRS'nin versiyon farklarından dolayı değişiklik gösterebilmektedir. Burada bahsedilenler MSRS V1.5 için geçerlidir.



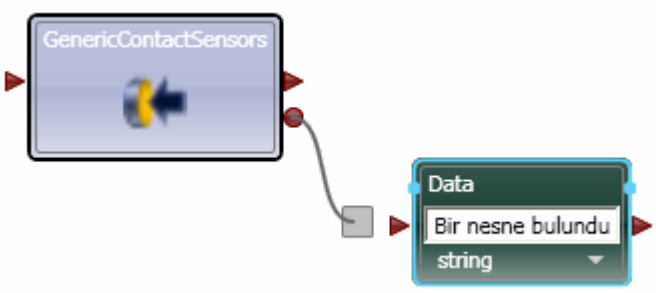
Şekil 6.11 Generic Contact Sensors servisini manifest dosyası ile yapılandırmak

Boe-Bot sadece generic contact sensors servisi ile birlikte kullanabilen bir robot değildir. Manifest diyalog kutusu başka robotların imalatçıları tarafından desteklenen bir manifesto dosyası listesi içermektedir. Bu liste aşağıdakileri kapsamaktadır:

- iRobot'un Create
- Lego'nun NXT ve RCX
- Mobil Robot'un Pioneer 3DX
- Fischerteknik'in Bionic Walker

Generic contact sensors servisi yapılandırıldıktan sonra dizayn yüzeyinin üstündeki “Diagram” sekmesi tıklanarak diyagram bölümüne geri dönülebilir. Daha sonra “”Basic Activities” (Temel Aktiviteler) araç kutusundan “Data” aktivitesi sürüklenip alınır. Data aktivitesi Generic contact sensors servisinin sağına yerleştirilir. Data aktivitesi bloğun içindeki text kutusuna tıklanır ve içerisinde gösterilen 0’ı seçilerek yerine “Bir nesne bulundu” şeklinde yazılır. Sonra da yazı yazdığımız text kutusunun hemen altından data yazısının formatı için “string” seçilir.

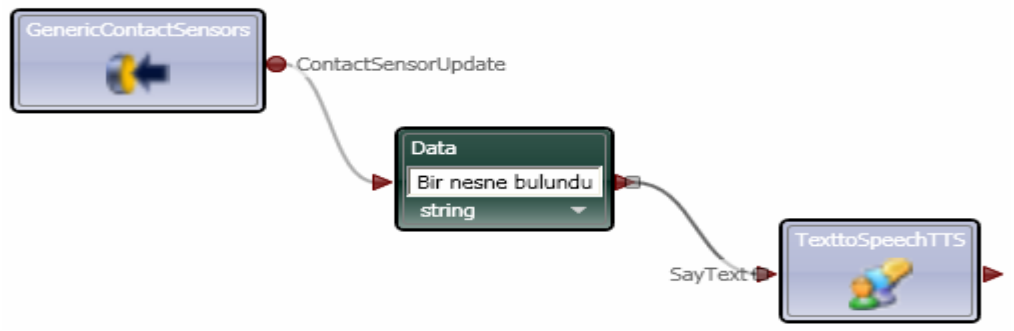
İki bloğu birbirine bağlamak için, Şekil 6.12’de gösterildiği gibi generic contact sensors bloğunun sağındaki yuvarlak “Bildiri” (Notification) bağlayıcısından mouse ile tutarak data bloğunun solundaki giriş bloğuna bir çizgi şeklinde sürüklenir. Bu aksiyon bağlantı diyalog kutusunu başlatır ve yapılacak olay prosedürünün seçilmesini sağlar. Açılan kutudan varsayılan değerler olduğu gibi bırakılır. Bu işlem gerçekleştirildikten sonra bağlantı diyalog kutusunu kapatmak ve seçilen işlemi onaylamak için “Tamam” tıklanır.



Şekil 6.12 Generic contact sensors servisi ile Data aktivitesi arasındaki bağlantı

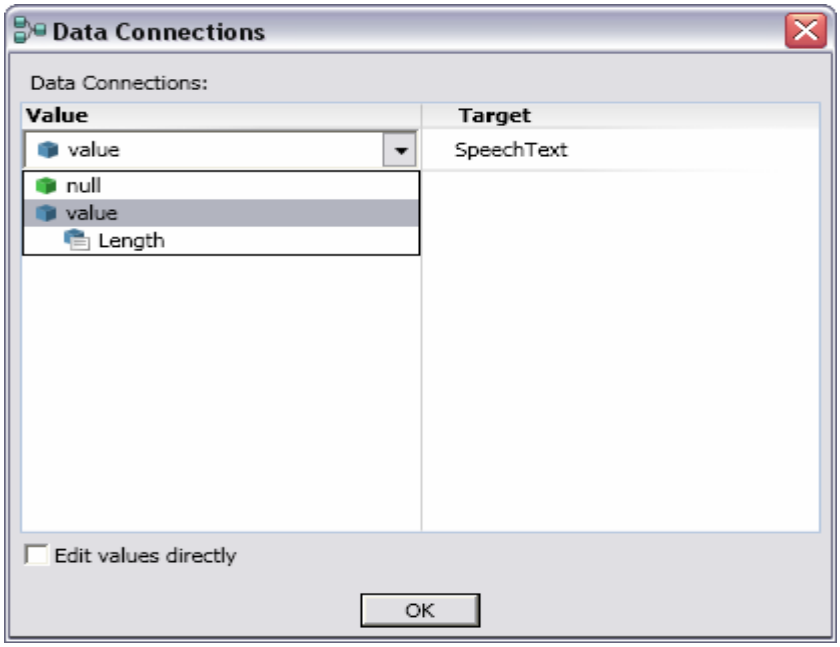
Bir sonraki adımda diyagram üzerine ve data bloğunun sağına “Text to Speech” servisi yerleştirilir. Bu servis, Konuşma Uygulama Programlaması Arayüzü (Speech Application Programming Interface [SAPI]) aracılığıyla uyumlu olarak, bir text’i konuşma olarak söylemeye ilişkin birçok fonksiyonu destekler.

Data bloğunu text to speech hizmetine bağlamak için, Şekil 6.13’de görüldüğü gibi data bloğunun sağındaki üçgenden mouse ile tutup sürükleyerek text to speech bloğunun sol tarafına doğru götürülerek bağlantı oluşturulur. Bu bağlantı bir diyalog kutunu açar ve bir değer seçilmesini sağlar.



Şekil 6.13 Bloklar arası bağlantı

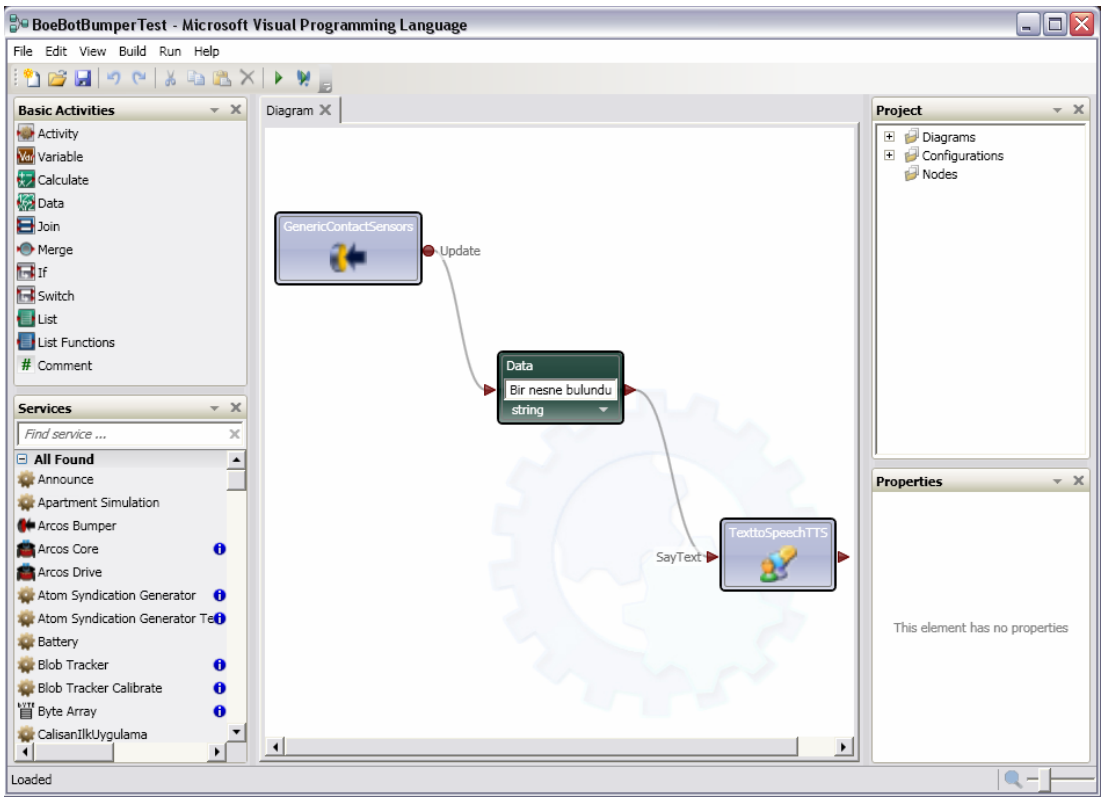
Text'in konuşma işlemini yapabilmesi için açılan diyalog kutusundan "SayText" seçilir ve "Tamam"a tıklanır. Bu işlem yağıldıktan sonra verileri ilgili değişkenlere atama yapmak için Şekil 6.14'de görülen "Data Connections" diyalog kutusu açılacaktır. Varsayılan olarak seçilmiş değer "null" yani sıfırdır. Açılır liste kutusundan "value" değeri seçilir ve "Tamam"a tıklanır. Aksi halde hiçbir konuşma duyulmayacaktır.



Şekil 6.14 Data Connections diyalog kutusu

6.8 Bir VPL Uygulamasını Çalıştırmak ve Hata Ayıklamak

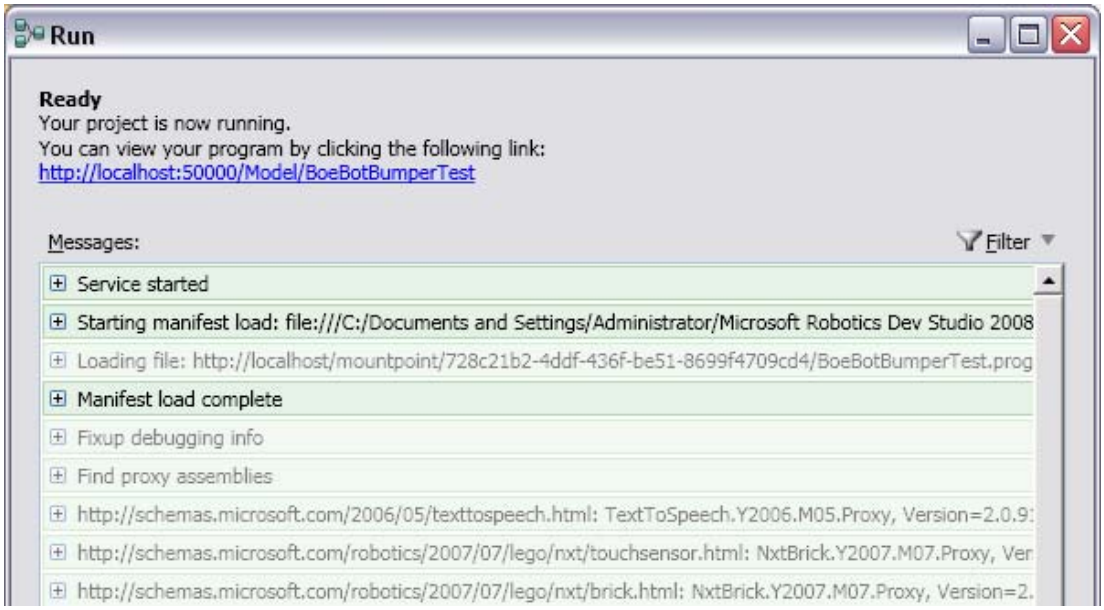
Bütün işlemler tamamlandıktan sonra, VPL uygulaması Şekil 6.15'deki gibi olacaktır. Uygulama "mvpl" dosya uzantısı olarak kaydedilir. Uygulama "BoeBotBumperTest.mvpl" diye adlandırarak kaydedilir.



Şekil 6.15 Basit bir uygulamanın VPL görüntüsü

Uygulamayı çalıştırmak için, "Run" menüsünden "Start" tıklanır veya F5 tuşuna basılarak başlatılır. Bir başka alternatif olarak, üstteki menü çubuğundan yeşil ok ikonunu tıklanarak da çalıştırılabilir.

Bu işlem Şekil 6.16'da görüldüğü gibi "Run" diyalog kutusunu başlatır. DSS modülü varsayılan port ayarlarını http portu için 50000 ve TCP portu için 50001 olarak kullanmaktadır.

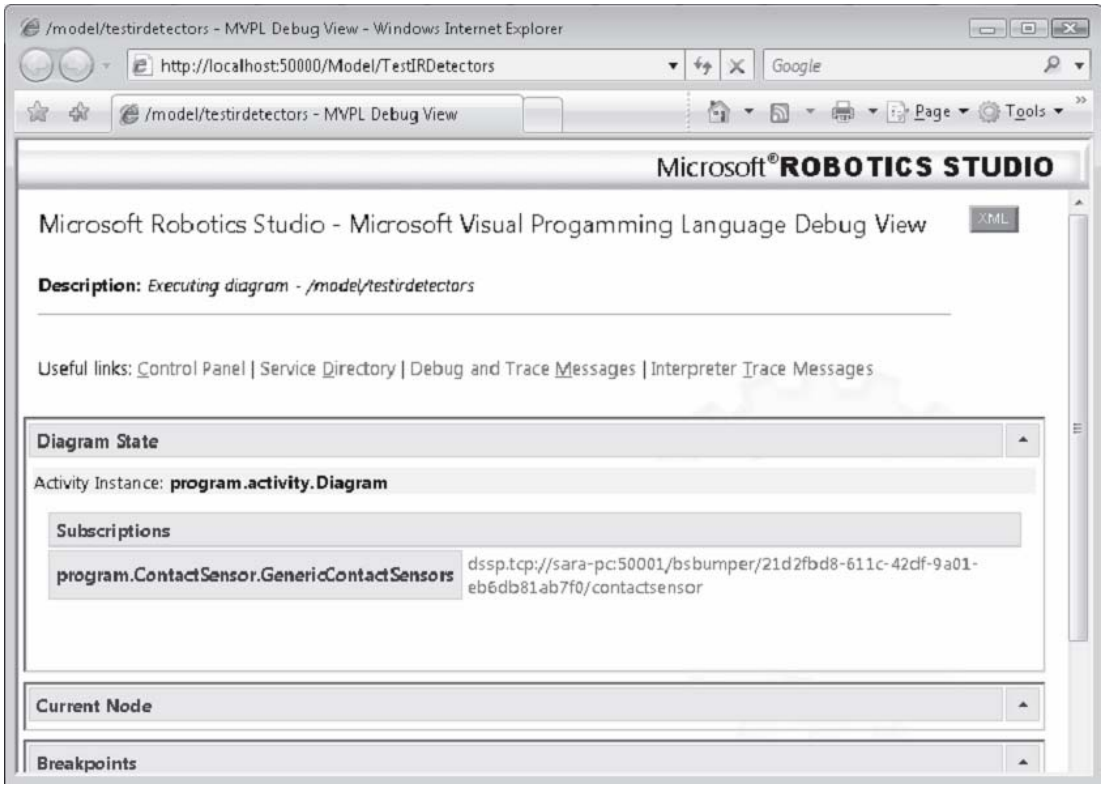


Şekil 6.16 Run diyalog kutusu

Bağlantı kurulduktan sonra, uygulama işlemi başlar. Bu süreç birkaç saniye alır. Bağlantı gerçekleştiği zaman Boe-Bot robotun yeşil ışığı yanar ve hoparlörden 2 defa beep sesi gelir. Artık, Boe-Bot'un üzerindeki kontak sensörleri tetiklenerek uygulama test edilebilir. Kontaklardan her hangi birinden bir nesne algılandığında bilgisayar hoparlöründen "Bir nesne bulundu" sesi duyulacaktır.

Uygulamayı internet explorer'dan takip etmek için "Run" diyalog kutusundaki "http://localhost:50000/Model/BoeBotBumperTest" linke tıklayarak veya bu linki internet explorer adres çubuğuna yazarak takip edilebilir.

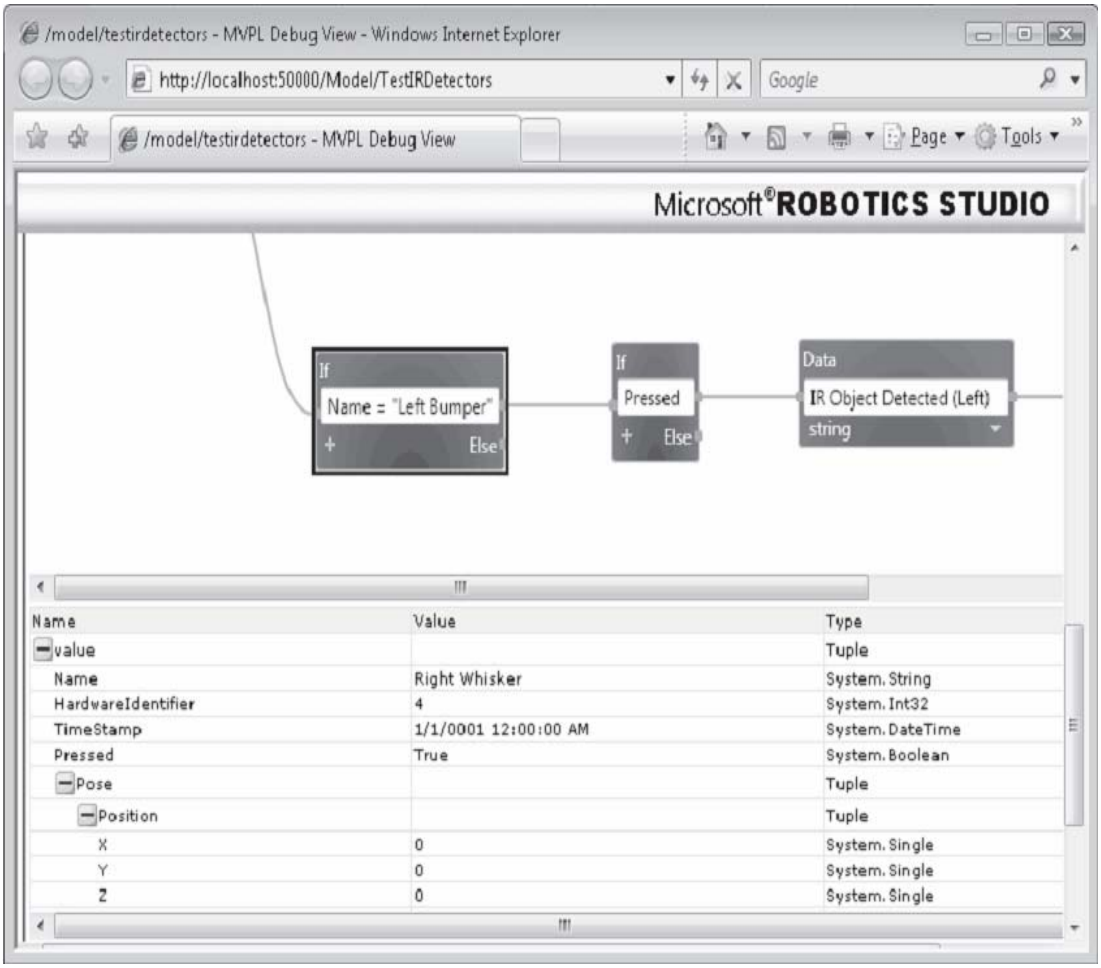
Uygulamayı hata ayıklama modunda çalıştırmak için "Run" menüsünden "Debug Start" işlemi seçilmesi gerekir veya F10 tuşuna basılarak da başlatılabilir. Bu yine "Run" diyalog kutusunu başlatır. Aynı şekilde uygulama linki tıklandığında Şekil 6.17'deki gibi internet explorer da Microsoft VPL hata ayıklama ekranı görülür.



Şekil 6.17 Microsoft VPL hata ayıklama (Debug) ekranı

Internet explorer içinde bulundurulan MVPL hata ayıklama görünümü, aşağıdaki 4 seçeneği içerir.

- **Diyagram Durumu:** VPL'deki programın durumunu gösterir. Ve kurulmuş olan her partner aboneliklerini listeler.
- **Güncel Mod:** VPL diyagramını gösterir ve programın durumunu belirtir. Ok düğmeleri bloklara girmeyi sağlar ve program oluşturulurken kullanılan değişken durumlarının değerlerini incelememizi sağlar. Örneğin, Şekil 6.18'de Boe-Bot bumper test programı oluşturulurken kullanılan değişken durumların değerini gösterir.



Şekil 6.18 Değişkenlerin incelendiği Güncel Mod ekranı

- **Kesme Noktası (Breakpoints):** Program çalışırken, kesme noktaları oluşturularak "Bekleme Modu"na geçer ve program bu noktalarda durdurulur. Bu özellikle birçok blok içeren VPL programını hatalarından arındırırken çok yararlı olmaktadır. Kesme noktası bölümünden, kesme noktalarını silebilir, etkinleştirebilir veya etkisizleştirebilirsiniz.
- **Bekleme Modu:** Güncel örneğin bekleme mod'larını listeler. Bu şekilde istenen mod'un yanındaki SetBP düğmesi tıklanarak kesme noktalarını kurulabilir.

6.9 Bir Servis Olarak Derlemek

VPL'nin iyi özelliklerinden birisi de uygulamaları bir servis olarak derleyebilme yeteneğidir. Maalesef tersi durum olan bir servisi tekrar VPL uygulamasına dönüştürebilecek hiçbir yol yoktur.

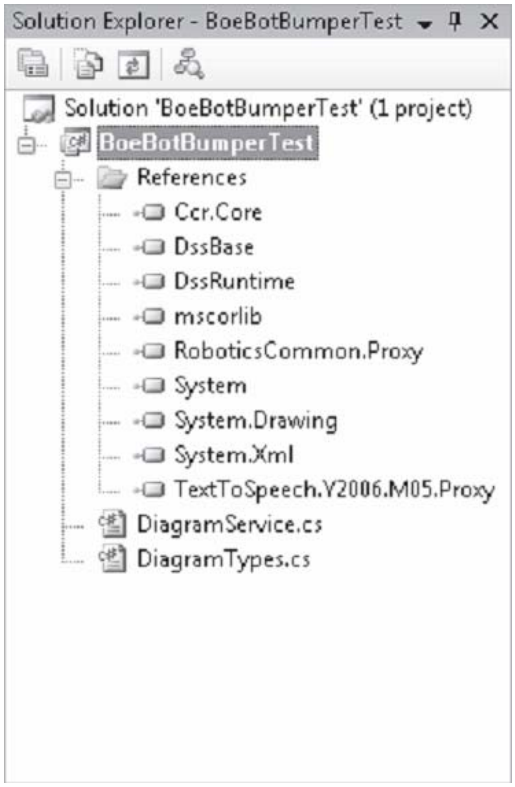
Bir uygulamayı servis olarak derlemek başka uygulamalarda vakit kaybetmeden tekrar kullanabilmek adına çok yararlıdır. VPL uygulamasını bir servis olarak derledikten sonra "Services" araç kutusu listesinden seçim yapılarak kullanılabilir.

