İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

DONANIM İÇEREN SİMÜLASYON SİSTEMLERİ VE SİMÜLATÖR TASARIMI

YÜKSEK LİSANS TEZİ

Müh. Murat DEMİRCİ

Anabilim Dalı : DİSİPLİNLERARASI PROGRAMLAR

Programı : MEKATRONİK MÜHENDİSLİĞİ

HAZİRAN 2006

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

DONANIM İÇEREN SİMÜLASYON SİSTEMLERİ VE SİMÜLATÖR TASARIMI

YÜKSEK LİSANS TEZİ
Müh. Murat DEMİRCİ
(518031042)

Tezin Enstitüye Verildiği Tarih : 08 Mayıs 2006
Tezin Savunulduğu Tarih : 08 Haziran 2006

Tez Danışmanı : Prof. Dr. Levent GÜVENÇ

Diğer Jüri Üyeleri Yrd. Doç. Dr. Gökhan İNALHAN (İTÜ)

Yrd. Doç. Dr. Ümit SÖNMEZ (İTÜ)

HAZİRAN 2006

ÖNSÖZ

Günümüzde artık hemen hemen hiçbir endüstriyel sistemi sadece mekanik boyutuyla ele almak mümkün değildir. Mekanik sistemlerin içerisine gelişen dijital elektronik teknolojisi ile birlikte elektronik ve bilgisayar sistemleri ilave edildikçe, daha kapsamlı ve daha karmaşık sistemler ortaya çıktı. Mekatronik sistemler olarak adlandırılan bu sistemlerdeki gömülü kontrol sistemlerinin artan sayısı ile birlikte, bu kontrol sistemlerinin tasarım, geliştirme ve testleri ayrı bir önem kazanmıştır. Bu çalışmada, gömülü kontrol sistemlerinin geliştirilmesinde kullanılan simülasyon teknolojileri ve yöntemleri araştırılmış ve gerçek zamanlı helikopter ve yol taşıtı simülatörü tasarlanmıştır.

İTÜ Otomotiv Kontrolü ve Mekatroniği Araştırmaları Grubu (AUTOCOM - MEKAR) bünyesinde çalışma ve araştırma yapma fırsatı tanıyan ve tecrübeleri ve bilgisi ile bana bir çok yardımları dokunmuş ve sağlam bir mekatronik vizyonu kazandırmış değerli ve saygıdeğer koordinatörüm, danışmanım ve hocam sayın Prof.Dr. Levent GÜVENÇ'e sonsuz teşekkürlerimi sunuyorum.

Bunun yanında desteklerinden ötürü Yrd.Doç.Dr. Bilin AKSUN GÜVENÇ'e, İTÜ MEKAR, İTÜ AUTOCOM ve İTÜ ROTAM çalışanlarına tek tek teşekkür ederim. Hayatımın her anında ve her konuda sevgi ve desteklerini esirgemeyen ve her zaman yanımda olan aileme de çok teşekkür eder, saygı ve sevgilerimi sunarım. Ayrıca çok değerli dostlarıma ve arkadaşlarıma da desteklerinden ötürü teşekkür ederim.

Mayıs 2006 Murat DEMİRCİ

İÇİNDEKİLER

KISALTMALAR

ABS	: Anti-lock Braking System
ESP	: Elektronic Stability Program
ECU	: Electronic Control Unit
CAN	: Controller Area Network
TCS	: Traction Control System
HILS	: Hardware-In-The-Loop Simulation
RCP	: Rapid Controller Prototyping
MVEM	: Mean Value Engine Model
SBW	: Steer-By-Wire
MILS	: Man-In-The-Loop Simulation
UDP	: User Datagram Protocol
TCP / IP	: Transmission Control Protocol / Internet Protocol
PC	: Personal Computer
PWM	: Pulse Width Modulation

ŞEKİL LİSTESİ

Sayfa No

DONANIM İÇEREN SİMÜLASYON SİSTEMLERİ VE SİMÜLATÖR TASARIMI

ÖZET

Bu tezde, otomotivden havacılık ve uzay teknolojilerine, robotikten savunma teknolojilerine kadar birçok sektörde yaygın olarak kullanılmakta olan hızlı kontrolcü prototiplendirme teknikleri ve araçları, donanım içeren simülasyonlar ve donanım içeren simülasyon sistemleri, gerçek zaman ve gerçek zamanlı simülasyonlar ve gerçek zamanlı simülasyon sistemleri araştırılmış ve örnek uygulamalar ile bu simülasyon teknolojileri hakkında bilgi verilmiştir. Ayrıca helikopter kontrol sistemlerinin tasarımı ve testleri için bir gerçek zamanlı helikopter simülatörü ve otomotiv kontrol sistemlerinin tasarımı ve geliştirilmesi için de bir yol taşıtı simülatörü geliştirilmiştir.

Birinci bölümde, tez konusunda motivasyon için bir giriş yapılmış, mekatronik sistemler tanıtılmış, değişik sektörlerdeki mekatronik sistemlerden örnekler verilmiş ve mekatronik sistemlerde kontrolcü tasarımından bahsedilmiştir.

İkinci bölümde, öncelikle hızlı kontrolcü prototiplendirmenin ne demek olduğu, nasıl yapıldığı ve yöntemleri hakkında bilgi verilmiş ve örnek uygulamalar ile hızlı kontrolcü prototiplendirme tekniği anlatılmıştır. Ayrıca bu bölümde tez kapsamında tasarlanmış olan bir adet otomobil için savrulma engelleyici kontrol ve Vario model helikopter için de bir adet savrulma ekseni stabilite artırıcı kontrolcü tasarlanmıştır.