Uygulamayı bir servis olarak derlemek için, "Build" menüsünden "Compile as a service" (Bir servis olarak derle) işlemi tıklanarak kolayca yapılabilir. Bu VPL uygulamasıyla birlikte birçok kodu, Visual Stüdyo'da açılabilen Visual C# (C Sharp) projesi oluşturur. Bu işlem genel programlamaya ve MSRS'ye yeni olan programcılar için çok faydalıdır. VPL'nin içindeki blokların bir servis oluşturmak için gereken kodla ilişkisinin nasıl olduğunu anlamak için iyi bir imkân sağlar.

Örneğin "BoeBotBumperTest" uygulamasını bir servis olarak derlersek, aşağıdaki 2 sınıf dosyayı içeren bir Visual Stüdyo projesi oluşturulur.

- **DiagramService.cs:** Bu dosya uygulama sınıfıdır. Ve uygulamayı çalıştıran kodun yerleştirildiği yerdir. Burası aynı zamanda mesaj işleyicilerinin abonelere bildirilerin üretilmesinden sorumlu olduğu yerdir.
- **DiagramTypes.cs:** Bu dosya ifadeleri işleyen yazı tanımlarının yerleştiği yerdir. PortSet'in de yerleştiği yerdir. PortSet DSSP operasyonlarının ne çeşitlerde olduğunu belirten yerdir.

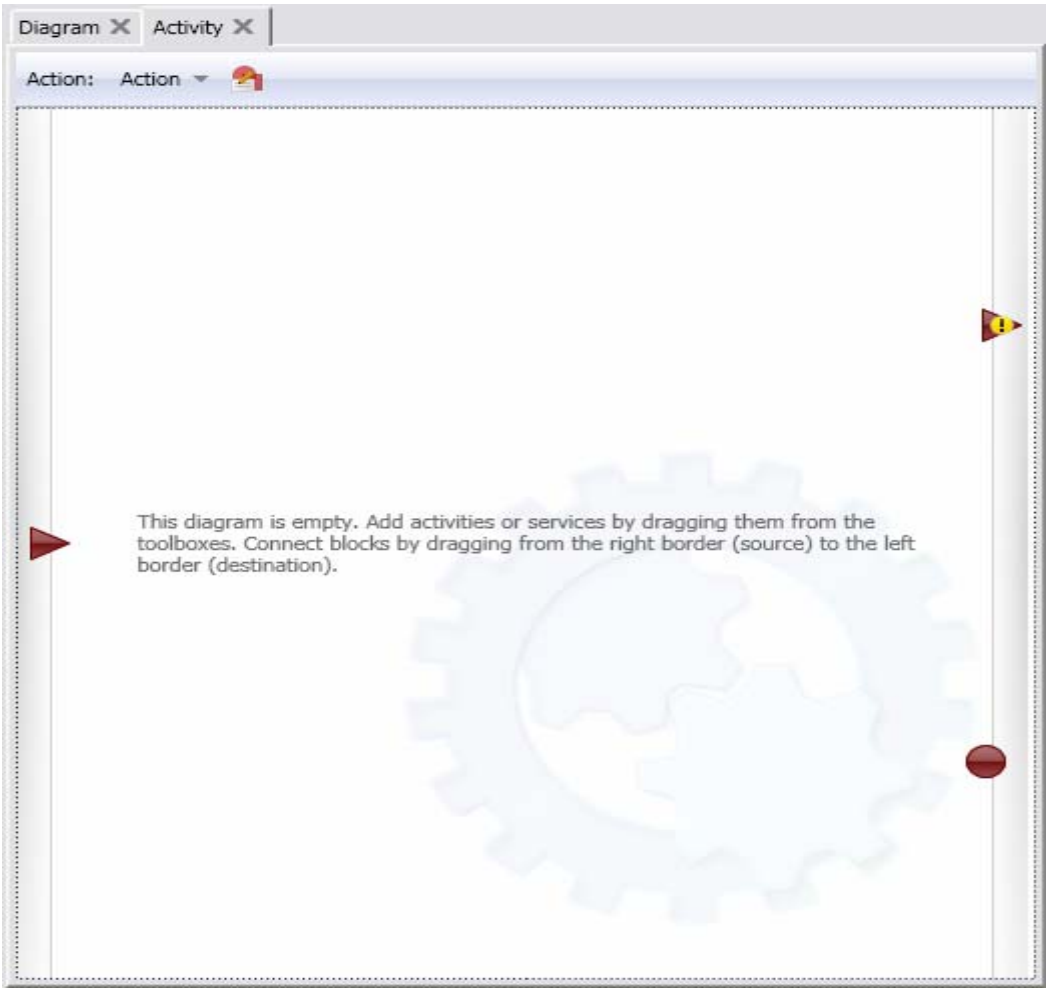
Proje Visual Stüdyo'da açılırsa, Şekil 6.19'da görüldüğü gibi "Solution Explorer" bölümündeki "References" klasörünün içerisine programa ilişkin referanslar görünür. RoboticsCommon.Proxy ve TextToSpeech.Y2006.M05.Proxy dosyaları Boe-Bot bumper test uygulamasının içerdiği iki servise giden çağrıları ileten proxy dosyalarıdır. RoboticsCommon.Proxy dosyası MSRS'ninde mevcut olduğu bütün "generic service" sınıflarını kapsar.



Şekil 6.19 Servis olarak derlenen VPL'nin Visual Stüdyo'daki referans dosyaları

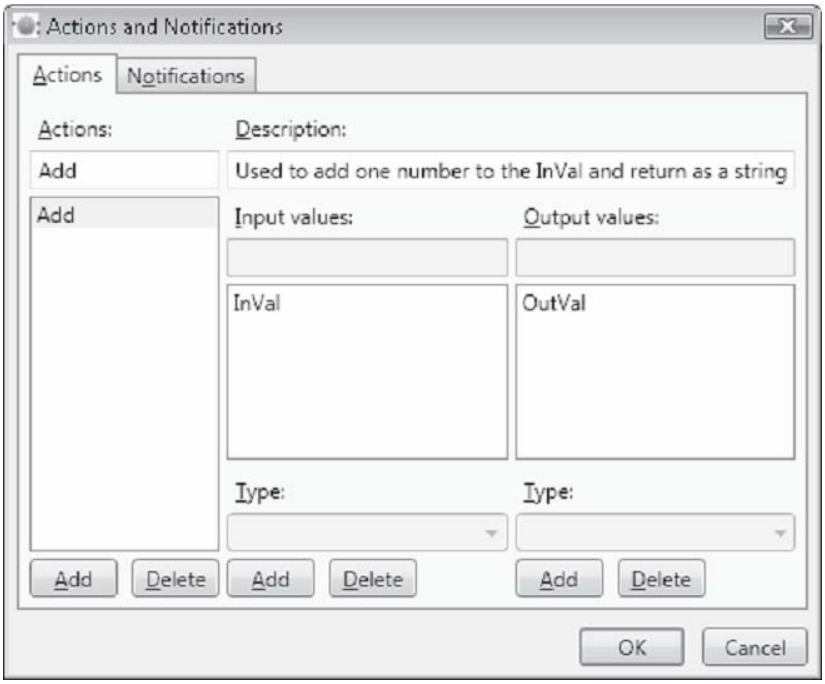
6.10 Özel Bir Activite Oluşturmak

VPL, programcılara servis ve aktiviteler gibi bir çok işi yapan blokları tek bir blok şeklinde, dizayn yüzeyine sürükle-bırak şeklinde kullanılan özel aktiviteler yapma izni verir. Yeni bir activite oluşturmak için "Basic Activities" kutusundaki "Activity" isimli temel activite diyagram yüzeyine yerleştirilir. Sonra çift tıklanarak içerisine girilir. Şekil 6.20'de görüldüğü gibi activity dizayn yüzeyi ana diyagramdan biraz farklı gözükür. Şekil 6.20'de sol kenardaki üçgen girişi ve sağdaki üçgende çıkışı temsil eder. Sağ kenardaki daire ise bildirimlerle (notification) ilgili çıkışı temsil etmektedir ve bir activity'nin bir ya da birden çok ilişkili bildirimi olabilir.

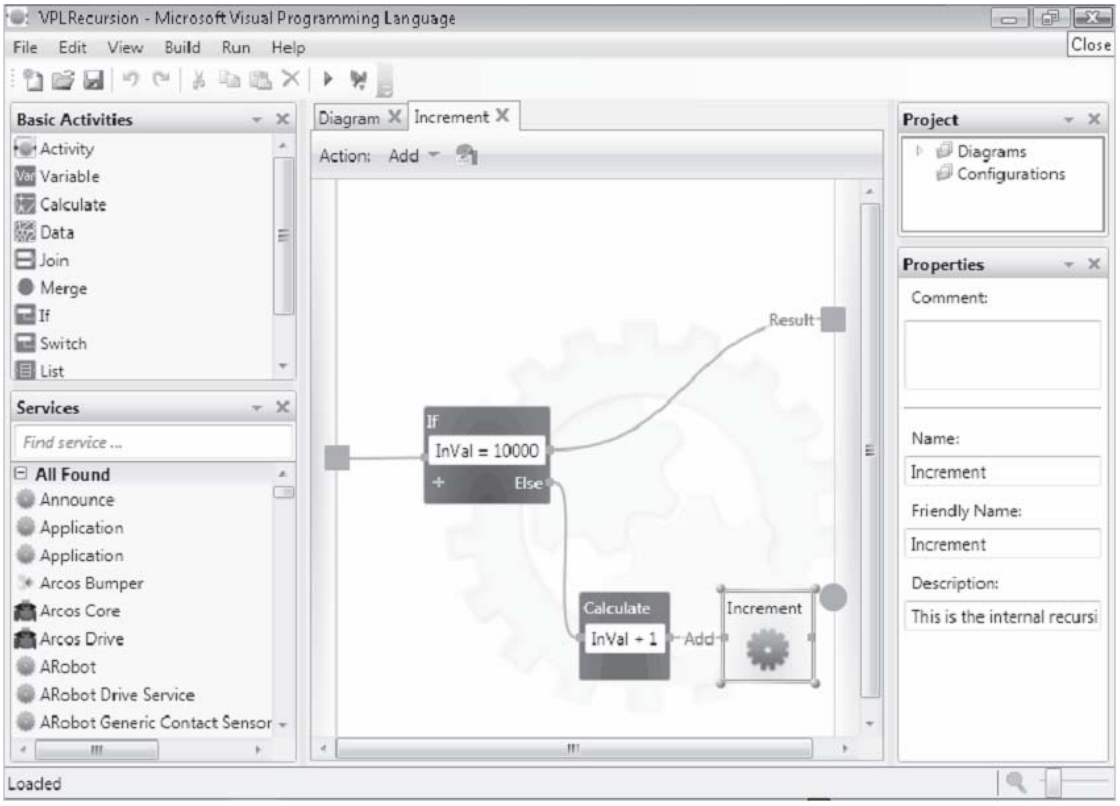


Şekil 6.20 Activity diyagram ekranı

Her activite için bir ya da daha fazla aksiyon tanımlanabilir. Örneğin Ekleme ile Güncelleme isimli iki aksiyon aynı aktivitede bulunabilir. Şekil 6.21'de görülen "Actions and Notifications" diyalog kutusu activity diyagramının sol üst köşesindeki açılır menüsünün "Action" isminin yanındaki ikon tıklanarak ulaşılır. Aksiyonlar kullanıcı tanımlıdır ve ne istenirse o ismi verilebilir. Her aksiyonun bağlantılı olan bir ya da daha çok giriş ve çıkış değerleri olabilir. Activite ana diyagramdaki başka bir bloğa bağlandığında aksiyon seçilmiş olur. Şekil 6.22'de örnek olarak verilen bir özel activity uygulaması görülmektedir.



Şekil 6.21 Actions and Notifications diyalog kutusu



Şekil 6.22 Örnek bir özel aktivite uygulaması

7 SİMÜLASYON

MSRS açısından simülasyon, bir veya daha fazla fiziksel nesneyi temsil etmede kullanılan matematiksel bilgisayar modelidir. Yeni çıkan yazılım oyunlarına simülasyon ortamı oldukça benzemektedir.

Bu bölüm görsel araçla ilgili olmasına rağmen, VPL bölümündekine benzer değildir. VPL, esasen tasarım alanına blokları sürükleyip bırakmayı gerektirir. Bu robotik uygulamaları oldukça kolaylaştırır. Çünkü kodun çoğunluğu araç tarafından otomatik olarak üretilir.

Bir simülasyon oluşturmak için, ilk önce dağıtılmış yazılım servisini (DSS) oluşturulur ve bu servis simülasyon motorunu yardımcı servis olarak kullanır. Tez çalışmasının bu bölümünde, bir simülasyonun nasıl yapıldığı, entity'lerle nasıl çalışıldığı, MSRS tarafından sağlanan grafik editörüyle çalışmayı ve aynı zamanda DSS servisini kullanarak bir simülasyon yapılması anlatılmıştır.

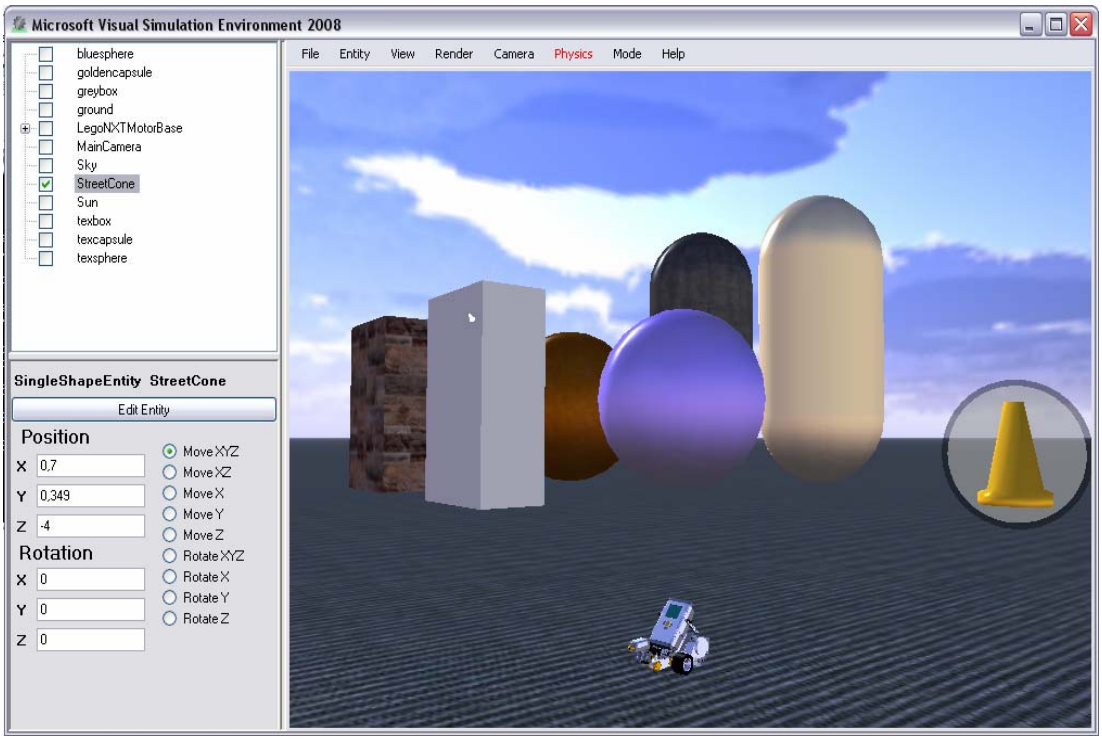
7.1 Simülasyonların Yapılma Amacı

Robotlar için simülasyon kullanmanın en büyük avantajı, gerçek bir robota ihtiyaç duyulmaması ve her şeyin sanal olmasıdır. Programcılar için pahalı robotlarla çalışmak yerine simülasyon büyük avantaj olabilir. Bu özellikle bir programcı grubunun çalışmasında her bir programcının robotla çalışması gerektiğinde daha açık ortaya çıkmaktadır. Simülasyon motoruyla, her bir programcı kendi simülasyon servisini oluşturur ve robotla bağımsız olarak çalışabilir. Programcı deneyleri yapabilir ve robotun farklı senaryolara tepkisini gözlemleyebilir.

Bazı robotlar oldukça pahalı olabilir, programcının ilk deneyini gerçek robottan ziyade simülasyonda yapması gerçek robota zarar verme riskini ortadan kaldıracaktır. Simülasyon sayesinde programcı gerçek robota yaptıracağı uygulamayı simülasyonda kod yazarak yaptıktan sonra aynı kodu sadece küçük bazı değişikliklerle gerçek çalışmada kullanabilir.

7.2 Görsel Simülasyon Ortamının Kullanılması

Simülasyon motoru sadece bir servis olmasına rağmen, MSRS simülasyon sahnesinde yeni entity (nesne) oluşturmakta ve düzenlemekte kullanılan bir editör sağlamaktadır. VSE olarak adlandırılan grafik editörü iki temel moda çalışır: çalışma (run) ve düzenleme (edit) modları. Çalışma modunda; simülasyon çalışır fakat simülasyon sahnesine etki edebilecek herhangi bir özelliğin değiştirilmesine izin vermez. Şekil 7.1'de gösterilen düzenleme mod mevcut entity'lerin özelliklerinin değiştirilmesine veya yeni entity'ler eklenerek mevcut sahneyi değiştirmeye izin verir.

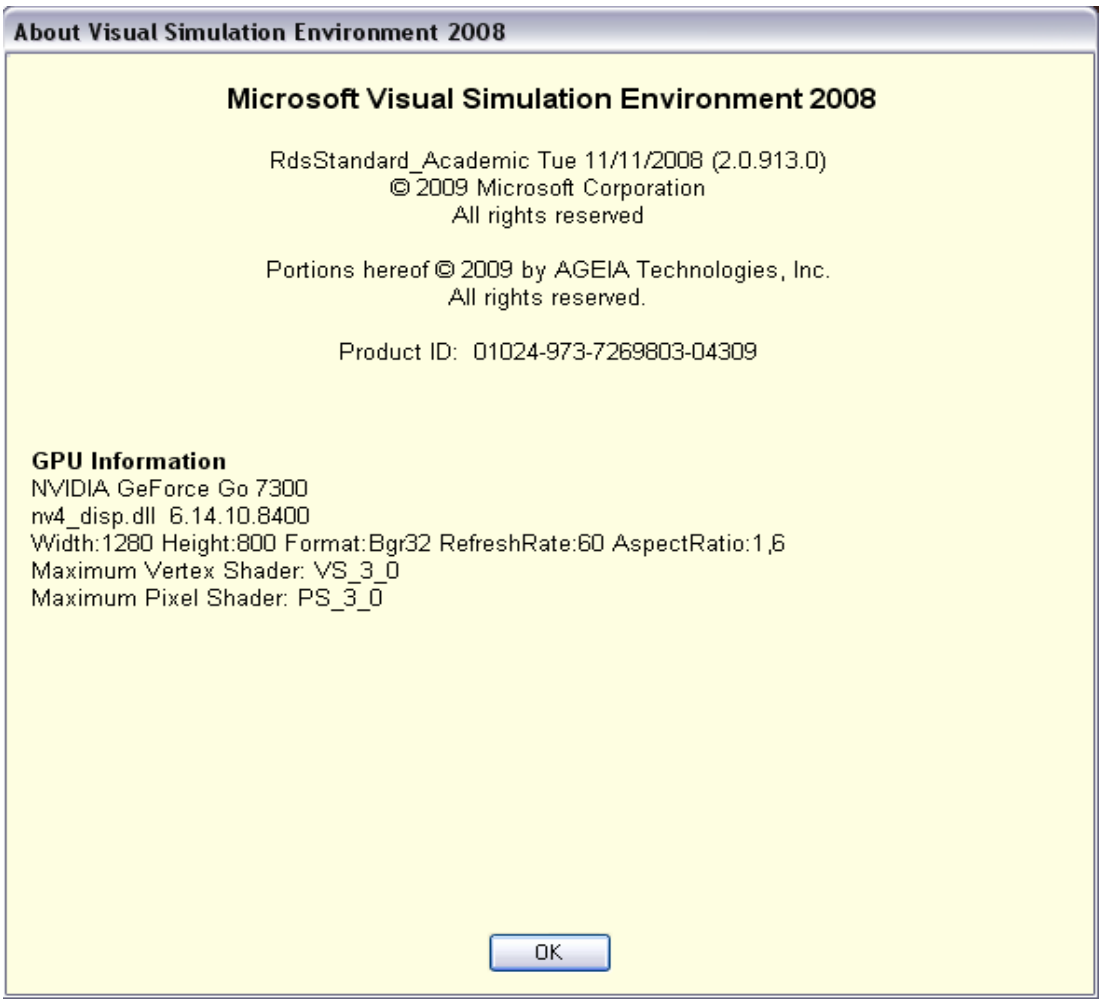


Şekil 7.1 Düzenleme modunda simülasyon sahnesinin görüntüsü

7.2.1 Donanım gereksinimleri

Simülasyon yapmak ve VSE kullanmak için, bilgisayarın uyumlu bir grafik kartı olmalıdır. Grafik kartı, DirectX sürüm 9 veya daha fazlasına uyumlu olmalıdır.

VSE'lerin herhangi birini örneğin temel simülasyon ortamını açtıktan sonra yardım menüsünü altında bulunan "MVSE Hakkında" kısmını seçilirse Şekil 7.2'deki gösterildiği gibi grafik kartı ile ilgili bilgiler görülebilir.



Şekil 7.2 MVSE hakkında diyalog kutusu.

Eğer fonksiyonlar donanım tarafından yerine getirilirse, grafik hızlı bir şekilde oluşur. Eğer donanım grafik fonksiyonlarını desteklemezse, simülasyon yazılım modunda çalışır. Genelde, yazılım donanımdan daha yavaş çalışır ve önemli miktarda performans kaybı görülür.

7.2.2 Saniye başına çerçeveleri izlemek

VSE, bir opsiyonel durum çubuğunu kullanarak çerçeveleri izlemeye olanak sağlar. Her saniye başına çerçeveler (FPS) ayarı, bilgisayarın ve grafik kartının yeteneklerine oldukça bağlıdır. Aynı zamanda, simülasyondaki objelerin karmaşıklığına ve entity'lerin sayısına bağlıdır. Basit bir şekil entity'si birkaç poligona sahipken, daha karmaşık bir entity robot gibi binlerce poligona sahiptir.

VSE kullanılırken, FPS ayarı monitörün oranıyla sınırlıdır. Grafik kartı yükseltilse ve daha fazla hafıza kullanılsa bile, saniyede yaklaşık 58 çerçeveden daha fazlasını vermeyecektir.

Şekil 7.3, küreler, kapsüller ve kutular gibi birçok basit şekilli nesnelerden oluşmaktadır. Aynı zamanda tablolar, robotlar hatta arazinin kendisi gibi daha karmaşık nesneleri de içermektedir. Bu nesnelerin nasıl yapıldığı konusunda fikir sahibi olmak için, simülasyon çalıştırılır ve Render menü aracı kullanılarak bir wireframe renderlama oluşturulur. Wireframe renderlama, tüm köşelerdeki hatları gösterir, fakat bunların içlerini doldurmaz. Bu entity formunda bulunan poligon şekillerinin tamamını gösterir.



Şekil 7.3 MSRS tarafından gerçekleştirilen arazili simülasyon görüntüsü

Simülasyonu daha hızlı çalıştırmanın tek yolu, render menü aracı seçildikten sonra no rendering seçeneğinin seçilmesi gerekmektedir. Renderlanmış sonuçları görmemenize rağmen, Log Messages özelliği kullanılarak robotu kontrol ederken mesajlar kaydedilir.

7.3 Simülasyon Çalıştırma

Bir simülasyon çalıştırılır ve DssNode yürütülerek VSE başlatılır. Sonra simülasyon motor servisini içeren manifest dosyası yüklenir. Manifest dosyası servis contracts listesini sağlar. Bu servis contracts'ları diğer servislerin hedef servisle çalışmak için ihtiyaç duydukları bilgileri içermektedir. Bir manifest dosyası örnekte olduğu gibi simülasyon servisini partner modülünü içerecek şekilde yükler.

```
<Partner>
<Contract>http://schemas.microsoft.com/robotics/2006/04/simu
lationengine.html
</Contract>
<Service>IRobot.Create.Simulation.cml</Service>
<Name>StateService</Name>
</Partner>
```

MSRS'nin Microsoft Robotics Studio menüsünde ve görsel simülasyon ortamı (VSE) dosyasında bulunan birkaç simülasyon içerir. Bu simülasyonlar VSE ile nasıl çalışılacağını gösteren ve MSRS tarafından desteklenen entity nesnelerinin kullanılarak simülasyon yapmayı gösteren değişik senaryoları temsil etmektedir. MSRS V.1.5 aşağıdaki simülasyonları içermektedir.

- **Temel Simülasyon Ortamı:** Temel simülasyon sadece birkaç entity içerir, bu entity'ler dünya gibi görünen büyük bir küre ve küçük bir gri kutu içermektedir. Bu simülasyonda robot bulunmamaktadır. Bunun sebebi ise VSE ile çalışmanın temellerinin öğretilmesi hedeflenmektedir. Çünkü sahne birkaç basit nesne içermektedir ve gökyüzü ile yeryüzü entity'lerini içermektedir, kullanıcılara kendi simülasyonlarını oluşturmada yardımcı olacaktır.

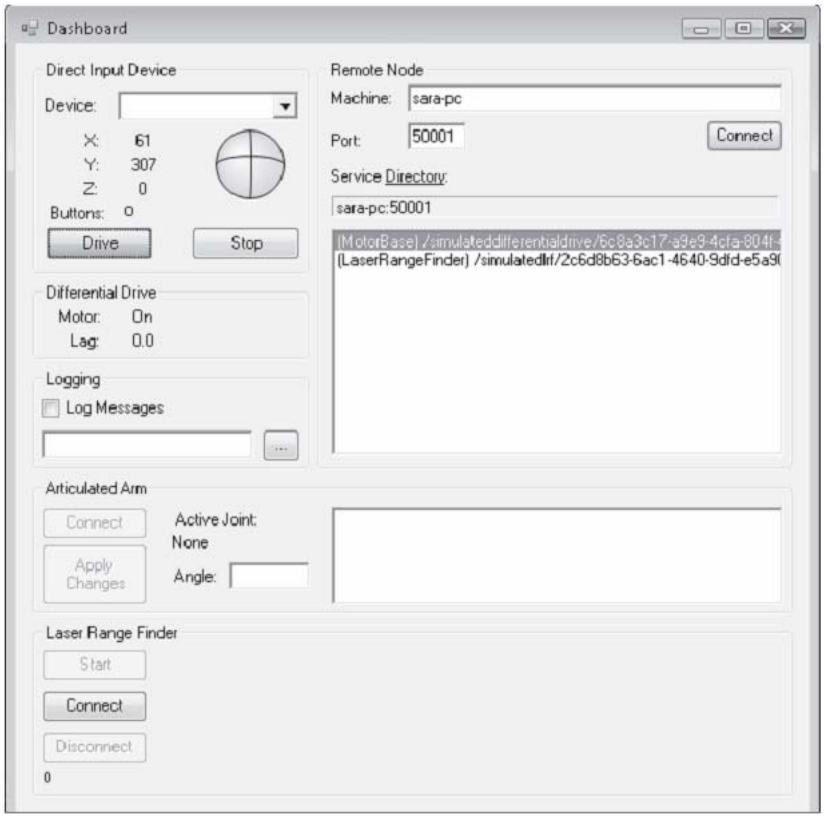
- **IRobot Oluşturma Simülasyonu:** Bu simülasyon sadece bir robot içermektedir. Bu simülasyonun komutları iki manifest dosyasına yüklenmiştir: iRobot.Create.Simulation.Manifest.xml ve SimpleDashboard.Manifest.xml. Simple Dashboard, simülasyonla birlikte sağlanan servisleri kullanarak robotları kontrol etmenizi sağlayan MSRS servislerinden oluşmaktadır.
- **KUKA LBR3 Kol Simülasyonu:** Bu simülasyon KUKA LBR3 kolu ve domino gibi birkaç büyük kutudan oluşmaktadır. Altı eksenli robotik kol Simple Dashboardla her bir birleşim noktasının açılarını ayarlayarak kullanılma ile kolu hareket ettirmeyi sağlar.
- **LEGO NXT Tribot Simülasyonu:** Bu simülasyon LEGO NXT Tribot ve küreler, kapsüller gibi birkaç basit nesneden oluşmaktadır. Bu simülasyonun kaynak koduna "http://msdn2.microsoft.com/en-us/library/bb483043.aspx" adresinden ulaşılabilir.
- **Çoklu Simülasyonlu Robotlar:** Bu simülasyon bir masayla birlikte Pioneer 3DX ve LEGO NXT Tribot robotları içermektedir. Simple Dashboard kullanılarak ve her robotla birlikte servisler sürülerek, robotlar ortam etrafında sürülebilir.
- **Pioneer 3DX Simülasyonu:** Bu simülasyon Pioneer 3DX ve kutular, kapsüller ve küreler gibi birkaç basit entity'lerden oluşmaktadır. Bu ilave entity'ler robota engel olarak rol almaktadırlar. Bu simülasyonda Pioneer 3DX ile birlikte gelen lazer rangefinder aracı kullanılarak etkiler görülebilir. Simülasyon çalışırken, lazer rangefinderin görüş alanındaki engeller kırmızı yanacaklardır. Simple Dashboard servisi robottan gelen verileri izlemek için kullanılabilir.
- **Arazili Simülasyon Ortamı:** Bu simülasyonda tepelik bir arazi bulunmaktadır. Pioneer 3DX robotu simülasyonda bulunmaktadır fakat burada robot diğer simülasyonlardakilerden farklıdır. Çünkü entity ile birlikte gelen mesh nesne dosyası burada çıkarılmıştır. Robot yine aynı robot olmasına rağmen, mesh dosyası olmaksızın gerçekçi görünmemektedir.

Simülasyonla çalışırken, bilmeniz gereken ilk şey komut ekranının görünmesidir. Simülasyon komut ekranını DssNode'u başlatmak için ve aynı

zamanda bir veya daha fazla manifest dosyasını yüklemek için kullanır. Bu olay simülasyon motorunu yükler ve VSE'yi başlatır. VSE çalışma modunda iken simülasyonu başlatır ve sahneyi üç boyutlu katı bir görüntü olarak render eder. Alternatif olarak, sahne physic'lerle ve wireframe'le render edilebilir. Eğer simülasyonda çok sayıda entity bulunuyorsa render'lama kapatılabilir.

7.4 Simple Dashboard Servisinin Kullanılması

MSRS farklı türdeki robotlarla çalışmaya olanak sağlar. Şekil 7.4'de görülen Simple Dashboard servisi Windows tabanlı bir diyalog kutusu başlatır. Bu DSS modülü ile bağlantı yapmayı ve bu modül üzerinde bir veya daha fazla servisi başlatmayı sağlar. Dashboard simülasyon çalıştığında otomatik olarak ortaya çıkar. MSRS tarafından sağlanan simülasyonların çoğu Dashboard içermektedir.



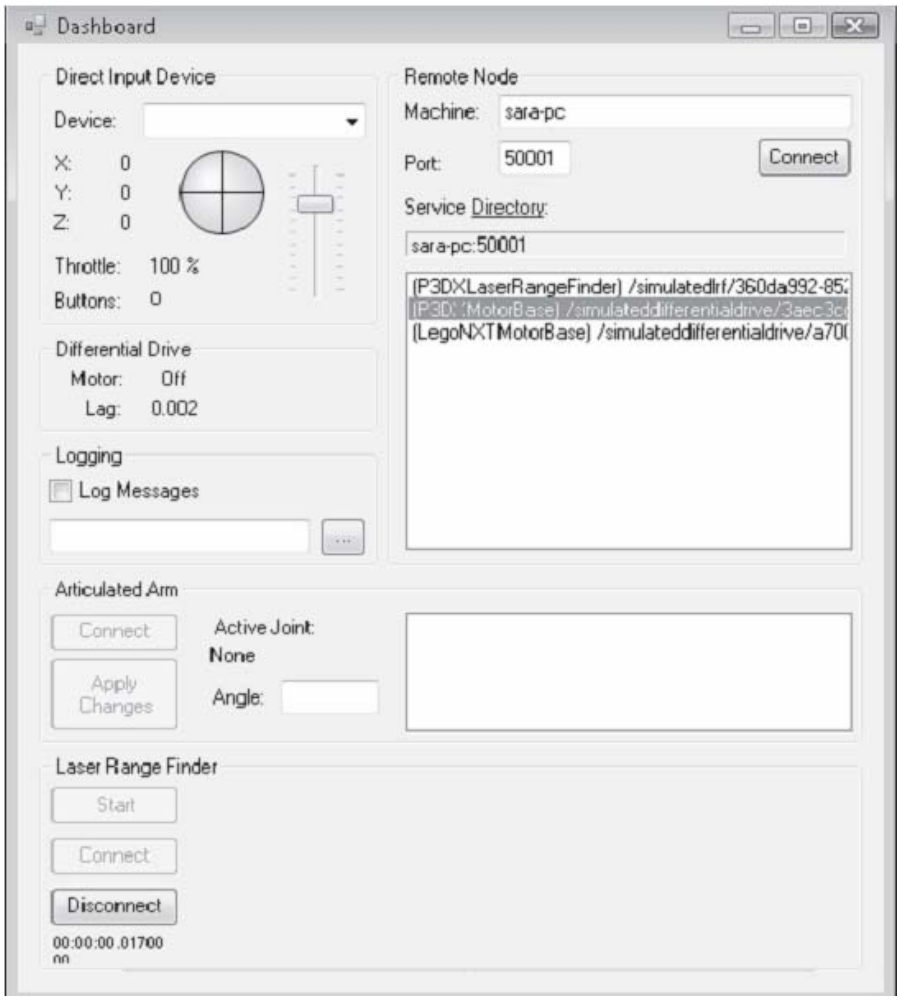
Şekil 7.4 MSRS tarafından sağlanan Dashboard servisi.

Bilgisayardan DSS modülüne bağlanmak için, Dashboard diyalog kutusu için bilgisayardaki metin kutusuna “localhost” ifadesini girilir. Sonra connect’e tıklanır ve bu modülde çalışan servislerin listesi list box’ta görünür. Bu servisler, simülasyon başlatıldığı zaman manifest dosyası tarafından yüklenir. Seçilen servise bağlı olarak, izleyen fonksiyonları yerine getirilebilir.

- **Robot sürme**: Lego NXT, iRobot Create ve Pioneer 3DX gibi diferansiyel sürme sistemli robotlar için, simülasyonlu diferansiyel sürme servisi robotu sürmek için kullanılır. Bunu yapmak için, servisler listesinden sürme servisinin seçilmesi gerekmektedir. Sonra diyalog kutusunun sol tarafında bulunan sürme butonuna tıklanır. Bundan sonra mouse, joystick gibi araçları kullanarak robot kontrol edilir.
- **Eklemli kolu hareket ettirme:** KUKA LBR3 gibi robotlarda, simülasyonla her bir aktif birleşme noktasının açısını değiştirerek kol hareket ettirilir. Aynı şekilde Simple Dashboard diyalog kutusunu kullanılarak kol kontrol edilebilir. KUKA LBR3 simülasyonunu, servis listesinden simüle edilmiş LBR3 kol servisi seçilerek kontrol edilebilir.
- **Lazer rangefinder aracından gelen verileri izleme:** Lazer rangefinder içeren robotlar, Pioneer 3DX gibi, cihazdan gelen sonuçlar rahatlıkla izlenebilir. Sonuçlar robotun bir engelle karşılaşıp karşılaşmadığını ve robotla engel arasındaki mesafeyi ölçmede kullanılmaktadır.
- **Log mesajlar:** Simple Dashboard aracılığıyla simülasyon çalışırken log mesajları kullanılabilir. Bunu yapmak için, log dosyaya bir yol belirlenmesi gerekmektedir ve MSRS kurulumunun bir parçası olan depolama alanına log dosyası depolanmak zorundadır. Sonuçta XML tabanlı dosya DSS operasyonları tarafından yapılan ifadeleri içerir. Simülasyon çalıştığı sürece, veri log dosyasına yazılacaktır. Log dosyasına kaydedilen veri geri dönülmesini ve robot sensörlerinden geri dönen hassas değerleri görmeyi sağlar. Örneğin, lazer rangefinder çalışırken log mesajlar üretilebilir ve sonra geri giderek simülasyon çalışırken elde edilen kesin ölçüm sonuçlarına erişilebilir.

7.4.1 Simple Dashboard servisinin gelişmiş versiyonu

MSRS, Simple Dashboard Servisi gibi servisleri içerdiği için, fonksiyonları arttırmak mümkündür. Şekil 7.5'te Simple Dashboard servisinin gelişmiş versiyonu bulunmaktadır. Bu servis http://www.benaxelrod.com/MSRS/index.html adresinden ücretsiz olarak indirilebilir. İndirilen dosyanın içerikleri "\samples\misc folder"a kopyalanmalıdır. Dosyalar kopyalandıktan sonra, Visual Studio kullanılarak çözüm dosyası açılması gerekmektedir ve çözüm tekrar yapılır.



Şekil 7.5 Simple Dashboard Servisinin gelişmiş bir versiyonu.

Gelişmiş Simple Dashboard servisi zaman tasarrufu sağlayan birkaç servis sağlamaktadır. Aynı zamanda doğrudan Simple Dashboard servisi, robotları diferansiyel sürme sistemiyle sürmek için kullanılan doğrudan giriş cihazı daha gelişmiş özelliklere sahiptir. Orijinal dashboard servisi robot sürmede joystick kullanıldığı zaman %60 güçle sınırlandırılmıştır. Gelişmiş versiyon %100 güçte sürülmesini sağlar. Robotu mouse ve joystick ile kontrol etmenin yanında klavye ve klavye oklarıyla kontrol edebilmenizi de sağlamaktadır.

Geliştirmeler aynı zamanda lazer rangefinder bölümünde de yapılmıştır ve geliştirilmiş servis özellikleri iki görüntülüdür: silindir görüntüsü ve tepeden görüntü. Yeşil noktalar ileri hedefleri ve kırmızı noktalı engeller engellerin robotun önünde olduğunu gösterir.

7.5 Editör Ayarları

VSE grafik ayarlarını güncellemeye izin verir. DSS servisinde kod kullanarak ek ayarlamalar yapabilmenize rağmen, Şekil 7.6'da gösterilen grafik ayarları yaygın olarak kullanılan parlaklık veya sahnenin kalite seviyesi gibi ayarlamalar için kullanılır. Grafik ayarları, aşağıdakileri içermektedir ve Render menü aracından ulaşılır.

- **Exposure:** Simülasyon sahnesinin parlaklığına etki eder. Varsayılan durumda, exposure 0 (sıfır) değerine ayarlanmıştır. Değer arttırıldığı zaman sahne parlaklaşır, azaltıldığı zaman sahne kararır.
- **Antialiasing:** Grafik kartının kapasitesine bağlı olarak, bu ayar için herhangi bir seçenek olmayabilir. Antialiasing kontrolü, kenarları pürüzsüz hale getirerek simülasyon sahnesinin kalitesini etkiler. Antialiasing olmazsa kenarlar pürüzlü görünür.
- **Rotation Movement Scale:** Bu kontrol, simülasyon sahnesi etrafında mouse hareketlerinin hassasiyetini ayarlar. Varsayılan durumda, 1 (bir) değerine ayarlanmıştır. Değer arttırıldığı zaman mouse hareketi hızlanır ve değer azaltıldığı zaman mouse hareketi yavaşlar.

- **Translation Movement Scale:** Rotation movement scale ayarına benzer, bu ayar simülasyon sahnesi etrafında klavye veya joystick hareketlerinin hassasiyetini ayarlar. Ayar yükseltildiği zaman denetim hızlı hareket eder ve azaltıldığında yavaş hareket eder.
- **Quality Level:** Sahnenin kalitesinin düşük, orta veya yüksek olarak ayarlanmasını sağlar. Varsayılan durumda, VSE, kullanılan grafik kartı için en uygun değeri seçer. Değer arttırıldığı zaman muhtemelen simülasyon hızı etkilenecektir.



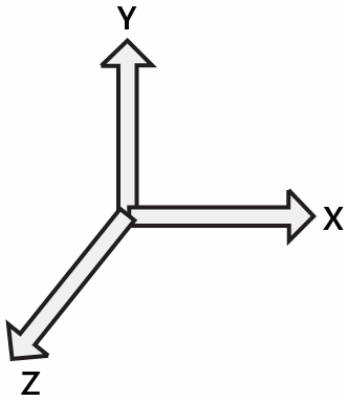
Şekil 7.6 Grafik ayarları diyalog kutusu.

Aynı zamanda, Physics menüsü aracılığıyla physics ayarları değiştirilebilir. Physics ayarları yerçekimi gibi prensipleri physics motoruna uygulayarak kontrol etmeyi sağlar. Yerçekimi ayarı, varsayılan değeri 9.81'dir ve simülasyon

sahnesindeki yerçekimini kontrol eder. Tipik olarak, bu ayar varsayılan değerde tutulmak istenir, fakat düzenleme modunda entity'lerde değişiklikler yaparken yerçekimi değeri sıfır yapılarak kapatmak istenebilir.

7.6 Koordinatlarla Çalışma

Üç boyutlu uzayda fiziksel bir objeyi tanımlamak ve objenin pozisyonunu tanımlamak için koordinatlarla çalışılmak zorunludur. Bu koordinatlar X,Y ve Z eksen hatları boyunca noktaları içerirler. MSRS simülasyon motoru, sağ-el koordinat sistemini kullanır. Bu pozitif Z-ekseninin yönünü belirtir. Şekil 7.7 X,Y ve Z eksen hatlarının yönlerini göstermektedir. Zıt yöndeki hatlar negatif eksenleri göstermektedir.



Şekil 7.7 VSE sahnesinde X,Y ve Z eksen hatlarının yönleri

Entity'leri oluştururken veya düzenlerken, entity'nin pozisyonu gibi şeyler entity'nin koordinatları değiştirilerek yapılır. Y-eksen koordinatını daha yüksek pozitif değere değiştirerek, entity simülasyon sahnesinde yukarı hareket edecektir. Eğer değer düşürülürse aşağı hareket edecektir.

7.7 Entity Nedir

Entity, Türkçe karşılığı varlık, nesne gibi anlamlara gelmektedir ve simülasyonda kullanılan fiziksel objeyi temsil etmektedir. Bu sadece robotları içermez, ortamdaki masalar, sandalyeler gibi engelleri de içerir. Aynı zamanda yeryüzü ve gökyüzü gibi nesneleri de içerir. Simülasyonlar ya kapalı ya da açık ortamlarda yer alabilir.

Entity hiyerarşik olabilir ve üst ve alt ilişkiler gerektirebilir. Çoğu robotik donanımda, motor objesi üst entity'dir. Sensörler robotun alt entity'si olabilir. Örneğin, Pioneer 3DX simüle edilmiş robot bir web kamera, bir lazer rangefinder ve tampon alt entity'lerini içermektedir.

Birçok temel simülasyon bir düzine veya daha fazla entity içerebilir. Karmaşık simülasyonlar yüzlerce hatta binlerce entity içerebilir. Tüm simülasyonlar bir yeryüzü entity'sine gereksinim duyarlar. Aynı zamanda gökyüzü objesine ihtiyaç duyarlar, fakat gökyüzünde görünen güneş gibi diğer elementlere ihtiyaç duymazlar. Eğer simülasyon kapalı ortamda yapılıyorsa, genelde bir kutunun içerisinde yapılacaktır ve ışık gibi demirbaşların genelde dahil edilmesine gerek yoktur.

7.8 Yeni Bir Entity Oluşturmak

VSE'de yeni bir entity oluşturmak için, simülasyon ortamında düzenleme moduna geçilmelidir. Düzenleme modda iken, sol-üst pencere gözünde simülasyon sahnesinde bulunan tüm entity'leri gösterecektir. Bir öğeyi seçildiği zaman, bu entity'nin özellikleri sol-alt pencere gözünde görünecektir. Çizelge 7.1'de, bir entity'nin temel özellikleri listelenmektedir.