Üçüncü bölümde, gerçek zaman kavramının ne anlama geldiği anlatıldıktan sonra neden gerçek zamana ve gerçek zamanlı simülasyonlara ihtiyaç duyulduğundan bahsedilmiş ve gerçek zamanlı simülasyon sistemleri hakkında bilgi verilmiştir.

Dördüncü bölümde, donanım içeren simülasyonlar hakkında bilgi verilmiş, gömülü kontrol sistemlerinin geliştirilmesi sürecindeki önemi üzerinde durulmuş, nasıl yapıldığından bahsedilmiş ve örnek uygulamalar ile donanım içeren simülasyon sistemleri tanıtılmıştır.

Beşinci bölümde, uçuş kontrol sistemlerinin geliştirilmesi ve test edilmesi amacıyla geliştirilen düşük maliyetli gerçek zamanlı pilotlu helikopter simülatörü anlatılmış ve simülatör bileşenleri hakkında bilgi verilmiştir.

Altıncı bölümde ise, otomotiv kontrol sistemlerinin saha testlerine çıkmaksızın laboratuar ortamında geliştirilmesi ve test edilmesi amacıyla İTÜ MEKAR bünyesinde geliştirilmiş gerçek zamanlı yol taşıtı simülatörü, bileşenleri ve simülatörde kullanılan teknolojiler ile birlikte tanıtılımıştır.

Yedinci ve son bölümde ise sonuçlar verilmiş ve gelecekte yapılabilecek çalışmalar hakkında yorumlarda bulunulmuştur.

HARDWARE-IN-THE-LOOP SIMULATION SYSTEMS AND SIMULATOR DESIGN

SUMMARY

In this thesis, Rapid Controller Prototyping techniques and tools, Hardware-In-The-Loop Simulations (HILS) and HILS systems, Real- Time and Real-Time Simulation systems which are commonly used in the automotive, aerospace and aeronautics, robotics and defense sectors are examined. These simulation technologies are introduced with sample applications. In addition, a helicopter simulator for designing and testing of helicopter flight control systems and a road vehicle simulator for designing and testing of automotive control systems are developed in this thesis.

In the first chapter, an introduction about the subject of the thesis is presented for motivation. Some mechatronics systems in different sectors are presented and controller design in mechatronics systems are mentioned.

In the second chapter, meaning of Rapid Controller Prototyping (RCP) is explained along with rapid controller prototyping methods and tools. Rapid controller prototyping is explained in detail using sample applications. Also in this chapter, an automotive yaw stability controller and a model helicopter attitude controller are treated.

In the third chapter, meaning of real-time simulations and the need for them are mentioned. Some of the most commonly used real-time simulation systems are introduced.

In the fourth chapter, Hardware-In-The-Loop Simulations (HILS) are introduced. Their importance in the development of embedded control systems is explained in detail using sample applications.

In the fifth chapter, a low cost, real-time, piloted helicopter simulator which is designed for development and testing of flight control systems is introduced. The general structure of the simulator is explained.

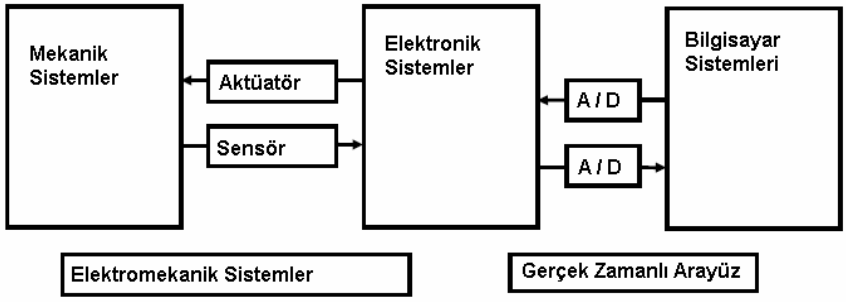
In the sixth chapter, a real-time road vehicle simulator which is designed in MEKAR for development and testing of automotive control systems in a laboratuary environment are introduced.

The seventh and last chapter of the thesis includes comments and recommendations for future work.

1. GİRİŞ

1.1. Giriş ve Literatür Taraması

1950'lerde tranzistörün keşfi elektronik dünyasında bir devrim yaratarak yeni bir çağın başlangıcı olmuştu. Ardından gelişen mikroişlemci ve mikrodenetleyici teknolojileri de endüstriyel sistemlerdeki ağırlığını giderek arttırdı. Günümüzde artık endüstriyel sistemleri sadece mekanik boyutuyla ele almak mümkün değildir. Mekanik sistemlerin içerisine gelişen dijital elektronik teknolojisiyle birlikte elektronik ve bilgisayar sistemleri ilave edildikçe, daha kapsamlı ve daha karmaşık sistemler ortaya çıktı. Bu gelişimin sonucunda 'mekatronik' kavramı doğdu ve artık bu sistemler 'mekatronik sistemler' olarak adlandırılmaktadır. Şekil 1.1'de mekatronik sistemlerin genel yapısı verilmiştir.



Şekil 1.1 : Mekatronik sistemlerin yapısı [1].