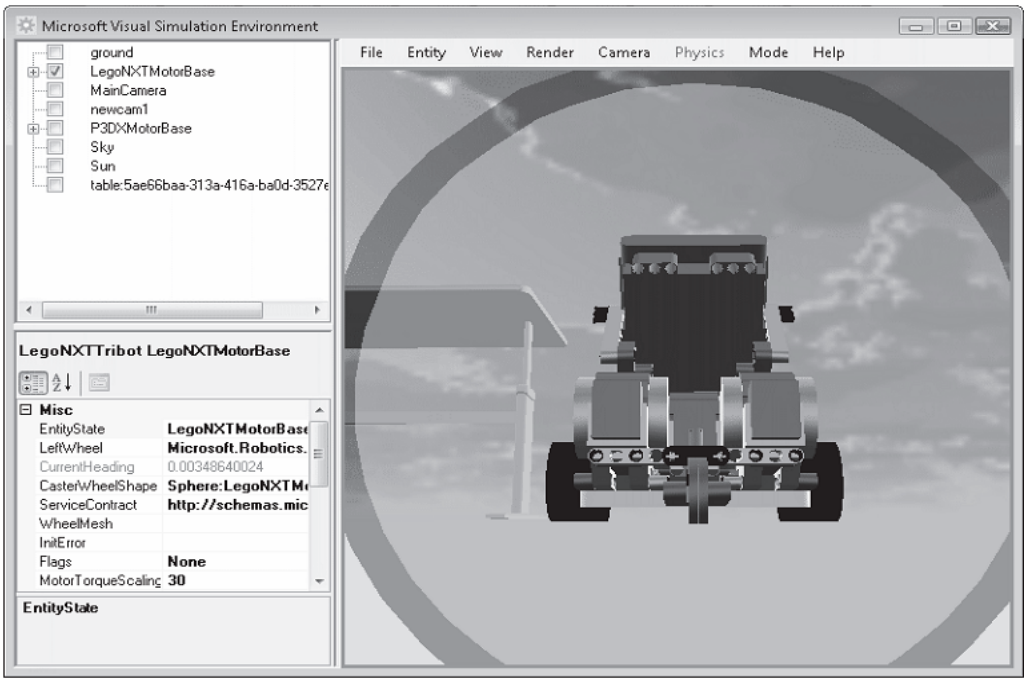
Entity'nin tipine bağlı olarak, ek özellikler olabilir. Örneğin, Pioneer 3DX robotunu temsil eden bir entity "MotorTorqueScaling" isminde bir özellik içerebilir.

Çizelge 7.1 Temel entity özellikleri

ÖZELLİK İSMİ	AÇIKLAMA
EntityState	Entity'nin görüntüsünün ve davranışlarının kontrol edilmesini sağlar. EntityState ile ilgili özelliklerin listesi Çizelge 7.2'de verilmiştir.
ServiceContract	Servis için contract belirtmede kullanılır. Contract, HTML dosyası ile ortak çalışır ve tüm yolun contract içinde belirtilmesi gerekmektedir. Entity bir servis contractla ortak olduğu zaman, sahne kaydedildiği zaman üretilen manifest dosyası içine dâhil edilecektir.
InitError	Entity'nin başlatılması süresince oluşan hataların tanımını içerir.
Flags	Entity'nin nasıl render edileceğini kontrol eder ve aşağıdaki opsiyonlardan birine ayarlanabilir.
	None: Render'lama özelliği ayarlanmamıştır, varsayılan ayardır.
	UsesAlphaBlending: Şeffaflığın ayarlanmasını sağlar, böylece daha gerçekçi sahneler elde edilir. Grafik kartının yeteneğine bağlı olarak, AlphaBlending kullanılarak performans azaltılabilir.
	DisableRendering: Render'lamayı kapatır.
	InitializedWithState: Entity'nin, XML'in simülasyon sahnesiyle birlikte yapıldığını gösterir.
	DoCompletePhysicsShapeUpdate: Entity'nin parçası olan tüm fiziki şekilleri güncellemeye zorlar.
	Ground: Yeryüzü entity'sini ifade etmede kullanılır.
ParentJoint	Entity'i üst entity'ye bağlamada kullanılan birleşme yerini belirtir. Bu entity'lerin, diğer entity'lerin birleşiminden oluşması durumunda kullanılır. Örneğin, kollu veya ayaklı robotlar.
Position	Entity'nin simülasyon sahnesindeki X,Y, ve Z koordinatlarını belirtmek için kullanılır. Koordinatlar metre olarak belirtilirler.
Rotation	Euler açılarına dayanır, entity'nin yönlendirilmesini kontrol etmede kullanılır.
Meshes	Entity ile ilgili bir veya daha fazla mesh'i belirtir. Mesh'ler, VisualEntityMesh koleksiyon editörü ile eklenir. Materyaller fiziksel objeleri oluşturmada kullanılırlar ve entity'ler bir veya daha fazla materyalle ilgili olabilirler. Aynı zamanda, entity materyal içermeyebilir, fakat bunun anlamı entity'nin görünmez olduğudur.
MeshRotation	Mesh nesnesinin yönlendirilmesinin kontrol edilmesinde kullanılır. Tipik olarak, başlangıçta sıfır değerine ayarlanır.

MeshTranslation	Entity'nin orijinine bağlı olarak, mesh nesnesinin çevrilmesinin kontrol edilmesini sağlar. Tipik olarak, başlangıç değeri sıfırdır.
MeshScale	Mesh'in ölçeğinin kontrol edilmesini sağlar. Tipik olarak, koordinatları başlangıçta 1 değerine ayarlanmalıdır. Daha yüksek değerlerde, nesne daha büyük görünür, negatif değerlerde nesne daha küçük görünür.
Shapes	Bu özelliğin sağlanabilirliği entity tipine bağlıdır. Bazı entity'lerde bu özellik yoktur. Entity silindir, kapsül veya kutu gibi tek şekle dayanıyorsa, hangi şeklin olduğu belirtilebilinir ve ayrı bir diyalog kutusunda şekille ilgili ek özellikler ayarlanabilir.

Düzenleme modunda iken, hangi entity'nin seçili olduğunu görmek için Ctrl tuşunu basılı tutulur. Seçili entity'nin etrafında Şekil 7.8'de gösterildiği gibi entity'yi içine alan bir çember görülecektir. Ok tuşlarını kullanarak, kamera sayesinde seçili entity'nin etrafında gezinerek obje incelenebilir. Örneğin, Şekil 7.8'de LEGO NXT'nin arkadan görüntüsü görülmektedir. Entity'nin pozisyonunu değiştirmek için Ctrl tuşu ve mouse kullanılabilir. Bu düzenleme modunda, nesne seçimi yapılarak ve aynı zamanda konum ve yönlendirme özellikleri seçilerek de yapılabilir. Ctrl tuşunu basılı tutarken, entity'yi taşımak ve yönlendirmek için mouse kullanılır.



Şekil 7.8 LEGO NXT'nin arkadan görüntüsü.

Yeni bir entity oluşturmak için, Entity menüsüne tıklanır ve Yeni (New) seçilir. Şekil 7.9'da görülen Yeni Entity Diyalog Kutusu ortaya çıkacaktır. Entity mevcut simülasyona ya da diğer bir simülasyona eklenecektir. Çizelge 7.2'de görülen entity tiplerinden birisi seçili olmak zorundadır. Entity tipi, entity için bir şablon görevi üstlenmektedir, böylece yeni entity için başlangıç özellik değerleri atanmış olur.



Şekil 7.9 Yeni entity diyalog kutusu.

Çizelge 7.2 MSRS'de bulunan entity türleri

ENTİTY TİPİ	AÇIKLAMA
ArmLinkEntity	Eklemli robotik kolun bir bağlantısını modellemede kullanılır. Örneğin altı derece serbestliği olan KUKA LBR3. KukaLBR3Entity tipine karşı olarak, bu tip robotik kolun sadece bir parçasını temsil eder ve kolun tamamı altı bağlantıdan oluşmaktadır.
BumperArrayEntity	Tampon sensörlerinin bir dizisini oluşturmada kullanılır. Tampon sensörlerini kullanan Pioneer 3DX ve iRobot Create robotları gibi robotlar için bu tipten oluşturulan entity alt entity olarak rol alacaktır.

CameraEntity	Doksan derece görüş açısına sahip olan kamera entity'sini oluşturmada kullanılır. Pioneer 3DX veya iRobot Create gibi web kamera taşıyan robotlar için bu entity alt entity olarak rol alacaktır.
HeightFieldEntity	Yeryüzü nesnesini temsil etmede kullanılır ve düz bir yeryüzünü temsil etmede de kullanılabilir.
iRobotCreate	iRobot Create robotunu modellemede kullanılır.
KukaLBR3Entity	KUKA LBR3 robotik kolunu modellemede kullanılır.
LaserRangeFinderEntity	Lazer rangefinder entity'sini oluşturmada kullanılır. Pioneer 3DX gibi lazer rangefinder kullanan robotlar için bu entity alt entity olarak davranır.
LegoNXTTribot	LEGO NXT Tribot'u modellemede kullanılır.
LightSourceEntity	Gökyüzü ve ışık gibi ışık-tabanlı entity'leri oluşturmada kullanılır. Işık kaynakları ya tek yönden ya da tüm yönlerden olabilir.
MultiShapeEntity	Çoklu entity'ler içeren entity'ler tanımlamada kullanılır. Örneğin, masa entity'si MultiShapeEntity tipi kullanılarak yapılır. MultiShapeEntity'deki herbir şekil diğer entity'lerle ilgili sabit bir konuma sahiptir.
Pioneer3DX	MobileRobots Pioneer 3DX robotunu modellemede kullanılır.
SimplifiedConvexMesh-EnvironmentEntity	Entity'nin etrafını belirten konveks mesh'i üretmede kullanılır. Mesh nesneden dolayı entity'ler aynı görsellikte olmasına rağmen, entity fizik model olarak render'landığı zaman farklı görünür.
SingleShapeEntity	Küre, kutu ve kapsül gibi tek şekilli entity'leri oluşturmada kullanılır.
SkyDomeEntity	İki boyutlu kubbe görüntüsünü kullanarak gökyüzü oluşturmada kullanılır. İki boyutlu görüntü kullanmak daha az kaynak yoğunluğu ve daha iyi seviyede detay istenmiyorsa kullanılmalıdır. Diğer durumlarda SkyEntity kullanılmalıdır.
SkyEntity	Altı kenarlı görüntü dosyası olan küp haritasını kullanarak gökyüzü oluşturur. İki küp haritası kullanılır: birisi gökyüzünü çizmekte diğeri de sahneyi aydınlatmak için kullanılır. Çünkü, SkyEntity iki boyutlu görüntüye zıt olarak küp haritalarına ihtiyaç duyar ve bu render için daha fazla işlemci gücünü kullanır. Eğer mümkünse, gökyüzü entity'sini render'lamak için SkyDomeEntity türü kullanılmalıdır.
TerrainEntity	Bitmap görüntü kullanan karmaşık yeryüzü entity'sini oluşturmada kullanılır
TerrainEntityLOD	Karmaşık yeryüzü entity'sini oluşturmada kullanılır. Bu entity türü daha iyi seviyede detaylı bir entity türüdür.

TriangleMesh-EnvironmentEntity	SimplifiedConvexMeshEnvironmentEntity ye benzer, fakat şekil olarak konkav olacaktır. Bunun anlamı, physics motorun bu entity ile çalışırken daha fazla çalışması gerektiği, bu yüzden bu entity kesinlikle gerekli olduğunda kullanılması gerekmektedir.
WheelEntity	Bir hareket sağlayıcı birimi temsil eder ve robotik motor tabanı için alt entity olacaktır.

Çizelge 7.3, EntityState özelliğinin yanındaki elipsler seçildiği zaman sağlanan temel özelliklerin listesidir. Entity türüne bağlı olarak, özellikler biraz farklı görülebilir. Bu özellikler, hem entity'nin görünümünü hem de physics motoruyla ilişkinin nasıl olacağını kontrol ederler.

Çizelge 7.3'te listelenen özellikleri ayarlarken, MSRS'de uzaklık için metrenin ve ağırlık için kilogramın kullanıldığı dikkate alınmalıdır. 1 metre 1,093 yard veya 39,37 inç'tir. 1 yard 0,0914 metre veya 36 inç'tir. Aynı zamanda, 1 kilogram 1000 gram, 453,59 gram 16 ons veya 1 pound'tur.

Çizelge 7.3 Temel entity durum özellikleri

ÖZELLİK ADI	AÇIKLAMA
DefaultTexture	Tüm şekillere uygulanan texture dosyasına başvurur. Texture file DirectX texture dosyası olmalıdır, dosya uzantısı .dds'tir. Texture dosyasının entity state'e uygulanması seçenekseldir ve sadece mesh nesnesi belirtilmezse gereklidir.
Effect	Entity'yi render etmede kullanılan grafik etki dosyası (FX)'na başvurur. Grafik etki dosyası, DirectX'e hangi gölgelendiricilerin yükleneceğini anlatan betiktir. MSRS, .fx dosya uzantılı birkaç etki dosyasıyla birlikte gelir. Bunlar gökyüzü ve arazi gibi entity'ler için çeşitli render'lama etkilerini yapmada kullanılırlar. FX dosyaları, MSRS tarafından desteklenmektedir, \store\media alt dizini yoluyla bunlara ulaşılabilir.

Mesh	Entity'yi render'lamada kullanılan opsiyonel mesh dosyasına başvurur. Mesh nesne dosyası, .obj dosya uzantılıdır, 3D grafiksel düzenleme aracıyla oluşturulur. Mesh, entity'nin gerçekçi görünmesini sağlar ve mesh olmaksızın entity sadece temel bir şekil gibi görünür. Aynı zamanda, entity'nin daha hızlı render edilmesine yarayan binary dosyası belirtilebilir. Binary dosyası .bos uzantılıdır. MSRS, mesh dosyasını binary dosyasına dönüştürmede kullanılan Obj2bos.exe isimli komut-hattı dönüştürücü aracını da sağlamaktadır. MSRS'deki mesh dosyalarına \store\media dizini yoluyla erişilebilir.
Flags	Entity'nin nasıl render edileceğini ve nasıl davranacağını belirten flag'ları belirtir. Flag'lar aşağıdaki seçeneklerin birisine ayarlanabilir.
	Kinematic: Objenin sonsuz kütlede olduğunu gösterir, bu nedenle, obje hareketsiz kalacaktır.
	IgnoreGravity: Tek entity için yerçekiminin etkisini yok etmenizi sağlar.
	DisableRotationX: Entity'nin X-ekseni boyunca döndürülmesini engeller.
	DisableRotationY: Entity'nin Y-ekseni boyunca döndürülmesini engeller.
	DisableRotationZ: Entity'nin Z-ekseni boyunca döndürülmesini engeller.
Name	Entity ile birlikte kullanılan tek bir isimdir.
AngularVelocity	Entity döndürülürken hız ve yön belirtmede kullanılır. Bu X,Y, ve Z koordinatları ile belirtilir.
Velocity	Entity'nin lineer hızı ile ilgili X,Y ve Z koordinatlarını ayarlamada kullanılır.
AngularDamping	Entity'nin kütlesiyle ilgili olarak uzayda ne kadar dönmesiyle ilgili bir tamsayı katsayı ayarlamada kullanılır. Varsayılan değer sıfırdır.
Density	Nesnenin yoğunluğunu belirtmede kullanılır. Bu ayar, fizik motoru tarafından ayarlanır, bu yüzden bu değeri ayarlamanıza gerek yoktur. Varsayılan değer sıfırdır.
InertiaTensor	Kütlenin X,Y ve Z eksenleri boyunca nasıl dağıldığını belirtmek için kullanılır. Bu ayar, fizik motoru tarafından hesaplanmaktadır, bu nedenle varsayılan değerin dışında herhangi bir değişikliğe gitmenize gerek yoktur.
LinearDamping	Entity'nin sürtünme katsayısını belirtmede kullanılır. Varsayılan değer sıfırdır.
Mass	Entity'nin kütlesini kilogram olarak belirtmede kullanılır. Mass özelliğine atanan değer önemlidir.

Entity'nin durumuyla ilgili değerler simülasyon çalışırken değiştirilebilir. Örneğin, robotun pozisyonu, pozisyon özelliği kullanılarak X,Y ve Z koordinat

değerleriyle belirlenir. Bu değerlerin ayarlanmasıyla robotun simülasyon sahnesindeki pozisyonunun değiştiği görülecektir.

7.9 Fizik (Physics) Modun Aktif ve Pasif Yapılması

Düzenleme modu seçildiğinde, VSE otomatik olarak fizik motorunu kapatacaktır. Physics menüsünü kullanılarak fizik motorunun tekrar aktif hale getirilmesi tavsiye edilmez. Fizik motoru aktif edildiğinde, bazı şeyler yapılamaz. Örneğin aynı zamanda iki objenin aynı noktada bulunamaması gibi durumlar gerçekleşmeyecektir. Bu tür bir çelişki, entity'yi kopyalayıp yapıştırdığınızda oluşur.

Düzenleme modunda iken, bir entity'nin kopyası oluşturulduğunda, yeni entity üst-sol entity listesinde görünecektir, fakat simülasyonda yeni entity hemen görünmeyecektir. Çünkü, kopyalanmış entity orijinal entity ile aynı konumdadır. Bu yüzden, sadece birisi görünürdür. Eğer physics aktif edilirse, iki entity hızlıca birbirinden ayrılacaktır. Eğer physics mod aktifken aynı kopyalama yapıştırma işlemi yapılacak olursa, yeni entity orijinal entity'den hızlıca uzaklaşacaktır.

7.10 Simülatör Durum Dosyasının Düzenlenmesi

VSE sayesinde simülasyon sahnelerini File menüsünden Save Scene As seçilerek kaydedilebilir. XML tabanlı iki dosya oluşacaktır: biri manifest dosyası ve diğeri simülatörün durumu hakkındaki ve simülasyon sahnesindeki entity'leri içeren dosyadır.

Simülatör durum dosyasındaki bilgiler birçok amaç için faydalı olabilir. Simülasyon hakkında bilgi sahibi olurken ve her bir entity için birçok özellik sağlanırken, simülasyon durum dosyasını listelemek ve her bir özellikle ilgili hangi değerlerin üretildiğini görmek faydalı olabilir. Örneğin, aşağıdaki XML, grikutu (greybox) isimli bir tek şekil entity ile ilgili durumun sadece bir parçasıdır.

```
<BoxShape>
  <phys:BoxState>
    <phys:Name>greybox</phys:Name>
    <phys:ShapeId>Box</phys:ShapeId>
    <phys:Dimensions>
      <physm:X>1</physm:X>
      <physm:Y>1.5</physm:Y>
      <physm:Z>0.5</physm:Z>
    </phys:Dimensions>
    <phys:Radius>0</phys:Radius>
    <phys:Material>
      <phys:Name>gbox</phys:Name>
      <phys:Restitution>0.5</phys:Restitution>
      <phys:DynamicFriction>0.4</phys:DynamicFriction>
      <phys:StaticFriction>0.5</phys:StaticFriction>
      <phys:MaterialIndex>3</phys:MaterialIndex>
    </phys:Material>
    <phys:MassDensity>
      <phys:Mass>10</phys:Mass>
      <phys:InertiaTensor>
        <physm:X>0</physm:X>
        <physm:Y>0</physm:Y>
        <physm:Z>0</physm:Z>
      </phys:InertiaTensor>
      <phys:CenterOfMass>
        <physm:Position>
          <physm:X>0</physm:X>
          <physm:Y>0</physm:Y>
          <physm:Z>0</physm:Z>
        </physm:Position>
        <physm:Orientation>
          <physm:X>0</physm:X>
          <physm:Y>0</physm:Y>
          <physm:Z>0</physm:Z>
          <physm:W>0</physm:W>
        </physm:Orientation>
      </phys:CenterOfMass>
      <phys:Density>0</phys:Density>
      <phys:LinearDamping>0</phys:LinearDamping>
      <phys:AngularDamping>0</phys:AngularDamping>
    </phys:MassDensity>
    <phys:LocalPose>
      <physm:Position>
        <physm:X>0</physm:X>
        <physm:Y>0</physm:Y>
```

```
          <physm:Z>0</physm:Z>
        </physm:Position>
        <physm:Orientation>
          <physm:X>0</physm:X>
          <physm:Y>0</physm:Y>
          <physm:Z>0</physm:Z>
          <physm:W>1</physm:W>
        </physm:Orientation>
      </phys:LocalPose>
      <phys:TextureFileName />
      <phys:DiffuseColor>
        <physm:X>0.5</physm:X>
        <physm:Y>0.5</physm:Y>
        <physm:Z>0.5</physm:Z>
        <physm:W>1</physm:W>
      </phys:DiffuseColor>

<phys:EnableContactNotifications>false</phys:EnableContactNo
tifications>
  </phys:BoxState>
</BoxShape>
```

Simülasyonda kullanılan entity'ler hakkında bilgi toplamanın yanında, simülasyon durum dosyası özelliklerin hızlı bir şekilde düzenlenmesini veya yeni entity'lerin oluşturulmasında oldukça faydalıdır. Simülasyon durum dosyasında düzenlemeler yaptıktan sonra, File ve Open Scene menü araçlarını seçerek değişiklikler VSE'ye tekrar yüklenebilir.

7.11 Servisten Simülasyon Oluşturma

VSE kullanarak simülasyon sahnesi oluşturmaya ek olarak, Visual Studio kullanarak programla simülasyon sahnesi oluşturulabilir. Bu bölümde, programla simülasyon oluşturmak için gereken temel bilgiler verilmiştir.

Bunu yapmak için, MSRS ile birlikte gelen şablonu kullanarak yeni bir DSS servisi oluşturulmuştur. Proje oluşturulduktan sonra, aşağıda verilen derleyicilere referanslar eklemeniz gerekmektedir.

- **PhysicsEngine:** AGEIA yazılımında bulunan fizik motoruna erişimde kullanılan özellik ve metotları içerir.
- **RoboticsCommon:** Robotun fiziki özelliklerini tanımlamada kullanılan tür tanımlarını içeren PhysicalModel isim alanını içerir. Örneğin, analog sensörler, web kameraları ve iletişim sensörleri gibi sensörleri temsil eden tanımları PhysicalModel isim alanında bulunabilir. Aynı zamanda, eklemli kol servisi gibi soyut servislerin tür tanımları da burada bulunabilir.
- **SimulationCommon:** Hem simülasyonla hem de fizik motoruyla çalışırken kullanılan tür tanımlarını içerir.
- **SimulationEngine:** Simülasyon motoruna erişmede ve fonksiyonları yerine getirmede kullanılır.
- **SimulationEngine.proxy:** Simülasyon motoruna partner olarak kullanılan Proxy dir.

Derleyici referanslarını ekledikten sonra, aynı zamanda isim alanı tanımlarını da ilişkilendirilmiş referanslar için eklemeniz gerekmektedir. Örneğin, aşağıdaki isim alanı tanımlarını simülasyonun yürütme sınıf dosyasının en üstüne eklenmesi gerekmektedir.

```
using Microsoft.Robotics.Simulation.Physics;
using Microsoft.Robotics.PhysicalModel;
using Microsoft.Robotics.Simulation;
using Microsoft.Robotics.Simulation.Engine;
using engineproxy =
Microsoft.Robotics.Simulation.Engine.Proxy;
```

MSRS, Çizelge 7.4'te görülen diğer servisleri de içermektedir. Örneğin, tamponlu ve diferansiyel sürüş sistemli bir robot simülasyonu SimulatedBumper ve SimulatedDifferentialDrive servislerini içermelidir.