Örneğin günlük hayatımızda sıkça kullandığımız otomobiller bir zamanlar tümüyle mekanik olmasına rağmen artık komple bir mekatronik sistem halini almıştır. ABS fren sistemi, TCS çekiş kontrol sistemi, Steer-by-Wire direksyon sistemi, Brake-by-Wire fren sistemi, ESP güvenlik sistemi, hava yastıkları, seyrüsefer sistemleri, vb. bileşenler otomobillerde standart olmaya başladı. Benzer şekilde yakıt ve emisyon optimizasyonu için hibrit elektrikli araçlar ve hatta sadece elektrikli araçların üretilmesi, otomobillerde elektronik sistemlerin oranını giderek arttırmıştır. Hatta bundan sonraki geleceğin otomobillerindeki yeniliklerin ve gelişmelerin daha ziyade dijital teknoloji ekseninde olacağını söylemek ve böyle bir öngörüde bulunmak yanlış olmaz.

Uçak ve uzay teknolojilerinde de oldukça büyük teknolojik ilerlemeler sağlanmıştır. Bir zamanlar uçaklarda ve helikopterlerde olduğu gibi itki çubuklarının birbirlerini itmesi veya kablo mekanizmaları ile sağlanan kontrol yüzeylerinin kontrolü artık yerini yavaş yavaş X-by-Wire sistemlerine bırakmaktadır. Tasarlanan otopilot ve seyrüsefer sistemleri uçuş güvenliğini üst düzey seviyelere çıkartmış ve pilotaj işlevlerini otomatik hale getirmiştir. İnsansız hava araçları da geliştirilmiş ve gözetleme ve keşif amaçlı olarak günümüzde kullanılmaktadır. Ayrıca Ay'a ilk ayak basıldığı 1960'lı yıllardan sonra uzay teknolojisi de büyük mesafeler katetmiştir. Gerek askeri ve gerekse sivil amaçlar doğrultusunda dünya yörüngesine yüzlerce uydu yerleştirilmiş, dünya dışındaki gezegenlerde hayat olup olmadığını araştırmak amacıyla uzaya uydular gönderilmiştir.

Robotların dünyasına bir gözatıldığında da benzer gelişmeler görülmektedir. Gelişen bilgisayar teknolojileri ile birlikte robot manipülatörlerinin kontrolünde büyük mesafeler katedilmiştir. Oldukça hassas konum kontrolü sağlanan robotlar montaj, kesme, kaynak, vb. yüksek hassasiyetli endüstriyel uygulamalarda yaygın olarak kullanılmaktadır. Gerek bir insan için risk oluşturan uygulamalardan dolayı ve gerekse maliyet açısından endüstride yavaş yavaş tam otomasyona geçilmektedir. Benzer şekilde doğadaki canlıların taklidi robotlar da artık günlük hayatımızda sıkça kullanılmaktadır. İnsan taklidi robotlar, su altı robotlar, yılan robotlar, uçan robotlar insanoğlunun hizmetine sunulmuştur. MEMS ve mikroorganik teknolojiler sayesinde insan vücudunun içerisinde bile dolaşabilecek, gerekli tedavileri orada yapacak boyutta robotların üretilmesi planlanmaktadır.

Gelecekteki savunma sisteminin otonom uçak, otonom helikopter, otonom tank, otonom insansı robotlar gibi tamamen otonom araçlardan oluşacak olması bu alandaki teknolojik gelişmeleri öngörmek açısından yerinde bir örnektir. Benzer şekilde füze teknolojisi de çok ilerlemiştir. Kilometelerce uzaklıkta dünyanın herhangi bir yerindeki bir noktayı santimetreler mertebesinde hassasiyetle vurmak mümkündür. Buna karşılık refleks olarak füze savunma sistemleri de üst düzey seviyelere ulaşmıştır.

Genel olarak yukarıda bahsedilen alanlardaki mekatronik sistemlerin bileşenlerinin hemen hemen hepsi kapalı çevrim içinde çalışmakta ve dolayısıyla bunlar için geniş kapsamlı kontrol sistemlerinin tasarlanması gerekmektedir. En alt seviye kontrol işlevi, sensör girişlerinin okunması, buna karşılık gerekli kontrol mekanizmasının çalıştırılması ve kontrol sinyalinin üretilmesi ve aktüatörlerin sürülmesi olarak ifade edilebilir. Bir mekatronik sistemde bu tarz işlevlerden yüzlerce bulunabilir ve bunların

bazısı veya hepsi birbirleriyle etkileşimli olabilir. Hatta bir füze savunma sisteminde, bir uçak veya helikopter otopilot sisteminde, bir ameliyat robotunda, otomobillerdeki güvenlik sistemlerinde olduğu gibi bu kontrol sistemlerinin gerçek zamanlı olarak çalışması gerekmektedir. Öyle ki, bir füze savunma sisteminde saldırı füzesini havada imha etmek için gönderilen füzenin koordinatlarını tayin eden kontrolcünün, saldırı füzesinin değişen koordinatlarına göre füzenin koordinatlarını çok hızlı ve tutarlı bir şekilde değiştirmesi, gerçek zamanlı olarak değişen durumlara karşılık cevap vermesi gerekmektedir. Bir otomobildeki aktif güvenlik sisteminde, sürücünün ani bir manevra yaptığı veya yapmak zorunda kaldığı, aracın kontrolünün kaybedildiği durumlarda, sistemdeki kontrolcünün aracın bu devrilme ya da savrulma eğilimini önceden sezmesi, buna uygun kontrol ve karar mekanizmalarını işletmesi ve gerekli aktüatörleri sürmesi ve sürücünün güvenliğini sağlaması gerekmektedir. Bütün bunları gerçek zamanlı olarak yapması gerekmektedir. Bir uçuş kontrol sistemindeki otopilot sisteminde görev yapan kontrolcünün, uçağın değişken durumlarını izlemesi, buna göre gerektiğinde kontrol mekanizmalarını çalıştırarak kontrol sinyallerini üretmesi ve uçağı dengede tutması gerekmektedir. Yine bütün bu işlemlerin gerçek zamanlı olarak yapılması gerekmektedir. Bu gibi sistemlerde en ufak gecikme veya hata, telafisi mümkün olmayan sonuçlar doğurabilir. Dolayısıyla bu kontrol sistemlerinin tutarlı ve olabilecek her türlü senaryolara karşılık test edilmiş olması gerekmektedir.

Mekatronik sistemler için kontrolcü tasarlamanın bir çok evresi vardır. Öncelikle kontrolcünün tasarlandığı sistemin iyi tanımlanması ve mümkünse sensör ve aktüatör dinamiklerini de içerecek şekilde bir matematiksel modelinin geliştirilmesi gerekmektedir. Geliştirilen bu matematiksel model baz alınarak sistemin davranışı incelenir, kararlılık analizleri yapılır ve sistemi kararlı hale getirecek ve belli bir misyonu gerçekleyecek kontrol sistemi tasarlanır.

Bilgisayar ortamında geliştirilen bu kontrolcüler maliyet, zaman gibi nedenlerden dolayı doğrudan gerçek sistem üzerinde test edilemeyebilir. Örneğin bir uçak veya helikopter için geliştirilen kontrol sistemi düşünüldüğünde kontrolcü geliştirme sürecinin her evresinin doğrudan sistem üzerinde test edilmesi hiç de mantıklı olmaz. Yanlış tasarlanan bir kontrol işlevi aracın kontrolden çıkmasına ve düşmesine neden olabilir. Bu gibi durumlarda donanım içeren simülasyon sistemleri devreye girmektedir. “Donanım İçeren Simülasyon” terimi, İngilizce’de “hardware-in-the-loop simulation” teriminin Türkçe karşılığı olarak kullanılmaktadır. Her ne kadar

Türkçe literatürde “Çevrimde Donanım Simülasyonu” şeklinde kullanıldığı görülse de, bu tezde “Donanım İçeren Simülasyon” terimi kullanılacaktır.

Donanım içeren simülasyonlarda tasarlanan kontrolcü, birlikte çalışacağı sistem bileşenleri ile gerçek zamanlı olarak çalıştırılır, kontrolcünün davranışı buna göre incelenir ve buna göre parametreleri güncellenir. Böyle bir simülasyonda sistemdeki bileşenlerin bazısının matematiksel modelleri yerine gerçek sistem bileşenleri kapalı çevrim kontrol sistemine dahil edilir. Genel olarak iki farklı yapı kullanılır: 1) Kontrol edilen sistemin direkt kendisi ve kontrolcünün de sanal benzetimi kontrol çevrimine dahil edilebilir, 2) Tasarlanan kontrolcünün gerçekte çalıştırılacağı sistem veya elektronik kontrol ünitesi ve kontrolcünün tasarlandığı sistemin sanal matematiksel modelinden oluşabilir.

Ayrıca donanım içeren simülasyonların gerçekleştirilebilmesi için, otomobiller için donanım içeren gerçek zamanlı yol taşıtı simülatörleri, uçaklar ve helikopterler için donanım içeren gerçek zamanlı uçuş simülatörleri, robotlar için donanım içeren gerçek zamanlı robot simülatörleri geliştirilmiştir. Tasarlanan kontrolcüler öncelikle bu simülatörler üzerinde test edilmekte, kontrolcünün asıl çalışacağı ortamda maruz kalacağı durumlar, bu durumların sanal benzetimleri oluşturularak denenmekte ve ondan sonra son protiplendirme aşamasına geçilmektedir.

Kontrol sistemi tasarımını bir de maliyet ve zaman açısından ele almak gerekir. Projenin sahibi, koordinatörü veya patron en kısa zamanda mümkün mertebe en düşük maliyetli kontrol sisteminin tasarlanmasını ister. En kısa zamanda sonuçlanmasını isteyecektir, çünkü pazardaki rakiplerinden önce ürünü piyasa sürmek ve piyasada ilk olmak isteyecektir. Tasarım maliyetlerinin kontrol edilmesi gerekir, çünkü şirket veya kurum ayakta kalmak için kar elde etmeli ve mali dengeleri gözetmek zorundadır. Bütün bu tasarım kriterleri, kontrol sistemlerinin sistematik bir şekilde tasarlanması, geliştirilmesi ve prototiplendirilmesini gerektirmektedir. Bunun sonucunda hızlı kontrolcü prototiplendirme kavramı ortaya çıkmış, hızlı kontrolcü prototiplendirme araçları geliştirilmiş ve bu sayede gerek kontrol sisteminin maliyeti gerekse kontrol sisteminin tasarım evrelerinde harcanan zaman bakımından oldukça büyük kazanımlar elde edilmiştir.

Günümüze bakıldığında donanım içeren simülasyon sistemleri ve hızlı kontrolcü prototiplendirme teknikleri endüstride yaygın olarak kullanılmaktadır. Hatta otomotivden uçak endüstrisine bir çok alanda araştıma geliştirme sürecinde standart bir uygulama haline gelmiştir.