Çizelge 7.4 Özel simülasyon fonksiyonu için kullanılan ilave MSRS servisleri

SERVİS İSMİ	AÇIKLAMA
SimulatedBumper	Simüle edilmiş bir robot için tampon sensörleri dizisine erişmekte kullanılır. Kaynak kodlara \samples\simulation \sensors\bumper folder yoluyla erişilebilir. MSRS kontrol panelinde, bu servis Simulated Generic Contact Sensors servisi olarak listelenmektedir.
SimulatedDifferential Drive	Diferansiyel sürüş sistemli simüle edilmiş robot'u sürmek için kullanılır. Bu servis, diferansiyel sürüş servisine referans kullanır. Kaynak kodlara \samples\simulation \articulatedarms\LBR3 yoluyla erişilebilir.
SimulatedLBR3Arm	KUKA LBR3 eklemli kolunun birleşme noktalarının kontrolü için kullanılır. Kaynak kodlara \samples \simulation\articulatedarms\LBR3 yoluyla erişilebilir.
SimulatedLRF	Pioneer robot üzerinde bulunan sick lazer rangefinder aracını simüle etmek için kullanılır. Bu araç 180 derecelik infrared ışık demeti yayar. Bu sayede nesneler daha hassas olarak sezilir. Bu servis SickLRF servisiyle ilişkilidir. Kaynak kodlara \samples\simulation\sensors \laserrangefinder yoluyla erişilebilir.
SimulatedWebCam	Simüle edilmiş robot üzerinde bulunan web kameraya erişmek için kullanılır. Kaynak kodlara \samples\simulaton \simulatedwebcam yoluyla erişilebilir.

Çizelge 7.4'teki listelenen özelliklere referansları ayarlarken, ilgili Proxy dosyaları için referanslara ihtiyaç duyulacaktır. Örneğin, Simulated Drive servisi için kullanılan Proxy referansın ismi SimulatedDifferentialDrive.2006.M06.Proxy'dir.

Çünkü simülasyon motoru bir servistir, buna bir partner servis eklenmesi gerekmektedir. Bunu aşağıdaki kod kullanılarak yapılabilir.

```
[Partner("Engine",
   Contract = engineproxy.Contract.Identifier,
     CreationPolicy =
PartnerCreationPolicy.UseExistingOrCreate)]

private engineproxy.SimulationEnginePort _engineServicePort
= new engineproxy.SimulationEnginePort();
```

Bu durumda, simülasyon servisi UseExistingOrCreate kullanılarak oluşturulur. Bunun anlamı, eğer simülasyon motoru hala çalışıyorsa, yeni bir örnek denemektense bu kullanılmalıdır.

7.12 Yeni Entity'ler Oluşturmak

Entity oluştururken izlenecek adımlar entity türüne göre değişmektedir. MSRS türünü robot entity'si gibi birkaç entity sağlamaktadır. Aynı zamanda, MSRS gökyüzü ve yeryüzü gibi ortam entity'leri gibi entity türlerini de sağlamaktadır. Hem basit hem de kompleks nesneler SimpleShape ve MultipleShape entity'leri kullanılarak yapılabilir.

7.12.1 Robot Entityleri Oluşturmak

Robot entity'leri karışıktır ve genelde birkaç alt entity içerirler. Alt entity'ler simüle edilmiş robotta kullanılan sensörler olabilir. Örneğin, Pioneer 3DX robotun simüle edilmiş versiyonu lazer rangefinder cihazını ve tampon sensörlerini içermektedir. Simüle edilmiş Pioneer'a entity oluşturmak için, ilk başta üst entity oluşturmak gerekmektedir, bu entity'nin temeli olacaktır. Bu simülasyon motorunda bulunan Pioneer3DX sınıfı ile yapılabilir. Bu sınıf DifferentialDriveEntity'den kalıt alır, ayırt edici sürüş sistemi ile ilgili farklı fiziksel özellikleri tanımlamak için kullanılır. Aşağıdaki kod, Pioneer robotu temsil eden yeni bir entity oluşturmada kullanılabilir.

```
Pioneer3DX robotBaseEntity = new Pioneer3DX(new
Vector(0,1,0));
```

Bir önceki kodda, robotun simülasyon sahnesindeki konumunu tanımlamak için Vector3 yapısı kullanıldı. MSRS, her bir koordinat noktası ile ilgili değerleri depolamakta kullanılan vektör yapılarını sağlamaktadır. Vector2 yapısında, X ve Y koordinat değerleri depolanmaktadır. Vector3, X,Y ve Z koordinat değerlerini depolar. Vector4 yapısı, X,Y,Z ve W değerlerini depolamaktadır. Pioneer robot entity'sini oluştururken, robot için bir isim belirtilmeli ve Vector4 yapısı kullanılarak robotun rengi belirtilmelidir.

Motor taban entity'sini oluşturulduktan sonra, Pioneer robot servisini temsil eden servis başlatılabilir. Bunu yapmak için, CreateService metodu entity partner

servisi oluşturmak için kullanılabilir. Bu çalışma sırasında simülasyon servisine hangi diğer servislerin bağlanacağını anlatır. Aşağıdaki kod bunu yapmak için kullanılabilir.

```
CreateService(
   Drive.Contract.Identifier,
 Microsoft.Robotics.Simulation.Partners.CreateEntityPartner(
  "http://localhost/" + robotBaseEntity.State.Name)
);
```

Bu noktada, alt entity'ler, lazer rangefinder ve tampon sensörleri gibi, oluşturulabilirler. Bunu yapmak için, LaserRangeFinderEntity nesnesini kullanarak yeni bir entity oluşturulur ve CreateService metodunu da lazer rangefinder servisini buna bağlama işlemi için kullanılır.

7.12.2 Yeni Entity Türleri Oluşturmak

MSRS, Pioneer 3DX, iRobot Create, KUKA LBR3, ve LEGO NXT robotlarını destekleyen entity türlerini ve tanımlama sınıflarını sağlamaktadır. Yeni bir sensörü veya robotu simüle etmek için, ilk önce yeni bir entity türü oluşturulmalıdır. Bu entity sınıf dosyası güncellenerek yapılabilir. Bu dosyaya \samples\simulation \entities yoluyla erişilebilir.

8 GÖMÜLÜ BİLGİSAYAR TABANLI MOBİL ROBOT SİSTEMİ

8.1 Donanım

Bu tez çalışmasında Lego firmasının ürettiği gömülü bilgisayar sistemine sahip Lego NXT Mindstorms adlı robotu kullanılmıştır. Bu robotun tercih edilmesini sağlayan özelliklerinden biriside robotun parçalanabilir ve istenilen herhangi bir robot olarak tekrar yapılabilir şekilde esnek bir yapıya sahip olmasıdır.

8.1.1 Lego NXT Mindstorms robotu

Lego firmasının üretmiş olduğu Lego NXT Mindstorms robotu Lego RCX modelinin bir üst ve son versiyonu olan gömülü bilgisayar sistemine sahip gelişmiş bir robottur. Bu robotu, gömülü bilgisayar sistemine sahip diğer robotlardan ayıran en önemli özelliği her bir parçasının sökülüp takılabiliyor olmasıdır. Bu sayede hayal edilebilen birçok robot türüne dönüştürülebilmektedir. Bu özelliği Şekil 8.1'de insansı robot, iş makinesi ve akrep robot olarak üç şekilde dizayn edilmiş NXT robot türleri gösterilmiştir.

Lego NXT Mindstorms robotunu kontrol eden gömülü bilgisayar sistemine sahip bir kontrol donanımı vardır. Çevresini algılayabileceği 4 adet sensörü bulunmaktadır. Bu sensörler; mesafe ve engelleri algılayabilen ultrasonik alıcı verici, renkleri ayırt edebilen ışık sensörü, ses komutlarını algılayan ses algılama sensörü ve dokunma sensörüdür. Tasarlanan robota hareket özelliği kazandıran 3 adet enkoderli servo motorları vardır. Bu motorlar 1 derece hassasiyetle adım adım kontrollü bir şekilde döndürülebilmektedir. Motorların enkoder özelliği sayesinde motorun o andaki konum ve açı bilgileri alınabilmektedir.

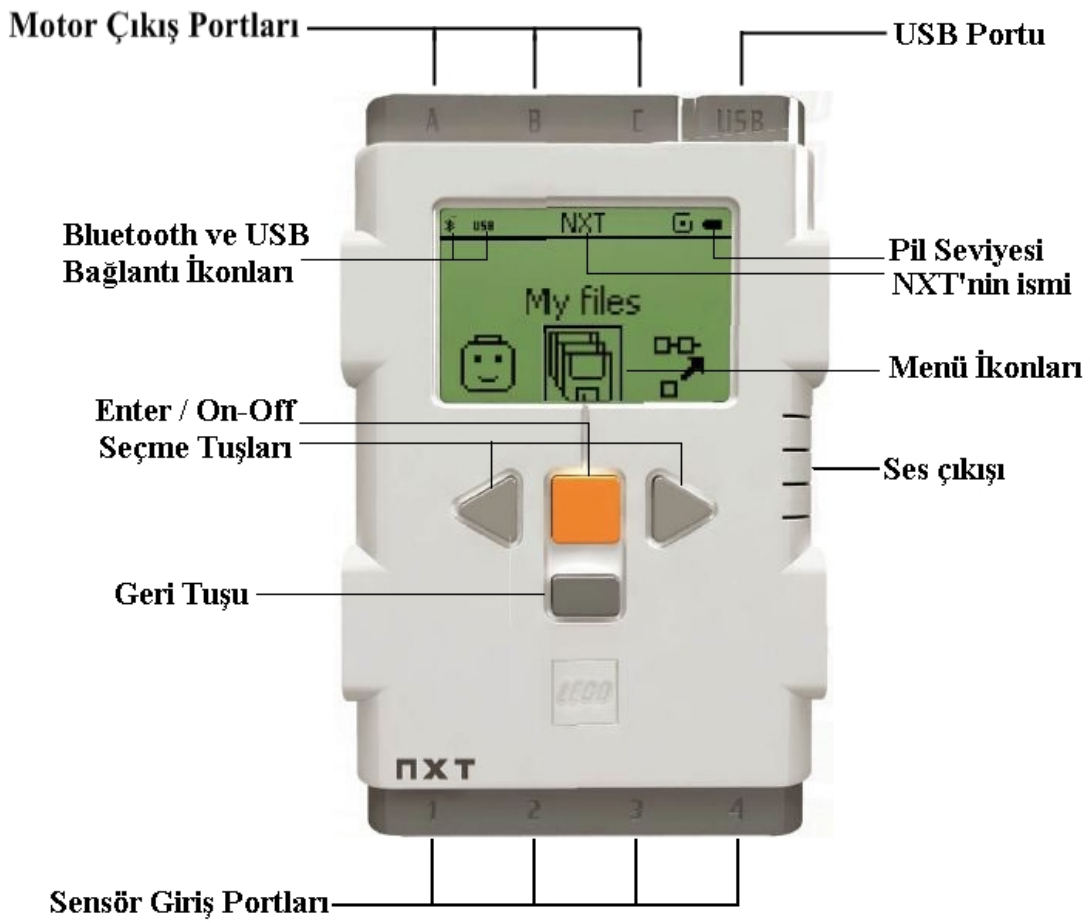


Şekil 8.1 Farklı şekillerde dizayn edilmiş NXT robotu

8.1.1.1 NXT Brick gömülü bilgisayar sistemi

Lego NXT Mindstorms robotunun beyni olarak kullanılan kontrol modülü gömülü bilgisayar sistemli NXT'dir. Bilgisayar yazılımların kullanılan ismi ise NXT Brick olarak geçmektedir. Bu zeki modül Mindstorms robotunun sensör ve motorlarını kontrol etmek için kullanılmaktadır.

Şekil 8.2'de NXT gömülü sistem modülü görülmektedir. NXT modülü üzerinde motor çıkış portları, sensörler için her biri bağımsız olarak ayarlanabilen sensör giriş portları, kontrol tuşları, pil seviyesi gibi birçok olayı takip edebildiğimiz grafik ekranı, kablosuz bağlantı için bluetooth modülü, kablolu bağlantı için USB portu ve ses çıkışı sağlayan hoparlörü mevcuttur.



Şekil 8.2 NXT Brick gömülü sistemi

Motor portları, Port A, B ve C şeklinde olmak üzere üç adet çıkışa sahiptir. Bu portlarla Mindstorms robotunun servo motorları tam olarak kontrol edilir. Ayrıca bu portlarda servo motorlardan gelen enkoder bilgisi de alınabilmekte ve işlenebilmektedir.

Sensör girişleri, Port 1,2,3 ve 4 şeklinde olmak üzere dört adet girişe sahiptir. Bu portlara ultrasonik sensör, ışık sensörü, ses sensörü ve dokunma sensörü olmak üzere dört çeşit sensör bağlanabilmektedir. Her bir sensör portu girişi istenilen herhangi bir sensöre göre yazılımdan ayarlanabilmektedir.

USB portu, bir USB kablo ile bilgisayardan NXT'ye veya NXT'den bilgisayara download ve upload işlemi için kullanılmaktadır. Bu işlemler NXT gömülü bilgisayar sistemi içerisinde bulunan bluetooth modülü sayesinde de yapılabilmektedir.

Hoparlörü sayesinde programda yazılan veya söylenmesi istenen sesler seslendirilebilmektedir. Ayrıca bağlantıların gerçekleşip gerçekleşmedikleri de hoparlörden gelen beep sesi ile kontrol edilebilmektedir.

NXT üzerindeki butonlarda hem NXT içerisindeki menü ve programın kontrolü için hem de robot için yazılan programda aktif olarak kullanılabilir. Bu butonlar:

- **Turuncu Buton:** Üç adet görevi vardır. Birincisi NXT'yi açmak için kullanılır. İkincisi menü içerisinde gezerken veya bir şeyi onaylamak için ENTER görevi yapar. Üçüncüsü NXT içerisindeki programları çalıştırmak için kullanılır.
- **Açık Gri Ok Butonları:** NXT menü içerisinde sağa ve sola hareket etmek için kullanılır.
- **Koyu Gri Buton:** NXT içerisinde bir önceki işleme yani geri gitmek için kullanılır.

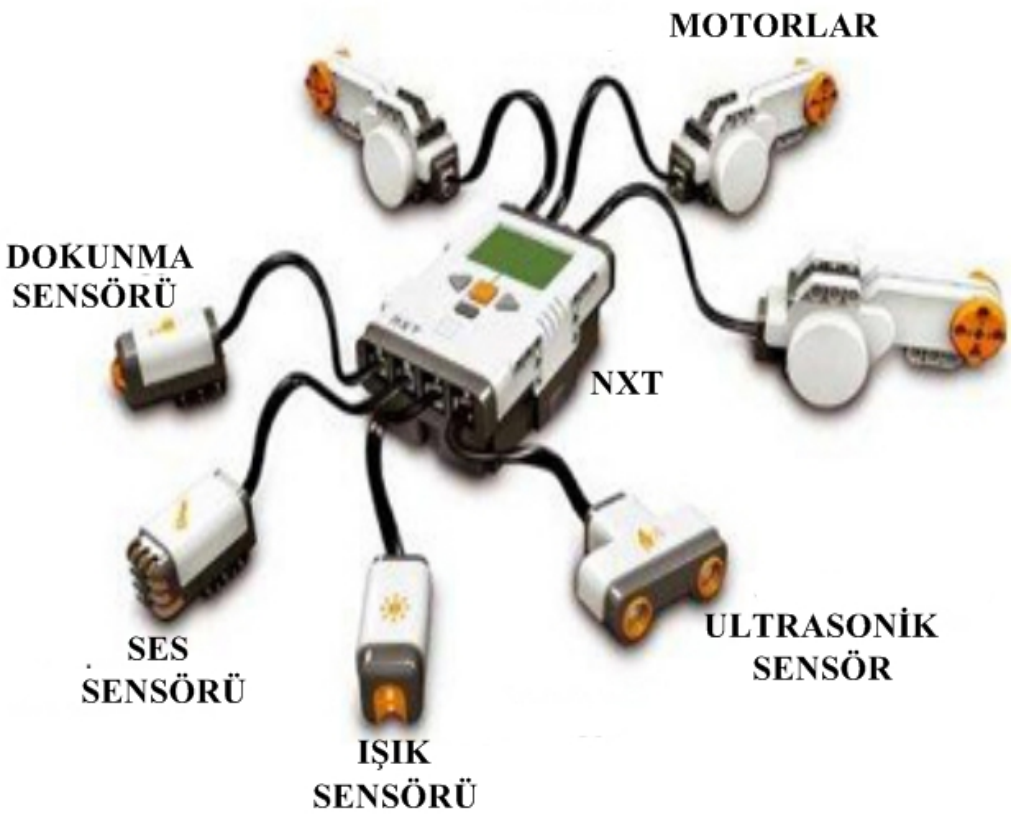
NXT display, 100 x 64 piksellik bir grafik ekrandan oluşmaktadır. Bu ekran sayesinde hem menü içerisindeki ikonları ve görevleri görmek için hem de robot programı sırasında programcı tarafından arayüz olarak kullanılabilir.

NXT'nin teknik özellikleri Çizelge 8.1'de verilmiştir.

Çizelge 8.1 NXT teknik özellikleri

İşlemci 1	32-bit ARM7 Mikrodenetleyici
Hafıza 1	256 Kbytes FLASH, 64 Kbytes RAM
İşlemci 2	8-bit AVR Mikrodenetleyici
Hafıza 2	4 Kbytes FLASH, 512 Byte RAM
Ekran	100 x 64 piksel LCD grafik ekran
Ses	Hoparlör - 8 kHz ses kalitesi. 8 bit çözünürlüklü ve 2-16 kHz örnekleme oranlı ses çıkışı.
Bluetooth	Bluetooth kablosuz iletişim (Bluetooth Sınıf II V2.0)
USB Portu	USB full speed port (12 Mbit/s)
Giriş Bağlantıları	4 adet 6-telli dijital platforma sahip sensör giriş portu (Portlar IEC 61158 Type 4 / EN 50 170 özellikli her biri farklı görev içerebilen bağımsız porttur.)
Çıkış Bağlantıları	3 adet 6-telli dijital platforma sahip motor çıkış portu

Tüm sensör ve motorların NXT gömülü sistemine takılmış hali Şekil 8.3'de gösterilmiştir.



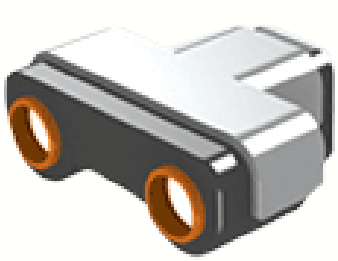
Şekil 8.3 NXT modülüne motor ve sensörlerin bağlantısı

8.1.1.2 Ultrasonik sensör

NXT Mindstorms robotunun iki adet görme sensörlerinden birisidir. (Diğeri ışık sensörüdür.) Ultrasonik sensör, NXT'nin nesneleri görmesini yani algılamasını sağlayan sensördür. Ultrasonik sensör ile robotun engellerden sakınmasını, onları fark etmesini, nesnelere olan mesafesinin ölçülmesini ve hareket eden nesnelerin algılanmasını sağlamaktadır.

Şekil 8.4'de ultrasonik sensörün şekli görülmektedir. Sensör içerisinde birisi sesi gönderen diğeri de sesi alan iki adet ultrasonik sensör bulunmaktadır. Sensörün birinden gönderilen ses bir engele çarparak geri gelir ve diğer sensör tarafından

algılanır. U yöntemde gönderilen sesin hızı bellidir. Sesin engele çarpıp geldiği süre tutulur ve bu süre ile ses hızı çarpılarak engelle olan mesafe bulunmuş olur. Eğer mesafe ölçme işleminde kullanılmayacaksa sadece algılama işlemi için kullanılacaksa gönderilen ses geri geliyorsa bir engel vardır, geri gelmiyorsa engel yoktur şeklinde de kullanılabilir.

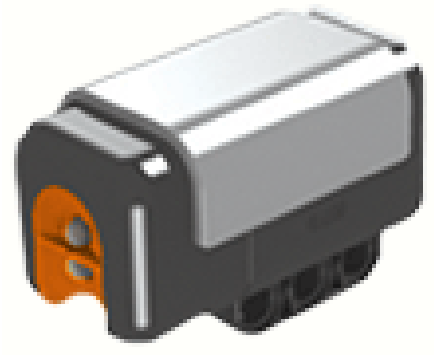


Şekil 8.4 NXT Mindstorms robotunun ultrasonik sensörü

Ultrasonik sensör sayesinde mesafe inç veya santimetre cinsinden ölçülebilmektedir. Sensörden 0 ile 255 cm arasında ±3 cm hata ile mesafe ölçümü yapılabilmektedir. Algılanacak nesneler geniş ve sert madden olursa çok rahat bir algılama yağılabilir. Ama kumaş gibi yumuşak, top gibi yuvarlak olan yüzeyler veya ince ve küçük olan nesnelerden sesin çarpıp gelmesinde bozulma olacağı için algılama zorlaşacak veya hassasiyeti bozulacaktır. Ayrıca aynı ortamda aynı tür ultrasonik sensör bulunursa onun ses dalgalarını da algılayarak bozulmalar olabilir.

8.1.1.3 Işık sensörü

NXT Mindstorms robotunun iki adet görme sensörlerinden birisidir. (Diğeri ultrasonik sensörüdür.) Işık sensörü robotun aydınlık ile karanlık arasındaki koyuluk oranını ayırt etmek için kullanılır. Diğer bir değişle renkler üzerine düşen ışık şiddetini okuyabilmektedir. Örneğin bir odada ki ışık şiddetini veya renkli yüzeyler üzerindeki ışık yoğunluklarını ölçebilir. Şekil 8.5'de ışık sensörü gösterilmiştir.



Şekil 8.5 NXT Mindstorms robotunun ışık sensörü

Şekil 8.6 (a)'dan da görüldüğü gibi insan gözü, renkleri tüm detayları ile algılayabilmektedir. NXT'nin ışık sensörü ise Şekil 8.6 (b)'de görüldüğü gibi renkli yüzey üzerine düşen ışık şiddetinin aydınlık karanlık ölçüsü üzerinden koyuluk oranı ile görmektedir.



(a) (b)

Şekil 8.6 (a) İnsan gözü ile görülen renkler (b) Işık sensörü ile görülen renkler

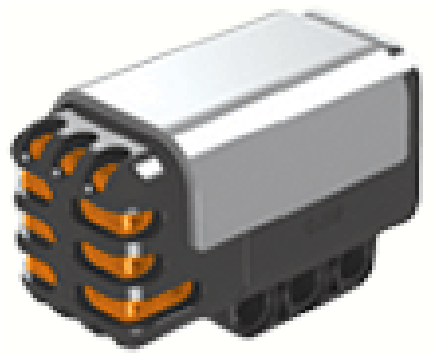
8.1.1.4 Ses sensörü

Ses sensörü ile hem desibel (dB) hem de ayarlanmış desibel (dBA) seslerin ikisi de algılanabilmektedir. Desibel, ses basıncının ölçü birimidir.