Literatürde donanım içeren simülasyon sistemleri, simülatörler ve hızlı kontrolcü prototiplendirme teknikleri ile ilgili bir çok çalışma yapılmıştır. Yoon vd. [2] otomotivde motor kontrol sistemleri için MATLAB/Simulink ve RTLAB temelli gerçek zamanlı donanım içeren simülasyon platformu geliştirmiş ve bir çok motor kontrol uygulaması için kullanılabilcek bu platform üzerinde motor için EGR kontrol, AFR kontrol, injection zamanlama, diyagnostik testler vb. uygulamalar gerçekleştirmiştir. Zanella vd. [3] otomotivde mekatronik sistemler için hızlı kontrolcü prototiplendirme teknikleri ve donanım içeren simülasyonlar hakkında çalışmış ve bu teknikleri X-mobile olarak adlandırılan otonom bir araç üzerinde uygulamalı olarak tanıtmıştır. Anakwa vd. [4] gömülü kontrol sistemlerinin hızlı prototiplendirilmesi için MATRIXx tabanlı bir platform geliştirmiştir. Wu vd. [5] güç elektroniği uygulamalarında dijital kontrol sistemlerinin hızlı prototiplendirilmesi ve gerçek zamanlı donanım içeren simülasyon sistemleri için bir platform geliştirmiştir. Papini ve Baracos [6] uçak teknolojileri alanında hızlı kontrolcü prototiplendirme ve donanım içeren simülasyonlar üzerinde çalışmış ve RTLAB ve MATRIXx tabanlı bir dağıtık uçuş simülatörü geliştirmişlerdir. Toyota'dan Kimura ve Maeda [7] otomotivde motor kontrol sistemlerinin geliştirilmesi için MATLAB/Simulink kullanarak hızlı kontrolcü protiplendirme ortamı geliştirmiş ve gerçek zamanlı bir simülatör tasarlamıştır. Jang vd. [8] otomotivde direksiyon kontrol sistemlerinin donanım içeren simülasyonlar ile geliştirilmesi üzerine çalışmalar yapmıştır. Hanselmann [9,10] özellikle otomotiv mekatroniğinde kontrol uygulamaları alanında hızlı kontrolcü prototiplendirme teknikleri ve donanım içeren simülasyonlar üzerinde çalışmış ve otomotive yeni açılımlar getirmiştir. Ptak ve Foundy [11] uzay araçları için tasarlanan kontrol sistemlerinin tasarımı ve testleri için RT-Linux ve MATRIXx tabanlı gerçek zamanlı donanım içeren bir simülasyon ortamı geliştirmişlerdir. Ford'dan Raman ve diğerleri [12] otomotivde araç aktarma organı kontrolcülerinin performans testleri için donanım içeren simülasyon sistemi geliştirmişlerdir. Lee ve Suh [13] ABS fren ve TCS çekiş kontrol sistemlerinin tasarlanması için donanım içeren simülatör geliştirmişlerdir. Temeltaş vd. [14] robot manipülatörleri için donanım içeren simülasyon ortamı geliştirmişlerdir. Oh [15] hibrit elektrikli araçların motor karakteristiklerinin donanım içeren simülasyonlarda test edilmesi konusunda bir çalışma yapmıştır. Wagner ve Keane [16] otomotiv şasi kontrol sistemlerinin tasarlanması üzerine çalışmalar yapmış ve gerçek zamanlı donanım içeren simülasyon testleri için araç setleri geliştirmişlerdir. Jeon vd. [17] donanım içeren simülasyon sistemleri kullanılarak uçaklar için ABS fren sisteminin tasarlanması üzerine bir çalışma yapmışlardır. Heo vd. [18] otomotivde elektrohidrolik fren sistemlerinin tasarımı ve kontrolü için donanım içeren simülatör geliştirmişlerdir.

Hagiwara vd. [19] donanım içeren simülasyon ve hızlı kontrolcü prototiplendirme teknikleri kullanılarak otomatik transmisyon kontrolcüsünün geliştirilmesi üzerine bir çalışma yapmışlardır.

1.2. Kapsam

Bu tezde ilk olarak hızlı kontrolcü prototiplendirme ve teknikleri hakkında bilgi verilecek, ardından gerçek zaman kavramı ve gerçek zamanlı simülasyonlardan bahsedilecek ve daha sonra donanım içeren simülasyonlara değinilecektir. Son iki bölümde de bu tez çalışması kapsamında tasarlanmış uçuş simülatörü ve yol taşıtı simülatörü hakkında bilgi verilecektir.

2. HIZLI KONTROLCÜ PROTOTİPLENDİRME

Otomotivden uçak ve uzay endüstrisine kadar bir çok sektörde yeni bir ürünün geliştirilmesi süreci hem maliyet, hem de zaman açısından büyük önem taşımaktadır. Bu süreç, otomotiv sektöründe yeni bir motor kontrol sisteminin veya şasi kontrol sistemlerinin geliştirilmesi veya benzer şekilde uçaklar için motor kontrol ve otopilot sistemlerinin geliştirilmesi örneklerinde olduğu gibi yeni bir kontrol algoritmasının tasarımı olabilir. Günümüzde bu kontrol algoritmalarının tasarım, geliştirme ve prototiplendirme evreleri için bir çok sistematik tasarım yöntemi, tekniği ve araçlar geliştirilmiş ve "Hızlı Kontrolcü Prototiplendirme" kavramı da bu süreçte çıkmıştır.

Daha önceleri kontrol sistemini geliştiren mühendis C, Fortran ve MATLAB gibi programlama dillerini kullanarak geliştirdiği kontrol algoritmasını kodlardı. Yine benzer programlama dillerini kullanarak geliştirdiği kontrol edilen sistemin matematiksel modeli ile birlikte kontrol algoritmasının simülasyonlarını yapardı. Kontrol mühendisi kontrol sistemini tasarladıktan ve simülasyonlar ile doğruluğunu test ettikten sonra, kontrol algoritmasının çalıştırılacağı mikrokontrolör ve DSP için assembly dilinde kod yazabilen bir yazılımcı devreye girer, kontrol mühendisinin tasarladığı algoritmayı yeniden kodlardı [20]. Bu da gösteriyor ki eskiden kontrol mühendisinin geliştirdiği algoritmanın simülasyonlarını yapabilmesi için C veya Fortran gibi yüksek seviye dillerden birini bilmesi gerekirdi. Hatta mikrokontrolörler için yüksek seviye dillerde programlar da yazılmasına imkan sağlayan derleyiciler ortaya çıkmadan önce assembly kod yazma becerisine sahip bir yazılım uzmanına ihtiyaç duyulmakta idi. Hatta daha da ötesi, sensör ve aktüatörlerin bağlanması, ara devrelerin tasarlanması gibi kontrol algoritmasının kontrol edilen sistem ile entegrasyonu esnasında bir elektronik uzamanına da ihtiyaç vardı.

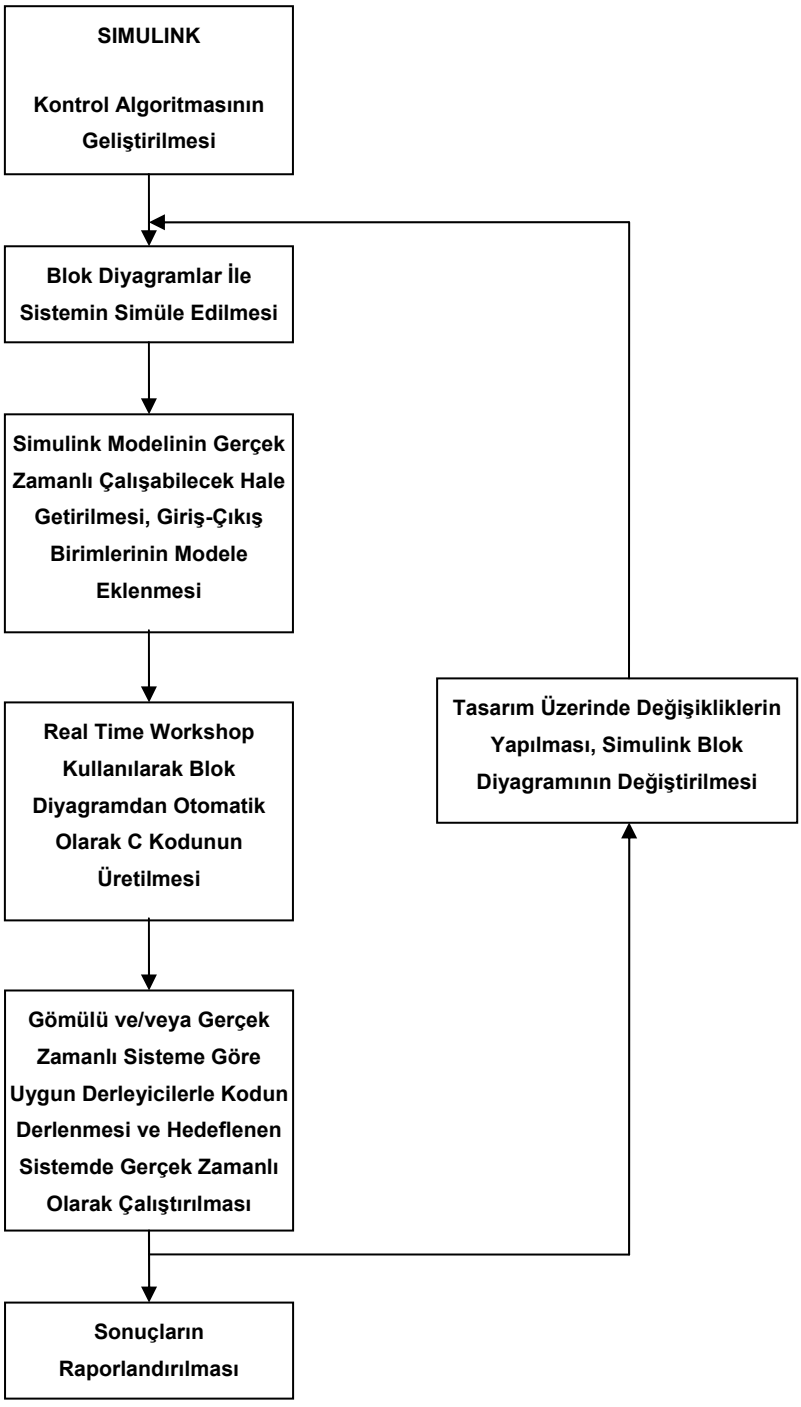
Günümüzde ise bu kontrol algoritmalarının prototiplendirilmesinde teknolojik gelişmelerin etkisi görülmektedir. Kontrol edilen sistem ve kontrolcü, model tabanlı görsel arayüzler üzerinde blok diyagramlar kullanılarak sürükle bırak yöntemiyle modellenmekte ve simülasyonları yapılmaktadır. İstendiğinde bu görsel modellerden otomatik olarak C kodu üretilebilmekte ve hatta kontrolcünün prototiplendirileceği

mikrokontrolör veya DSP'ye uygun derleyiciler aracılığıyla otomatik olarak derlenmekte ve yine otomatik olarak kontrol/kumanda kartına yüklenmektedir. Bütün bunlar model tabanlı simülasyonun yapıldığı ortamda bir tıklama ile saniyeler içerisinde gerçekleştirilebilmektedir.