- **dBA algılama:** Ayarlanmış desibel algılamada, ses sensörünün duyarlılığı insan kulağının duyma hassasiyetine aralığına ayarlanmıştır. Diğer bir ifade ile kulağımızla duyduğumuz sesleri algılamak için kullanılır.

- **dB algılama:** Standart (ayarlanmamış) desibel algılamada, tüm sesleri eşit hassasiyetle algılayabilmektedir. Yani insan kulağının duyabildiği veya duyamadığı tüm sesleri kapsamaktadır.

Şekil 8.7'de NXT Mindstorms robotunun ses sensörü görülmektedir.



Şekil 8.7 NXT Mindstorms robotunun ses sensörü

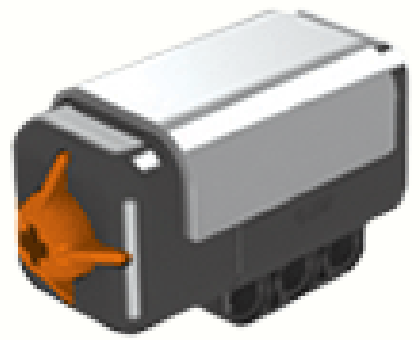
NXT'nin ses sensörü ile 90 dB'e kadar olan tüm ses basınç seviyeleri ölçülebilir. Ses basıncı seviyeleri oldukça karmaşık olduklarından dolayı NXT'nin ses sensörü okuduğu ses basıncı seviyesini yüzde (%) cinsinden göstermektedir. Bu yüzde seviyeleri şu şekilde belirtilebilir;

- % 4 – 5 civarı ses seviyesi, sessiz bir oda ortamı gibidir.
- % 5 – 10 civarı ses seviyesi, biraz uzak mesafede konuşan birisi gibidir.
- % 10 – 30 civarı ses seviyesi, sensöre yakın bir mesafeden ses sensörüne doğru normal bir konuşma yapmak veya normal bir müzik dinleme seviyesi gibidir.
- % 30 – 100 civarı ses seviyesi, insanların bağırması veya çok yüksek sesle müzik dinleme seviyesi gibidir.

8.1.1.5 Dokunma sensörü

Dokunma sensörü, robota direk olarak temas ile algılama yani dokunma duyusu özelliği sağlar. Dokunma sensörü, sensöre bir şey tarafından basıldığında veya basılı olan bir sensör bir şey tarafından bırakıldığında algılama sağlar.

Şekil 8.8'de dokunma sensörü gösterilmiştir. Algılama işlemini uç kısmındaki bir buton sayesinde yapmaktadır.



Şekil 8.8 NXT Mindstorms robotunun dokunma sensörü

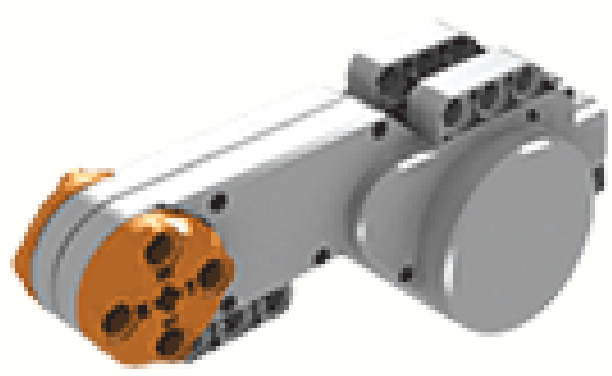
Dokunma sensörü üç durumda algılama yapmaktadır. Bunlar: basma hareketi, serbest bırakma hareketi veya basıp çekme yani çarpma hareketleridir. Bu üç durum Şekil 8.9'da gösterilmiştir.



Şekil 8.9 Dokunma sensörünün algılama şekilleri

8.1.1.6 Servo motorlar

NXT Mindstorms ile tasarlanacak robotlara hareket kabiliyeti kazandıran üç adet servo motorları bulunmaktadır. Şekil 8.10'da servo motorun resmi görülmektedir. Servo motorlar her bir harekette 1° hassasiyete sahiptir.



Şekil 8.10 NXT Mindstorms robotunun servo motoru

Her bir servo motor, içerisinde yerleşik olarak gelen dönme sensörü yani enkoder'a sahiptir. Enkoder sayesinde motorların devirleri ve konumları ±1° hassasiyetle derece olarak ölçülebilmektedir. Bir tam tur dönme 360° şeklinde ayarlanmıştır. Yani yazılımdan 180°'lik bir dönme hareketi yaptırılırsa motora teker yarım tur dönecektir.

Servo motor içerisindeki enkoder ile motorların hızlarına ilişkin bilgilere de ulaşmaya izin vermektedir. Bu sayede her bir servo motora farklı hız değerleri uygulanabilmektedir.

8.1.2 Kablosuz CMOS kamera

Bu tez çalışmasında NXT Mindstorms robotu için kablosuz bir kamera kullanılmıştır. Kullanılan kablosuz CMOS kamera ve alıcı modülü Şekil 8.11'de gösterilmiştir.



Şekil 8.11 Kablosuz CMOS kamera ve alıcı modülü

Kameranın alıcısı ile olan iletişim hızı frekansı 2.4GHz'dir. Ayrıca kameranın içerisinde yerleşik olarak gelen bir adet mikrofon bulunmaktadır. Kapalı olmayan ortamda 100 metre iletişim mesafesi bulunmaktadır. Tez çalışmasında kullanılan kablosuz kameranın teknik özellikleri Çizelge 8.2'de verilmiştir.

Çizelge 8.2 Kablosuz kameranın teknik özellikleri

Resim Sensörü	1/3" OMNIVISION renkli CMOS sensör
Resim Boyutu	PAL: 628(Y) x 582(G) NTSC: 510(Y) x 492(G)
Yatay Çözünürlük	380 TV satırı
İletim Gücü	10mW
Modülasyon Modu	FM modülasyon
Band Genişliği	20 MHz
İletişim Mesafesi	Kapalı alan 30m mesafe - Açık alan 100m mesafe
Güç	DC 8V 200 mA
Lens	6 mm lens

Kablosuz kamera alıcısının teknik özellikleri de Çizelge 8.3'de verilmiştir.

Çizelge 8.3 Kablosuz kamera alıcısının teknik özellikleri

Frekansı	2400 – 2483 Mhz
Anten Direnci	50 ohm
Frekans Kararlılığı	± 100 KHz
Demodülasyon Modu	FM demodülasyonu
Video Çıkışı	1Vp-p@75 ohm, S/N>38dB
Ses Çıkışı	1Vp-p@600 ohm
Güç	DC 12V

8.2 Yazılım

Gömülü sistemler temelinde bilgisayar donanımı ve yazılımına sahip oldukları için gömülü sistem için yazılım geliştirme yelpazesi oldukça geniş olmaktadır. Çünkü kişisel bilgisayarlar için yazılan hemen hemen bütün yazılım geliştirme araçlarını desteklemektedir. Örneğin National Instrument firması tarafından Lego NXT Mindstorms için geliştirilen NXT programı, Visual Basic.Net, Visual C#.Net, MATLAB, Microsoft Robotics Studio…vb. programlar kullanılabilmektedir.

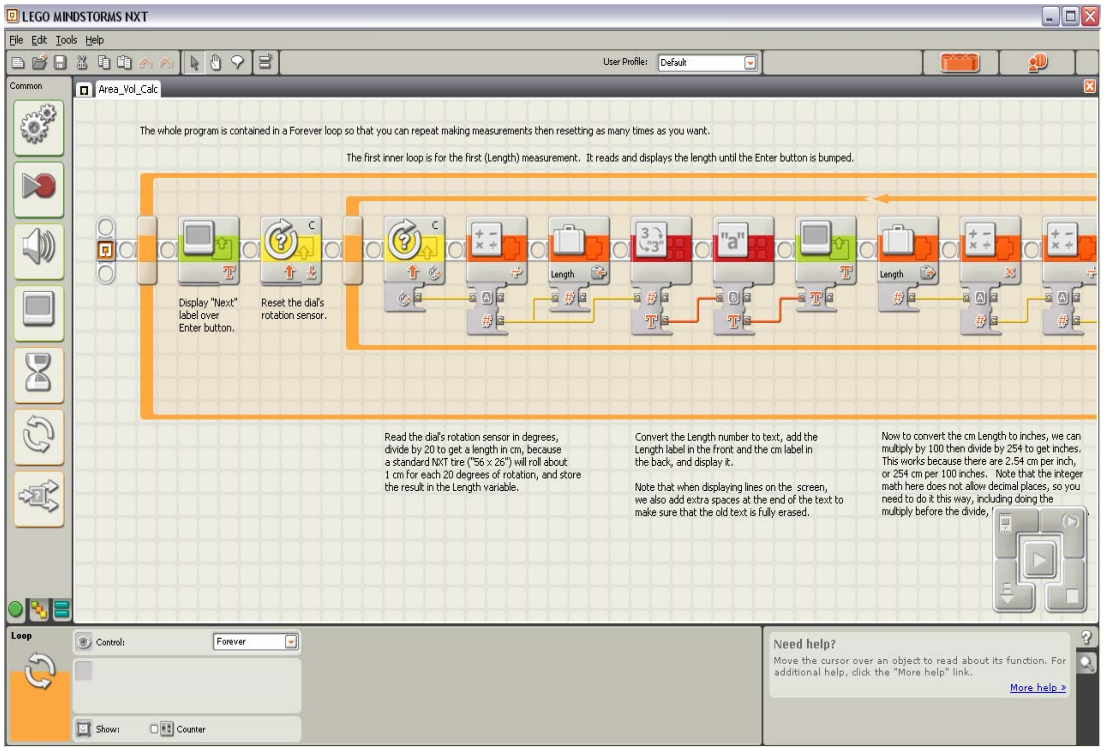
Hatta bazı profesyonel yazılımcılar, firmalar veya robot üreticileri tarafından tasarlanan birçok yazılım da gömülü sistemlerle uyumlu bir şekilde çalışmaktadır. Sorunsuz çalışmasının en büyük sebebi gömülü sistemin aslında bir bilgisayar tabanı üzerine kurulmasından kaynaklanmaktadır. Bu sayede yazılım süreci hem kolaylaşmakta hem de hızlanmaktadır.

Bu bölümde yazılım geliştirme ortamlarının sadece 2 tanesinden bahsedilecektir. Bunlar; Lego NXT Mindstorms için National Instrument firması tarafından geliştirilen NXT programı, Lego NXT Mindstorms için bir üniversite tarafından geliştirilen MATLAB araç kutusu programıdır.

8.2.1 Mindstorms için geliştirilen NXT programı

Elektronik tasarım simülasyonu ile uğraşan kişilerin kullandığı popüler LabView programının geliştiricisi National Instrument firmasının, Lego NXT Mindstorms robotu için özel olarak geliştirdiği gelişmiş bir programdır. Özel olarak geliştirildi denilmesinin en büyük nedeni Lego NXT Mindstorms robotunu aldığınız zaman bu yazılım standart olarak gelmektedir.

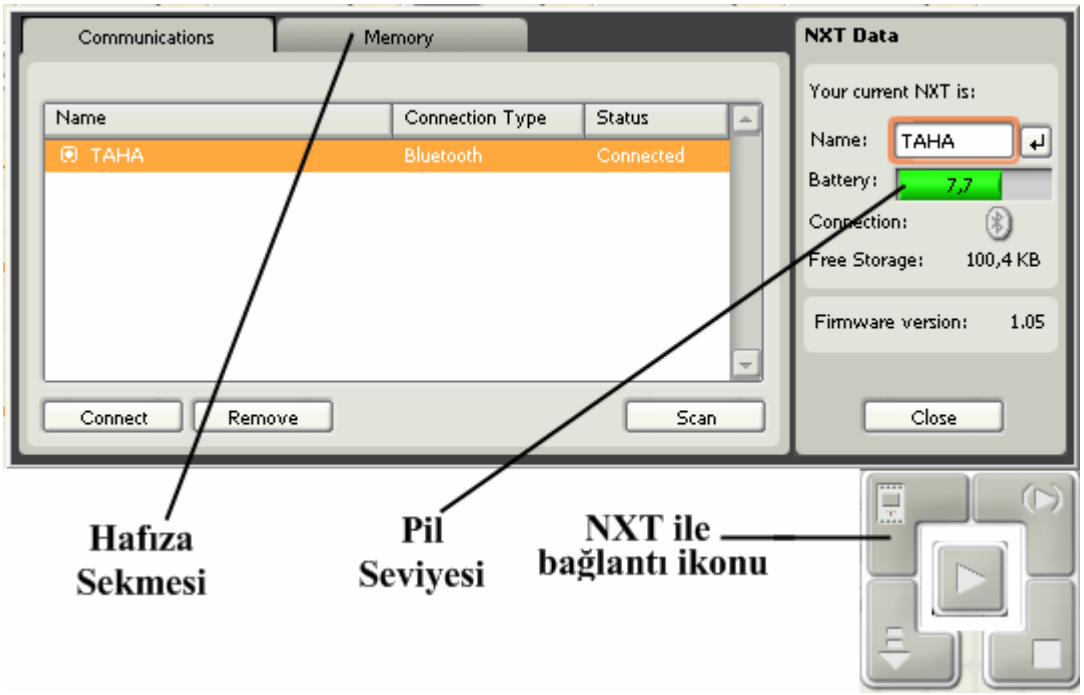
Şekil 8.12'de programın tasarım yüzeyinin bir görüntüsü görülmektedir.



Şekil 8.12 National Instrument tarafından geliştirilen NXT programı

NXT programının kullanımı da Microsoft Robotics Studio programına benzer bir mantık izlemektedir. Bu program içerisinde de yazılacak program sürükle–bırak mantığı ile tasarlanmakta ve MSRS'deki servisler gibi tasarım ekranının sol tarafındaki ikonların tasarım yüzeyinde birbirine bağlanması ile program yazılmaktadır.

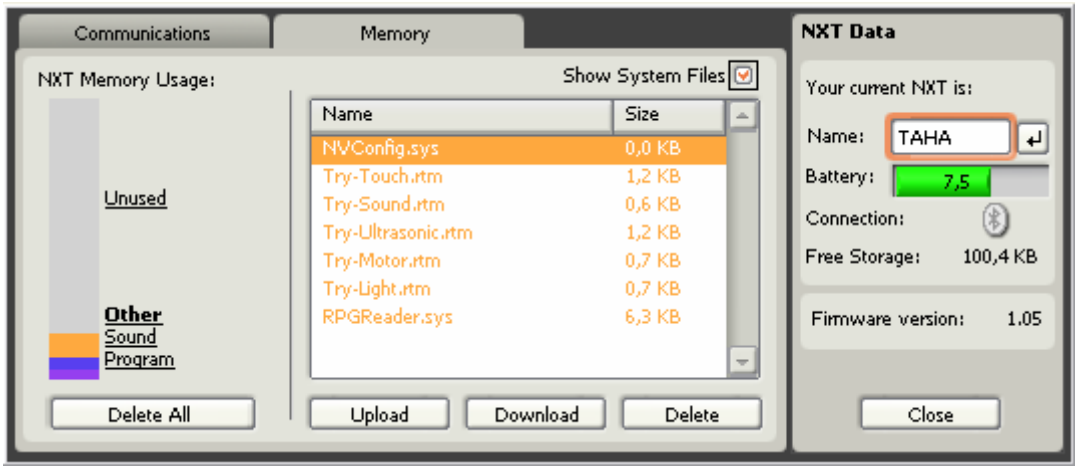
Şekil 8.13'de görüldüğü gibi bu programda NXT ile bağlantı kurmak için tasarım yüzeyinin sağ alt köşesinde bulunan kontrol tuşlarından NXT bağlantı ikonuna tıklayarak bağlantı kutusu açılır ve bağlantı gerçekleştirilir. Bağlantı türü ister USB portundan ister bluetooth modülünden sağlanabilir.



Şekil 8.13 NXT bağlantı kutusu

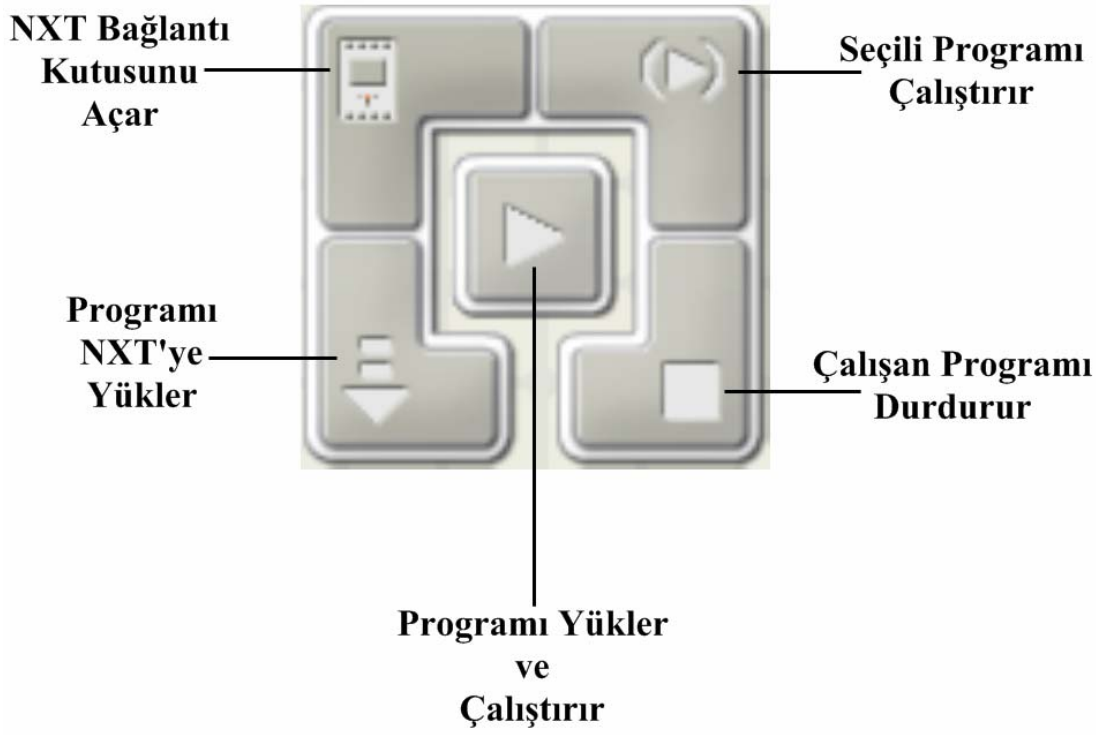
NXT bağlantı kutusunun "Hafıza" (Memory) sekmesinden NXT gömülü sistemi içersindeki tüm program ve diğer dosyalara ulaşılabilir. Buradan yeni programlar veya dosyalar yüklenebilir, indirilebilir ve silinebilir.

Şekil 8.14'de NXT bağlantı kutusunun NXT Data bölümünden de NXT'nin ismini değiştirilebilir veya pil seviyesini takip edilebilir.



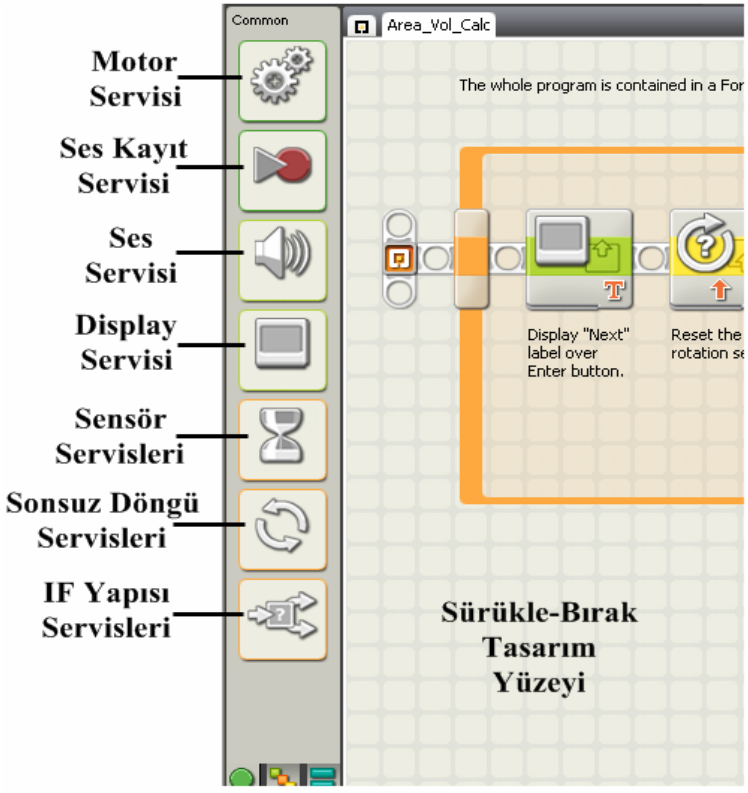
Şekil 8.14 NXT bağlantı kutusunun hafıza bölümü

NXT için geliştirilen NXT programında yazılan programı kontrol eden ve NXT'ye derleyip yükleme yapabilen Şekil 8.15'deki kontrol butonları kullanılmaktadır.



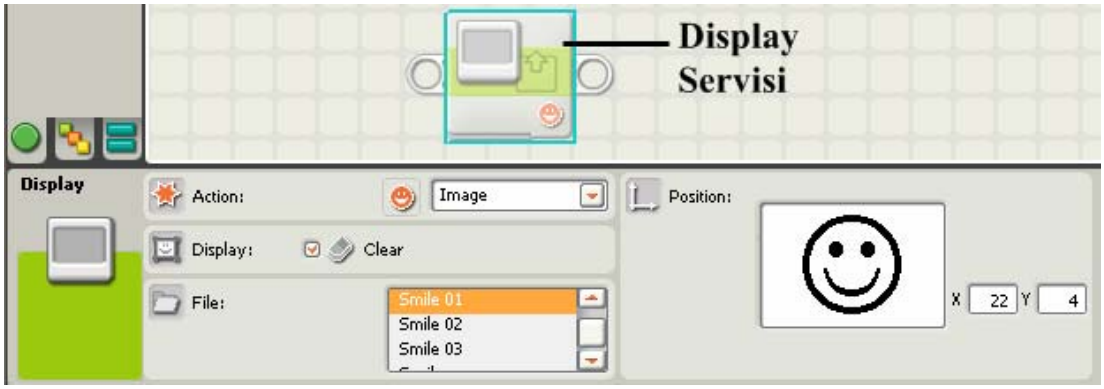
Şekil 8.15 NXT programı kontrol butonları

Lego Mindstorms NXT robotunu NXT programı ile kontrol etmek için Şekil 8.16'da görüldüğü gibi NXT programının sol kenarındaki Genel (Common) servis kutusundaki servisler kullanılarak yazılım gerçekleştirilir. Kullanılacak servis seçildikten sonra tasarım yüzeyine mouse ile sürükleyerek servis yerleştirilir. Bu şekilde gerekli servisler yerleştirildikten sonra servisler arasındaki gerekli bağlantılar yapılarak programlama işlemi tamamlanır. Kontrol butonları kullanılarak program NXT içerisine yüklenir ve çalıştırılır.