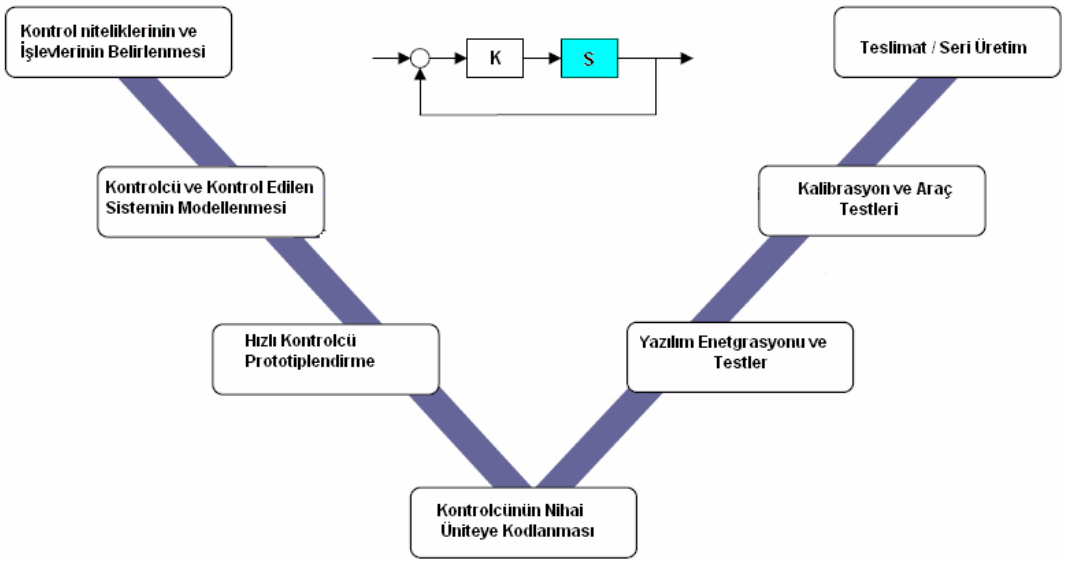
Hızlı kontrolcü prototiplendirme kapsamında bir çok araç geliştirilmiştir. Bunların çoğu MATLAB/Simulink, MATRIXx, LabView, Dymola, vb simülasyon programlarının üzerine inşa edilmiştir. MATLAB programı, Simulink ve bir çok mühendislik disiplini için geliştirilen araç setleri sayesinde model tabanlı simülasyona imkan vermektedir. Real Time Workshop araç seti kullanılarak, Simulink ortamında geliştirilen modelden otomatik olarak C kodu üretilebilmektedir. Ayrıca gerçek zamanlı ve gömülü sistem arayüzleri sayesinde üretilen bu kodlar, kontrol sisteminin çalıştırılması hedeflenen mikrokontrolör veya DSP üzerine otomatik olarak aktarılabilmektedir. Real Time Windows Target, xPC Target, dSPACE RTI, Embedded Target For Motorola MPC555, RTLAB, Embedded Target For TI 6000 DSP Matlab/Simulink tabanlı hızlı kontrolcü prototiplendirmeye olanak veren gerçek zamanlı ve gömülü sistemlerden bazılarıdır. Şekil 2.1'de Simulink üzerinde yapılan hızlı kontrolcü prototiplendirme akış şeması verilmiştir.

Hızlı kontrolcü prototiplendirme teknikleri ve araçları sayesinde, kontrol sistemi tasarım sürecinde kontrol algoritmalarının kodlanması, sistemin yazılım ve donanım prototiplendirmelerinin nasıl yapılacağıyla ilgilenilmekten ziyade, kontrol algoritmasının tasarlanması üzerinde odaklanılacaktır. Hızlı kontrolcü prototiplendirme teknikleri sayesinde geliştirme süreci oldukça kısalacak ve maliyetler düşecektir. Bu nedenle daha sonra değinilecek olan donanım içeren simülasyon ve hızlı kontrolcü prototiplendirme, otomotiv endüstrisinden robotik teknolojisine, enerji ve güç elektroniği uygulamalarından uçak ve uzay teknolojilerine kadar bir çok alanda standart haline gelmektedir.

Elektronik kontrol üniteleri için kontrolcü geliştirmeye yönelik modern kontrolcü geliştirme süreci Şekil 2.2'de verilmiştir (bkz. [21]). Buna göre ilk olarak tasarlanacak kontrolcünün yapması gereken işlevler tespit edilir. İkinci adımda kontrolcünün ve sistemin bilgisayar simülasyonları yapılır. Daha sonra hızlı kontrolcü prototiplendirme teknikleri ve araçları ile kontrol yazılımları geliştirilir ve gömülü sistemlere prototiplendirilir. Donanım içeren simülasyon testleri ile kontrolcünün performansı incelenir ve yapması gereken işlevler gerçek donanımlar da simülasyonlara dahil edilerek test edilir. Son geliştirme sürecinde de kontrolcünün kalibrasyonu ve araç testleri gerçekleştirilir.