Şekil 8.16 NXT programı kontrol servisleri

Tasarım yüzeyine yerleştirilen bir servisin ayarlarını yapabilmek için önce mouse ile ayar yapılacak servis tek tıklama ile seçilir. Sonra NXT programının alt kenarında servis ile ilgili ayarlar bölümü açılacaktır. Burada açıklamalar doğrultusunda ne yapılması gerekiyorsa gerekli veriler girilerek servis ayarları tamamlanır. Örneğin Şekil 8.17'de display servisinin ayar menüsü görülmektedir.



Şekil 8.17 Display servisi ve ayar menüsü

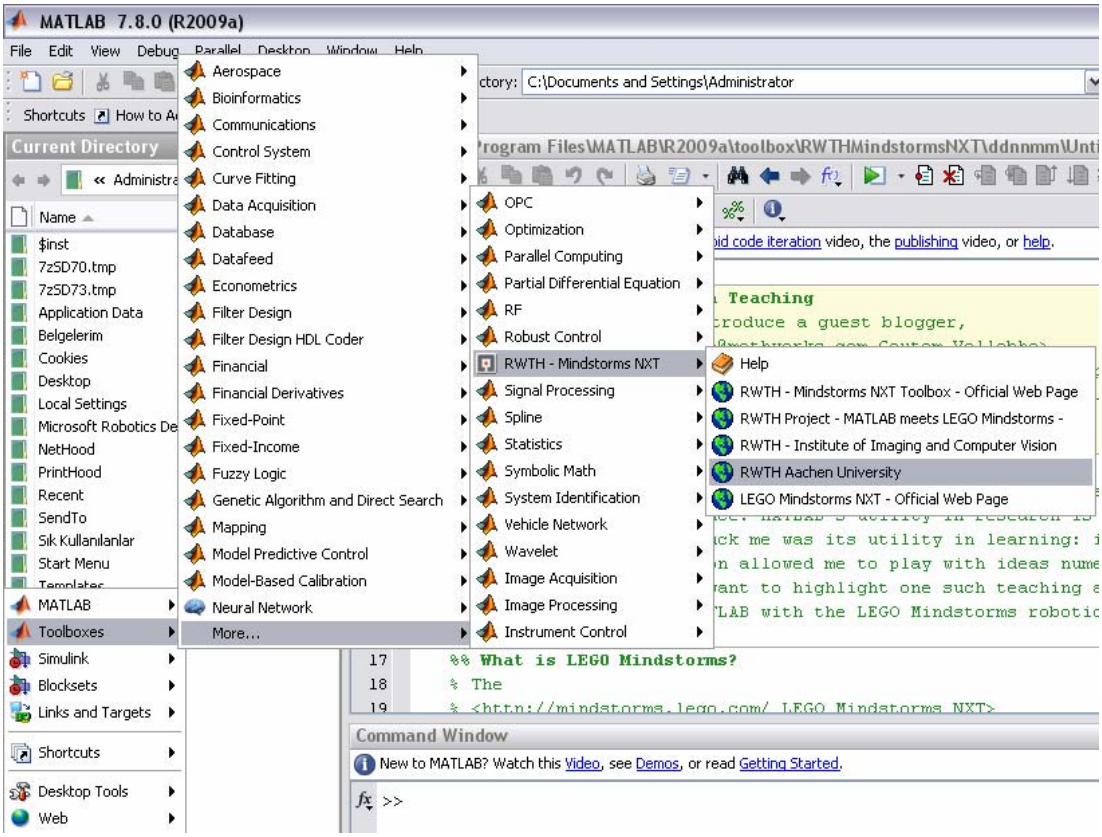
8.2.2 RWTH Mindstorms NXT MATLAB araç kütüphanesi

Almanya'da büyük bir teknoloji üniversitesi olan RWTH Aachen Üniversitesi, MATLAB programına Lego NXT Mindstorms robotunu kontrol etmek için bir araç kütüphanesi (toolbox) geliştirmiştir. Bu araç kütüphanesi sayesinde Lego NXT Mindstorms robotu MATLAB programı ile kontrol edilebilmektedir.

Lego'nun NXT programı ve Microsoft Robotics Studio programından farklı olarak MATLAB'de yazılımı geliştirmek sürükle-bırak mantığı şeklinde değil klasik satır satır komut yazarak yapılmaktadır.

RWTH NXT araç kütüphanesi "http://www.mathworks.com/matlabcentral/fileexchange/18646" adresinden ücretsiz olarak temin edilebilir. Araç kütüphanesi elde edildikten sonra MATLAB araç kütüphanesine eklenmelidir. MATLAB araç kütüphanesine eklenmiş hali Şekil 8.18'de gösterilmiştir.

MATLAB'de program yazmanın en büyük avantajlarından birisi de geliştirilen robot yazılımı için kullanılan araç kütüphanesinin yanında, MATLAB'deki diğer tüm araç kutularını da yazılımınıza dâhil edilebilmektir. Bu durumun çok avantajları vardır. Örneğin geliştirilen robot yazılımı içerisinde görüntü işleme gerekirse bunun için MATLAB görüntü işleme araç kütüphanesinden faydalanılabilir. Bu sayede robot yazılımı geliştirilmesi sırasında hem yazılım süresi oldukça kısalacaktır hem de robot yazılımı dışında görüntü işleme gibi başka bir alanda yazılım geliştirmek durumu ortadan kalmış olacaktır.

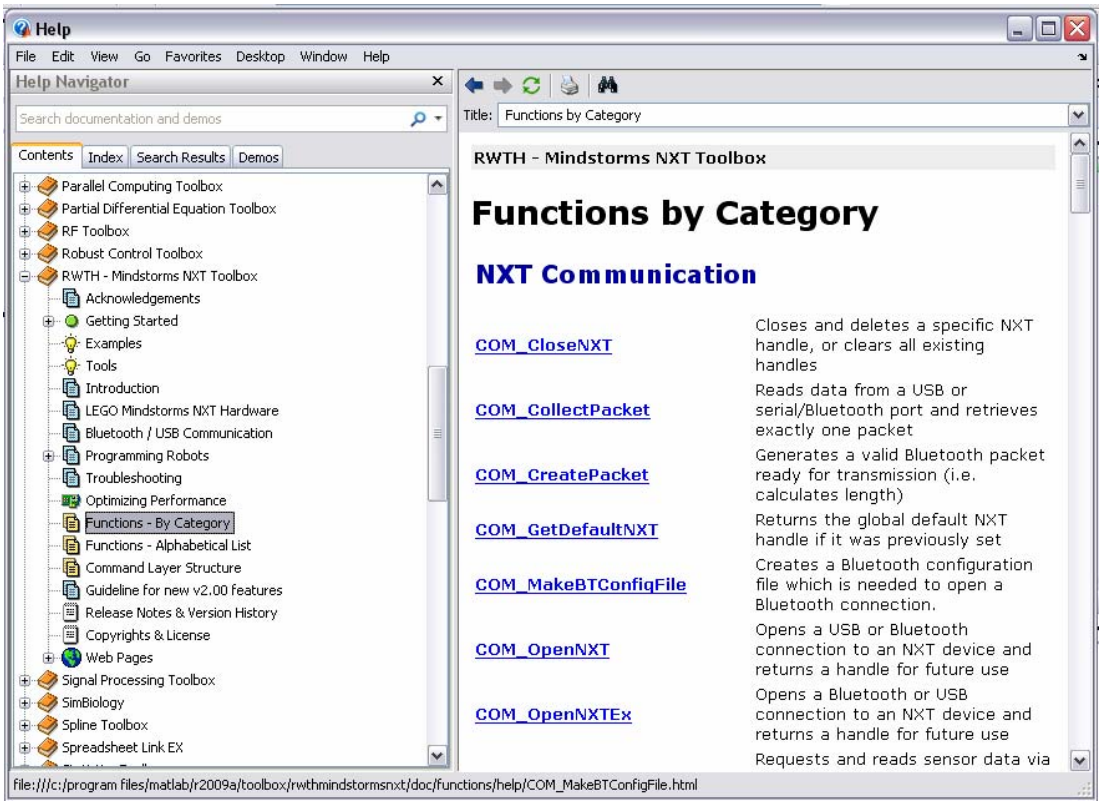


Şekil 8.18 RWTH Mindstorms NXT MATLAB araç kütüphanesi

RWTH Mindstorms NXT MATLAB araç kütüphanesi ile gelen robot kontrol fonksiyonlarının ne işe yaradığı ve nasıl çalıştıkları ile ilgili bilgiye yine araç kütüphanesi ile gelen yardım menüsünden ulaşılabilir.

Şekil 8.19'da görüldüğü gibi fonksiyonların program içerisindeki kullanıldıkları isimleri, ne işe yaradıkları, fonksiyonun nasıl ve hangi parametrelerle kullanıldıkları detaylı bir şekilde anlatılmaktadır.

Bu tez çalışmasının asıl konusu klasik programlama yapısı olan satır satır programlama mantığı olmadığı için RWTH Mindstorms NXT MATLAB araç kütüphanesi ve fonksiyonları hakkında detaylı bilgi verilmemiştir. Ancak RWTH Mindstorms NXT MATLAB araç kütüphanesi ile ilgili olarak Bölüm 9 içerisinde bir örnek üzerinde anlatılmıştır.



Şekil 8.19 RWTH Mindstorms NXT MATLAB araç kütüphanesi yardımı

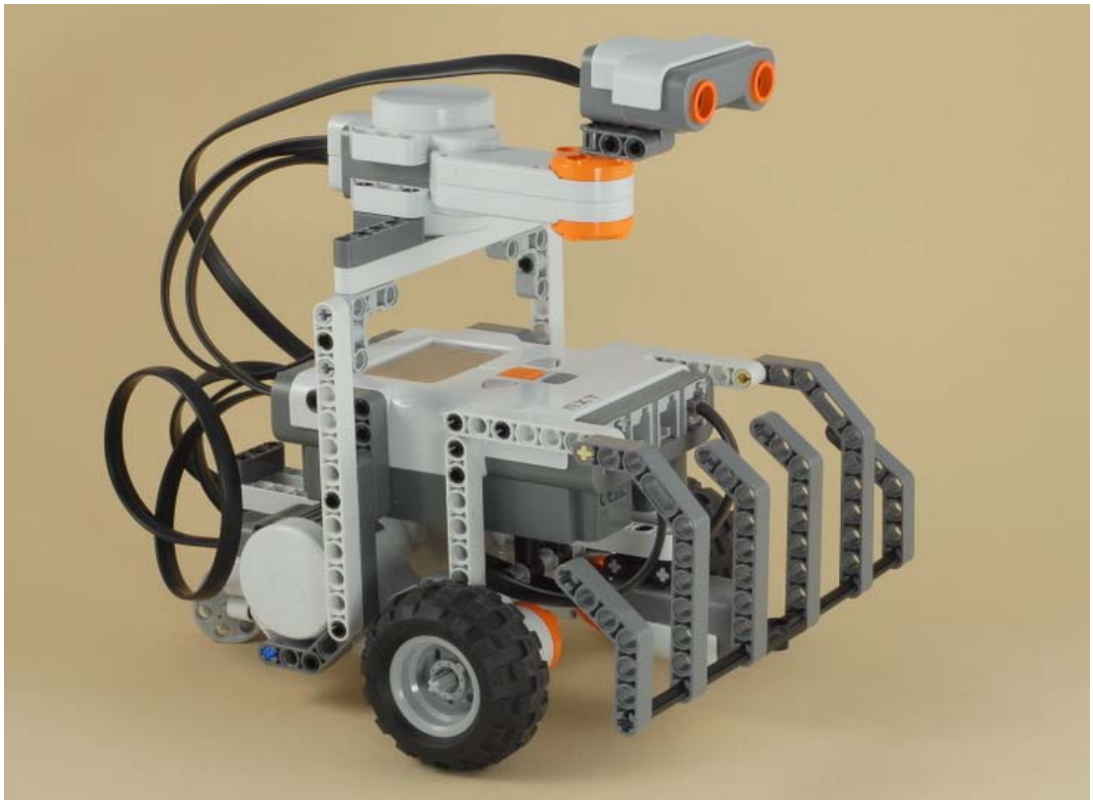
9 DENEYSEL SONUÇLAR

Tez çalışmasının bu bölümünde, tez boyunca anlatılan 3 programın gerçek ortamdaki uygulamaları yapılmış ve sonuçları gözlemlenmiştir. Programların tamamı Lego NXT Mindstorms robotu üzerinde denenmiştir.

9.1 Mindstorms için Geliştirilen NXT Programı Uygulaması

NXT programı için Lego NXT Mindstorms robotu ile "Kaşif" isimli bir robot tasarlanmış ve robotun hareket edebileceği herhangi bir ortamda engellere takılmadan hareket etmesi için NXT programında gerekli yazılım yapılmıştır.

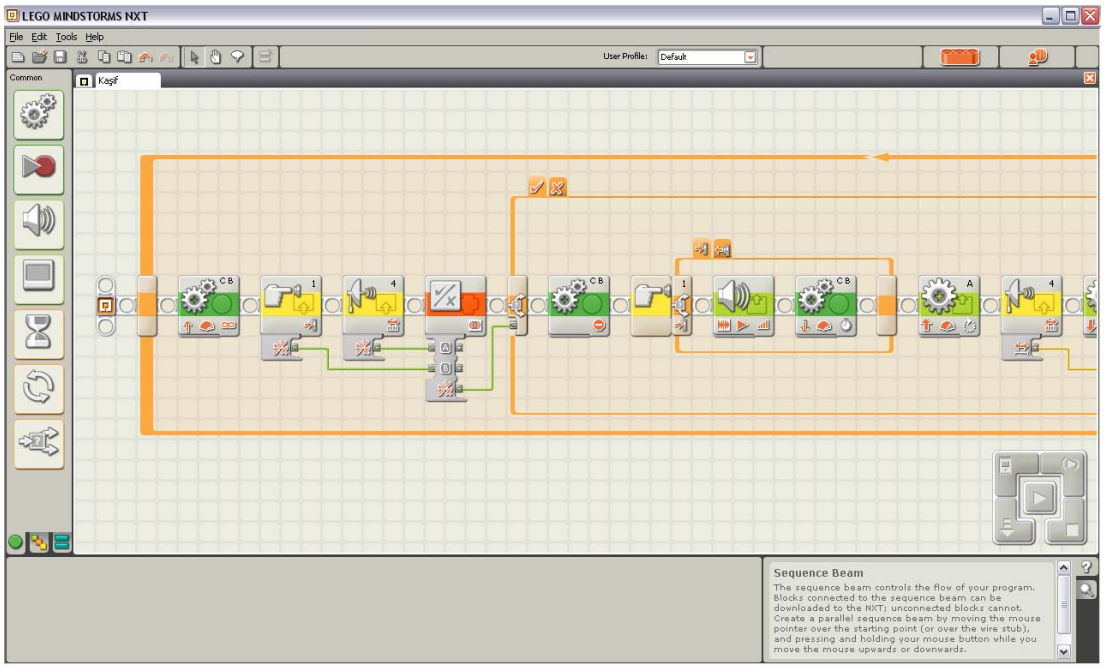
Şekil 9.1'de tasarlanan Kaşif isimli robot görülmektedir.



Şekil 9.1 Kaşif robotu

Kaşif robotun beyni yani gömülü bilgisayar sistemini oluşturan NXT modülü 2 sensör ve 3 servo motor kontrolü yapmaktadır. Sensörler bir adet ultrasonik sensör ve bir adet dokunma sensöründen oluşmaktadır. Ultrasonik sensör etrafını 180 derece açı ile tarayarak nesneleri algılamasını sağlamaktadır. Dokunma sensörü ise ultrasonik sensörün algılamadığı nesnelere dokunarak algılama yapmasını sağlamaktadır. Servo motorların iki tanesi Kaşif robotun zemin üzerindeki hareketini bir tanesi de ultrasonik sensörü 180 derece açı tarama işlemini gerçekleştirebilmesi için kullanılmıştır.

Kaşif robotun yazılımı Şekil 9.2'de görüldüğü gibi Lego NXT Mindstorms robotu için National Instrument tarafından özel olarak geliştirilen NXT programında geliştirilmiştir.



Şekil 9.2 NXT programındaki Kaşif robotu programı görüntüsü

Kaşif robotuna ilk önce ileri yönde bir hareket verilir. Daha sonra dokunma sensöründen ve ultrasonik sensörden gelen bilgiler VEYA (OR) işlemine tabi tutulur. Bu işlem sonucunda herhangi birisinden bir engel olduğuna dair bir cevap gelmişse ileri yönde hareket ettirilen robotlar hemen durdurulur. Sonra engel algılandı cevabının hangisinden geldiğine bakılır. Eğer cevap dokunma sensöründen gelmişse

gittiği yönün tam tersi istikamette motorlara belli bir süre kadar geri dönmesi emri verilir. Böylece Kaşif robotunun dokunarak algıladığı engelden kurtulması sağlanır. Eğer ultrasonik sensörden engel cevabı gelmişse hiçbir işlem yapılmasına gerek yoktur. Çünkü ultrasonik sensörler sayesinde engele çarpmadan algılanması sağlanmıştır. Daha sonra Kaşif robotun engel olmayan bir yön belirlemesi için ultrasonik sensör robotun bulunduğu yönün 90 derece sağı ve 90 derece solu kontrol edebilmesi için üçüncü motor önce 90 derece sağa sonra 90 derece sola çevrilerek kontrol sağlanır. Bu çevrilme işlemleri gerçekleştikleri sırada ultrasonik sensörden gelen bilgiler kontrol edilerek önünde engel bulunmayan yöne doğru motorlar çevrilir. Bütün bu işlemler bittikten sonra program bir döngü ile tekrar başa döndürülerek ileri yönde hareket etmesi sağlanır.

Gömülü bilgisayar sisteminin bir robot tasarımında kullanılmasının avantajı yazılım geliştirilmesi aşamasında oldukça ön plana çıkmıştır. Böyle bir özellikli robotu gömülü bilgisayar sistemi olmadan tasarlanacak bir robotta yapılmış olsaydı, öncelikle donanım anlamında çok fazla uğraşmak gerekecekti. Çünkü motorların açı kontrolü için enkoder bilgisini okumayı sağlayacak donanıma, ultrasonik sensörü ile mesafe algılamayı sağlayan bir donanıma ve Kaşif robotun yaptığı işleri hızlı bir şekilde kontrol edecek mikrodenetleyiciye ihtiyaç duyulacaktır. Geliştirilecek yazılımında eş zamanlı olarak bütün bu işlemlere yetişebileceği bir alt yapıya sahip olması gerekmektedir.

9.2 RWTH Mindstorms NXT MATLAB Araç Kütüphanesi Uygulaması

Lego NXT Mindstorms robotu ile “Akrep” isimli bir robot tasarlanmış ve robotun hareket edebileceği herhangi bir ortamda engellere takılmadan hareket etmesi için RWTH Mindstorms NXT MATLAB araç kütüphanesi programında gerekli yazılım yapılmıştır.

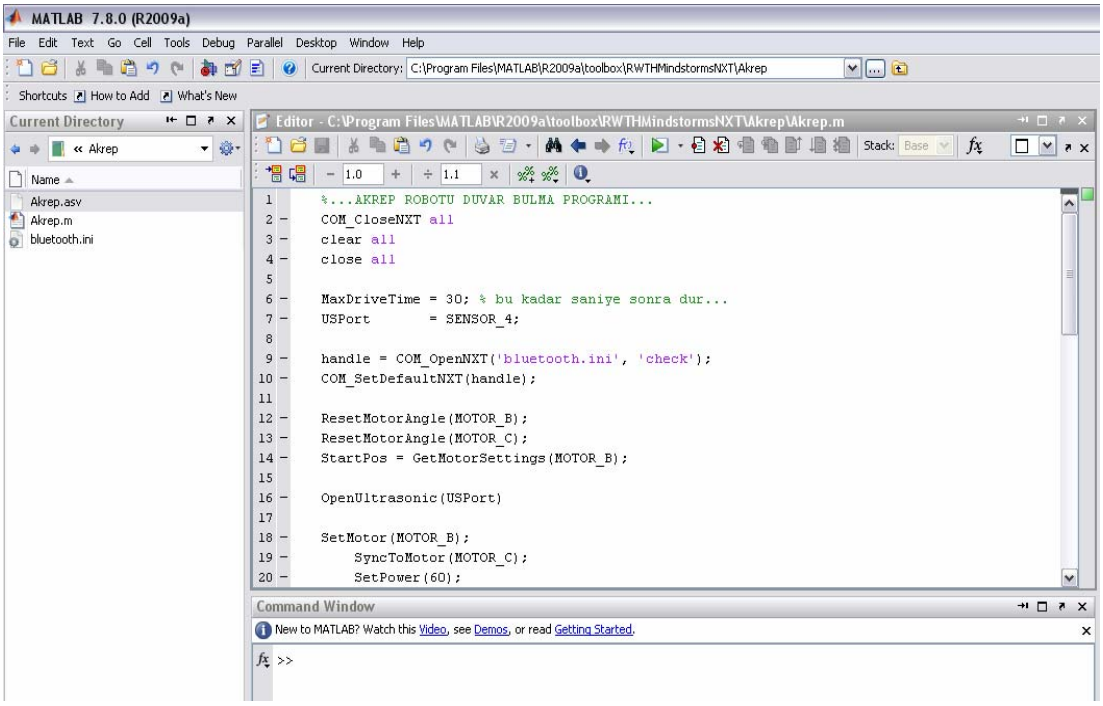
Şekil 9.3’de tasarlanan Akrep isimli robot görülmektedir.



Şekil 9.3 Akrep robotu

Akrep robotun gömülü bilgisayar sistemini oluşturan NXT modülü 1 sensör ve 2 servo motor kontrolü yapmaktadır. Sensör için bir adet ultrasonik sensör kullanılmıştır. Ultrasonik sensör robotun hareket yönündeki duvarın olup olmadığını algılamaktadır. Servo motorlar Akrep robotunun zemin üzerindeki hareketini gerçekleştirebilmesi için kullanılmıştır.

Akrep robotun yazılımı Lego NXT Mindstorms robotu için RWTH Aachen Üniversitesi tarafından özel olarak geliştirilen RWTH Mindstorms NXT MATLAB araç kütüphanesi ile MATLAB programında geliştirilmiştir. Şekil 9.4'de Akrep robotunun MATLAB yazılım ortamı görüntüsü verilmiştir.



Şekil 9.4 Akrep robotunun MATLAB yazılım ortamı görüntüsü

Akrep robotuna ileri yönde bir hareket verilmiş ve motorlardan elde edilen bilgilerle robotun tekrar başladığı noktaya dönmesi sağlanmıştır. Gömülü bilgisayar sistemli mobil robot yazılımının MATLAB programı kullanılarak farklı bir geliştirme ortamında da yazılması amaçlanmıştır.

Programın başlarında ilk önce MATLAB programı ve Akrep robotunun gömülü bilgisayar sistemi olan NXT ile ilgili bağlantı ve port ayarlama komutları yapılır. Daha sonra motorun başlangıç noktasındaki enkoder bilgileri ve açı bilgileri alınır. Sonra Akrep robotuna ileri yönde bir hareket verilir. Bu ileri yöndeki hareket karşına bir duvar çıkana kadar veya 30 saniye boyunca devam ettirilir. İleri yöndeki hareket tamamlandıktan sonra ne kadar mesafe geldiğimiz ve ne konumda olduğumuz bilgileri motorlardan gelen bilgilerle elde edilir. Bu bilgilerden ilk durumdaki bilgiler kıyaslanarak Akrep robotuna 180 derece ters yönde ne kadar gitmesi gerektiği bildirilir. Böylece bu değerler motorlara uygulandığında Akrep robotu başlangıç noktasına geri döner.

RWTH Mindstorms NXT MATLAB araç kütüphanesi kullanılarak MATLAB programında geliştirilen yazılım aşağıdaki kodda verilmiştir.

```
%...AKREP ROBOTU DUVAR BULMA PROGRAMI...
COM_CloseNXT all
clear all
close all

%...Süre ve Portlar belirleniyor...
MaxDriveTime = 30; % bu kadar saniye sonra dur...
USPort       = SENSOR_4;

%...Bluetooth'dan bağlantı için gerekli ayarlar yapılıyor...
handle = COM_OpenNXT('bluetooth.ini', 'check');
COM_SetDefaultNXT(handle);

%...Motorun başlangıç değerleri alınıyor...
ResetMotorAngle(MOTOR_B);
ResetMotorAngle(MOTOR_C);
StartPos = GetMotorSettings(MOTOR_B);

%...Ultrasonik sensör aktif hale getiriliyor...
OpenUltrasonic(USPort)

%...Akrep robotuna ileri yöndeki hareket veriliyor...
SetMotor(MOTOR_B);
SyncToMotor(MOTOR_C);
SetPower(60);
SetAngleLimit(0);
SetTurnRatio(0);
SetRampMode ('off');

%...Değerler motorlara uygulanır...
SendMotorSettings();

%...Duvar gelene kadar veya 30 saniye boyunca harekete devam
et...
tictic(1);
while(toctoc(1) < MaxDriveTime)
   if GetUltrasonic(USPort) < 30
       break
   end%if
end%while

%...Motorları durdurur...
StopMotor('all', 'off');
% ...şimdi neredeyiz bilisi alınır...
CurPos = GetMotorSettings(MOTOR_B);
```

```
%...Hoparlörden BEEP sesi verilir....
NXT_PlayTone(440, 1000);

%...Başlangıç konumu hesaplanır...
TotalAbsDist = StartPos.TachoCount - CurPos.TachoCount;
Dist         = CurPos.Angle;

%...Başlangıç yerine gitmesi için motorlara gerekli bilgiler
yüklenir...
SetMotor(MOTOR_B);
SyncToMotor(MOTOR_C);
SetPower(-60);
SetAngleLimit(abs(Dist));

%...Değerler motorlara uygulanır...
SendMotorSettings();

%...Motor işlemi bitene kadar bekle...
WaitForMotor(1, MaxDriveTime);

%...Motorları durdur...
StopMotor('all', 'brake');

pause(1);

%...Sensörü ve motorları durdur...
CloseSensor(SENSOR_4)
StopMotor ('all', 'off');

%...Bluetooth bağlantısını kes...
COM_CloseNXT(handle);
```

RWTH Mindstorms NXT MATLAB araç kütüphanesi kullanılarak MATLAB programında yazılan bu programla gömülü bilgisayar sistemlerinin farklı bir yazılım geliştirme platformunda da yapılabildiği ve bunun gibi daha pek çok ileri yazılım geliştirme programları ile yapılabileceği gösterilmiştir. Böylelikle robotik alanında veya başka tasarım alanlarında gömülü bilgisayar sistemlerinin yazılım geliştirme işlemini kolaylaştırdığı ve yazılım geliştirme süresini görülebilir ölçüde azalttığı gösterilmeye çalışılmıştır.

9.3 Microsoft Robotics Studio Uygulaması

Lego NXT Mindstorms robotu ile Bölüm 9.1'de tasarlanan "Kaşif" isimli robota bir web kamera takılmıştır ve robotun hareket edebileceği herhangi bir ortamda engellere takılmadan hareket etmesi için Microsoft Robotics Studio programında gerekli yazılım geliştirilmiştir.

Şekil 9.5'de tasarlanan Web kameralı Kaşif isimli robot görülmektedir.

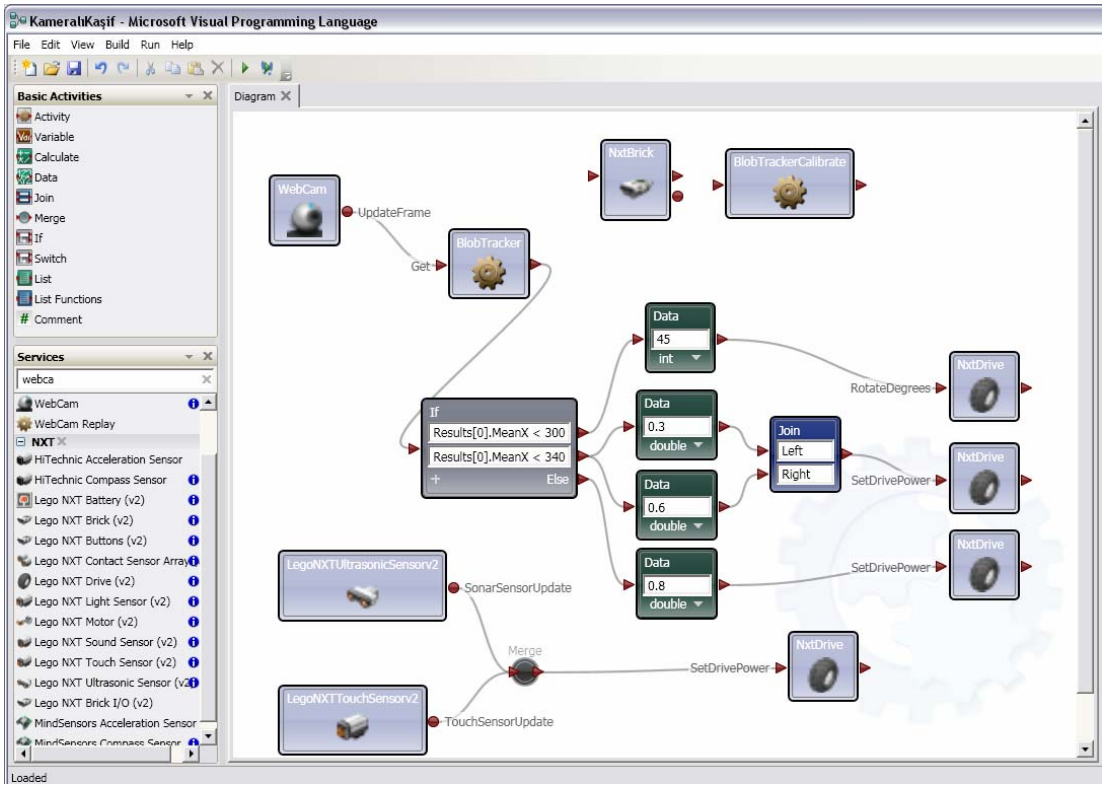


Şekil 9.5 Web kameralı Kaşif robotu

Web kameralı Kaşif robotunun gömülü bilgisayar sistemini oluşturan NXT modülüne bir adet ultrasonik sensör ve bir adet dokunma sensörü takılmıştır. Ultrasonik sensör etrafını 180 derece açı ile tarayarak nesneleri algılamasını sağlamaktadır. Dokunma sensörü ise ultrasonik sensörün algılamadığı nesnelere dokunarak algılama yapmasını sağlamaktadır. İki tane servo motor web kameralı Kaşif robotun zemin üzerindeki hareketini bir tanesi de ultrasonik sensörü 180 derece açı tarama işlemini gerçekleştirebilmesi için kullanılmıştır. Kaşif robotuna takılan

web kamera ile beyaz bir zemin üzerindeki siyah çizgiyi görüntü işleme teknikleri ile belirleyerek web kameralı kaşif robotun çizgi boyunca hareketini sağlayacak bilgileri üretecektir.

Web kameralı Kaşif robotun yazılımı Şekil 9.6'da görüldüğü gibi Lego NXT Mindstorms robotu için Microsoft firması tarafından özel olarak geliştirilen Microsoft Robotics Studio programında geliştirilmiştir.



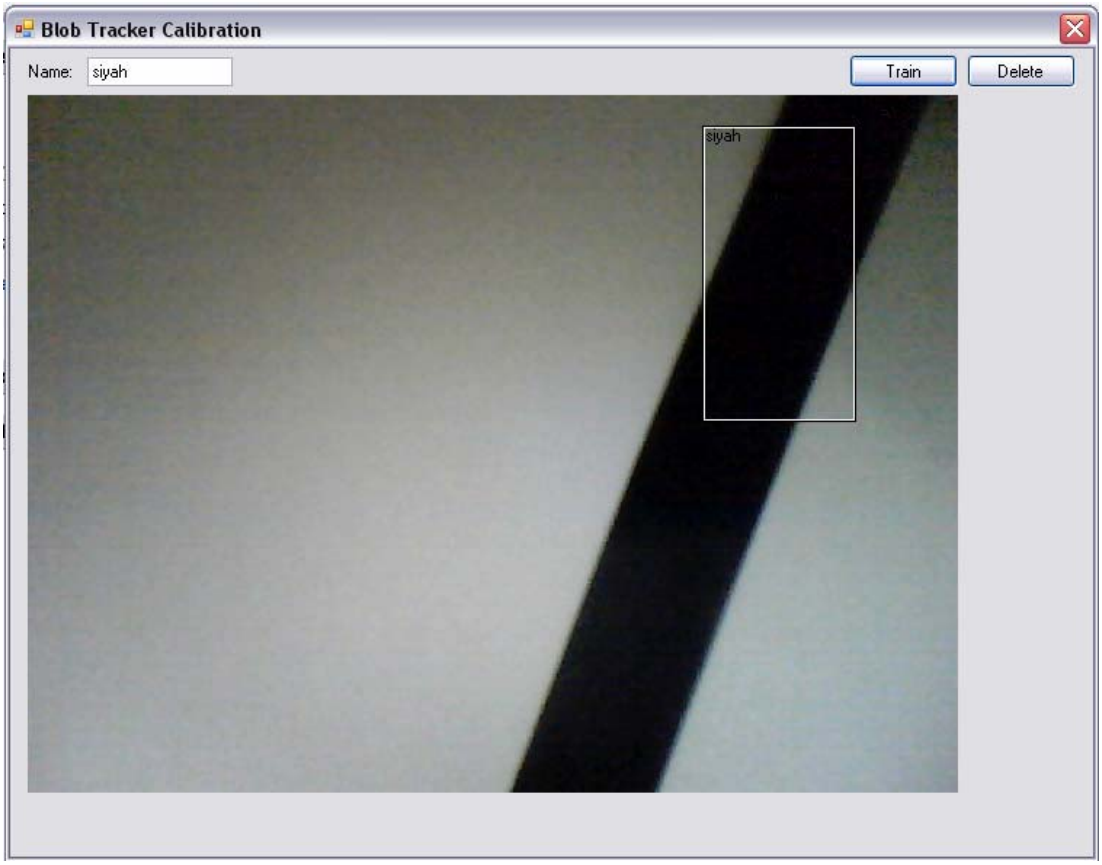
Şekil 9.6 Microsoft Robotics Studio programı

Web kameralı Kaşif robotu Bölüm 9.1'de tasarlanan Kaşif robotunun ultrasonik sensör ve dokunma sensörü ile gerçekleştirdiği özellikleri korumaktadır. Eklenen web kamera ile robotun hareket ettiği ortamda bulunan siyah bir çizginin görüntüleri alınır ve görüntü işleme teknikleri ile bu siyah çizginin görüntü üzerindeki koordinatları belirlenir. Daha sonra alınan bu koordinat bilgisi ile robotun resmin merkezinden ne kadar ve hangi yöne kaydığı tespit edilir. Tespit edilen mesafe kadar ve algılanan kayma yönünün tersine robotun hareket etmesi sağlanır. Bu sayede

hareket ortamında bulunan zemin üzerindeki siyah çizgi sürekli olarak web kameradan alınan görüntülerin ortasında kalması sağlanmaya çalışılır. Böylece web kameralı Kaşif robotu zemin üzerindeki siyah çizgiyi sürekli olarak takip etmektedir.

Eğer hareket sahası içerisinde bir engel ile karşılaşırsa engelden sakınmak için ultrasonik sensör ve dokunma sensörü bilgilerini de es zamanlı olarak işlemektedir. Bu bilgilere göre engelden uzaklaşmaktadır ve bulduğu ilk siyah çizgiyi tekrar takip etmeye devam etmektedir.

Web kamera ile 640 x 480 piksellik alınan görüntü ve takip edilen siyah çizgi Şekil 9.7'de gösterilmiştir. Resimden de görüldüğü gibi beyaz zemin üzerindeki siyah çizgi görüntü işleme teknikleri ile alınıyor ve koordinat bilgileri robota bildiriliyor. Gerekli hesaplamalar yapıldıktan motorlara gerekli değerler yüklenerek siyah çizginin 640 x 480 piksellik resmin 300. ile 340. pikseller arasında kalması sağlanıyor.



Şekil 9.7 Web kamera ile takip edilen siyah çizgi

Tez çalışmasında detaylı bir şekilde anlatılan Microsoft Robotics Studio ile gömülü bilgisayar sistemine sahip bir mobil robot üzerinde görüntü işleme gibi kompleks işlemlerin gerçekleştirilmesinin ne kadar kolaylaştığı bu uygulama ile gösterilmeye çalışılmıştır. Gömülü bilgisayar sistemleri ile tasarlanan sistemlerin ve özellikle robotik alanında kullanılmasının geliştirilen robot için hem donanım tasarlama gereksinimlerini büyük ölçüde kolaylaştırdığı hem de yazılım geliştirme aşamasında görüntü işleme, sinyal işleme gibi karışık işlemlerin geliştirilme süresini çok büyük ölçüde kısalttığı gözlemlenmiştir.

10 SONUÇ VE ÖNERİLER

Geçtiğimiz yıllarda, profesyonel ve amatör robotlar için hem maliyet anlamında hem de her bir robot için değişik yazılım dilleri öğrenme anlamında gereken ihtiyaçlar ciddi ölçüde bir yükseliş göstermişti. Bu ihtiyaçlar aynı zamanda robot alanında yeni olan kişilerin bu alana girişini oldukça zorlaştırmaktaydı. Bu anlamda MSRS, ister profesyonel isterse amatör anlamda olsun robot dünyasına giriş için standart bir platform sağlamıştır.

MSRS, Windows tabanlı, servis mantığı ile çalışabilen kullanıcı araçları ile kullanılabilen gelişmiş bir programlama tekniğidir. MSRS tarafından büyük kolaylıklar sağlayan bu platform, basitten karmaşığa birçok robotu destekler. Yapılacak olan işe veya iş için ayrılabilecek bütçeye göre bu robotlardan istenilenin seçilmesine imkân sunar. Yine MSRS tarafından sunulan Web tabanlı işlem takibi ile eşzamansız (asenkron) işlemleri yönetmek için kullanılan World Wide Web (www) teknolojisini içermesinden dolayı network ve internet uygulamalarına da büyük kolaylık ve avantajlar getirmiştir. MSRS'nin Uygulama Geliştirme Aracı (VPL) gibi yardımcı araçlar sayesinde de uygulamaları tasarım yüzeyine sürükle – bırak aktivitesiyle yapmayı sağlayarak programlama işlemini bariz bir şekilde kolaylaştırır.

MSRS'nin VPL programı bilinen klasik programlama yöntemlerinden oldukça farklı olan sürükle – bırak yöntemi ile bir robot uygulamasının programlama süresini kısaltmaktadır. Görsel bir programlama tekniği olduğu için herhangi bir programcı tarafından kolay bir şekilde anlaşılabilmekte ve programla ilgili hataların bulunabilmesini de ciddi ölçüde kolaylaştırmaktadır. MSRS, dünyaca ünlü birçok robot üreticisi tarafından desteklendiği için güçlü robot kontrol servislerine sahiptir.

VPL, yapılan uygulamayı bir servis olarak derleyerek, Visual C# kodlarına çevirebilmektedir. Bu özelliği sayesinde uygulamaları kontrol eden servis kodlarına müdahale etme şansı vererek her programcının kendine özgü uygulama servisleri yapabilmesini sağlamaktadır. Bu özellik aynı zamanda tamamen programcının tasarımı olan yeni servisler oluşturulmasını da sağlamaktadır.

VPL'nin hata ayılama (debug) özelliği sayesinde de çok kompleks robot uygulama programlarındaki hataların bulunmasını çok kolaylaştırmaktadır. Bu

özelliği ile programın o andaki durumunu, programda kullanılan bütün değişkenlerin değerlerini anlık olarak gösterebilmektedir. Programın istenilen yerinde programın duraklatılmasını sağlayabilmektedir. Bu sayede adım adım giderek tüm program detaylı bir şekilde incelenebilmektedir.

MSRS'nin üstün özelliklerinden biriside, sıfır maliyet ile istenilen bir robot üzerinde uygulama yapmayı sağlayan bir simülasyon (VSE) ortamı sunmaktadır. Bu özelliği sayesinde, grup çalışması yapan tasarımcıları aynı robotu satın almadan, VSE ortamında maliyet ve zaman kayıpları olmadan test edip geliştirebilirler. VSE'ye, MSRS'nin desteklemediği bir robotu Visual Stüdyo'yu kullanarak MSRS'nin desteklediği robotlar şeklinde simülasyon yapılabilir.

Bu tez çalışması kapsamında Microsoft Robotics Studio ile yapılan robotik alanındaki çalışmalar araştırılmış ve Türkçe kaynağa rastlanmamıştır. Bu çalışma ile Microsoft Robotics Studio kullanılarak yapılacak robotik alanındaki çalışmalarda kullanılmak üzere Türkçe bir kaynak oluşturulması amaçlanmıştır. Tez çalışması boyunca Sara Morgan'ın (2008) "Programming Microsoft® Robotics Studio" çalışmasından büyük ölçüde faydalanılmıştır.

11 KAYNAKLAR

Morgan, S. 2008. Programming Microsoft® Robotics Studio, 1st ed. Microsoft Press, USA.

Kosaka A. and Kak A. C. (1992) "Fast Vision – Guided Mobile Robot Navigation Using Model-Based Reasoning and Prediction of Uncertainties." Computer Vision, Graphics, and Image Processing – Image Understanding 56(3): 271-329.

Meng M. and Kak A. C. (1993) "Mobile Robot Navigation Using Neural Networks and Nonmetrical Environment Models." IEEE Control Systems 30-39, Oct.

Meng M. and Kak A. C. (1993). "NEURO-NAV: A Neural Network Based Architecture for Vision-Guided Mobile Robot Navigation Using Non-Metrical Models of the Environment." Proc. IEEE Int'l Conf. Robotics and Automation 2: 750-757.

Santos-Victor J., Sandini G., Curotto F., and Garibaldi S. (1993). "Divergent Stereo for Robot Navigation: Learning from Bees." Proc. IEEE CS Conf. Computer Vision and Pattern Recognition.

Pan J., Pack D. J., Kosaka A., and Kak A. C. (1995). "FUZZY-NAV: A Vision-Based Robot Navigation Architecture Using Fuzzy Inference for Uncertainty-Reasoning." Proc. IEEE World Congress NeuralNetworks 2: 602-607, July.

Kim D. and Nevatia R. (1995). "Symbolic Navigation with a Generic Map." Proc. IEEE Workshop Vision for Robots 136-145, Aug.

Dev A., Kröse B., and Groen F. (1997). "Navigation of a Mobile Robot on the Temporal Development of the Optic Flow." Proc. IEEE Int'l Conf. Intelligent Robots and Systems 558-563, Sept.

Guilherme N. D. and Kak A. C. (2002). “Vision for Mobile Robot Navigation.” IEEE Transactions On Pattern Analysis And Machine Intelligence 24(2): February.

Parhi D. R. (2005). "Navigation of mobile robots using a fuzzy logic controller." Journal of Intelligent & Robotic Systems 42(3): 253-273.

Ning, K. and Yang R. Q. (2006). "MAS based embedded control system design method and a robot development paradigm." Mechatronics 16(6): 309-321.

Albus, J., Bostelman R. et al. (2006). "Learning in a hierarchical control system: 4D/RCS in the DARPA LAGR program." Journal of Field Robotics 23(11-12): 975-1003.

Kulyukin, V., Gharpure C. et al. (2006). "Robot-assisted wayfinding for the visually impaired in structured indoor environments." Autonomous Robots 21(1): 29-41.

Hayet J. B., Lerasle F., Devy M.(2007). “A visual landmark framework for mobile robot navigation.” Science Direct.

Guzman J. L., Berenguela M., Rodrigueza F., Dormidob S.(2007). “An interactive tool for mobile robot motion planning.” Science Direct Robotics and Autonomous Systems.

Wanga X., Houa Z. G., Zoua A., Tana M., Cheng L. (2007). “Vision Based İnterception of a moving target with a nonholonomic mobile robot.” Science Direct.

Freda L., Oriolo G.(2007). “Vision Based İnterception of a moving target with a nonholonomic mobile robot.” Science Direct Robotics and Autonomous Systems 55: 419-432.

Squire, K. M. and Levinson S. E.(2007). "HMM-based concept learning for a mobile robot." Ieee Transactions on Evolutionary Computation 11(2): 199-212.

Ma, Z., Hu Y. et al. (2007). "A novel design of in pipe robot for inner surface inspection of large size pipes." Mechanics Based Design of Structures and Machines 35(4): 447-465.

Kim, T. H., Choi S. H. et al. (2007). "Incorporation of a software robot and a mobile robot using a middle layer." Ieee Transactions on Systems Man and Cybernetics Part C-Applications and Reviews 37(6): 1342-1348.

Esposito J. M., and Kim M.(2007). "Using formal modeling with an automated analysis tool to design and parametrically analyze a multirobot coordination protocol: A case study." Ieee Transactions on Systems Man and Cybernetics Part a-Systems and Humans 37(3): 285-297.

Clark, J. and Fierro R. (2007). "Mobile robotic sensors for perimeter detection and tracking." Isa Transactions 46(1): 3-13.

Chen, H. Y., Matsumoto K. et al. (2007). "Self-calibration of environmental camera for mobile robot navigation." Robotics and Autonomous Systems 55(3): 177-190.

http://tr.wikipedia.org/wiki/G%C3%B6m%C3%BCl%C3%BC_sistem

http://mindstorms.lego.com/eng/Overview/default.aspx