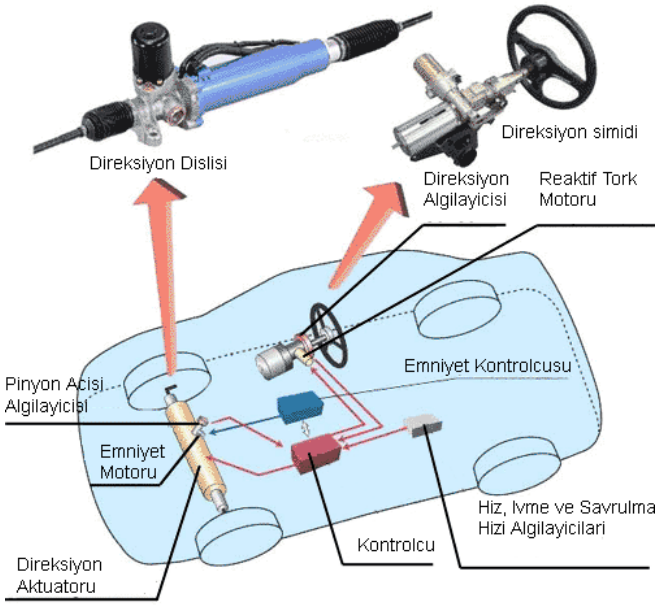
Şekil 2.1 : Simulink kullanarak hızlı kontrol prototiplendirme akış şeması.



Şekil 2.2 : Modern kontrolcü geliştirme sürecini gösteren V şeması [21].

2.1. Hızlı Kontrolcü Prototiplendirme Uygulaması - 1

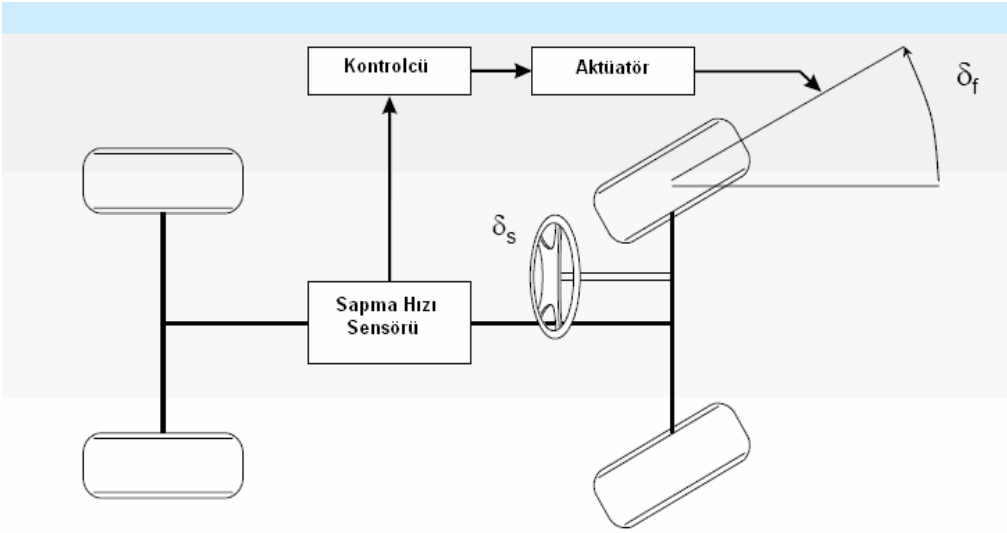
Birinci hızlı kontrolcü prototiplendirme uygulaması olarak, bir otomotiv mekatroniği uygulaması olan savrulma engelleyici kontrolcü tasarımı ele alınacaktır. Burada aracın Steer-by-Wire direksiyon sistemine sahip olduğu kabul edilmektedir. Şekil 2.3'de bu uygulamaya ilişkin blok diyagram verilmiştir.



Şekil 2.3 : Steer-by-Wire direksiyon ve savrulma engelleyici kontrolcü [22].

2.1.1. Kontrolcünün Nitelikleri ve İşlevleri

Aracın savrulma dinamiğine bağlı olarak tasarlanacak kontrolcünün, savrulmaya ters yönde bir moment oluşturacak şekilde direksiyon aktüatörüne sinyal göndermesi düşünülmektedir. Ayrıca kontrolcünün, araçtaki bu istenmeyen savrulma hareketini sürücünün panik tepki süresi içerisinde düzeltmesi, sürücü ve yolcuları rahatsız edecek düzeyde agresif olmaması, gerekli anlarda çok kısa süreli devreye girmesi ve kontrolü bütünüyle sürücüden devralmaması da istenmektedir [23].

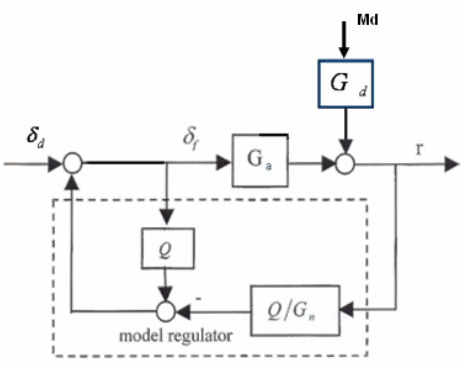


Şekil 2.4 : Savrulma engelleyici kontrolcü diyagramı.

2.1.2. Kontrolcü ve Kontrol Edilen Sistemin Modellenmesi

Tasarlanacak kontrolcünün özellikleri belirlendikten sonra kontrolcünün ve kontrolcü tasarlamak için gerekli olduğundan kontrol edilen sistemin modellenmesi aşamasına geçilir. Sözkonusu kontrolcü aracın yanal dinamiğini ilgilendirdiğinden, modellemenin basitliği açısından literatürde tek izli araç modeli olarak da adlandırılan bisiklet modelini kontrol edilen sistem, burada yol taşıtı, olarak kullanmak yeterlidir.

Savrulma stabilizasyonu kontrolcüsü olarak da Aksun Güvenç ve Güvenç'in [24] geliştirmiş oldukları model regülatörü kullanılacaktır. Şekil 2.5'de kontrolcünün yapısı verilmiştir.



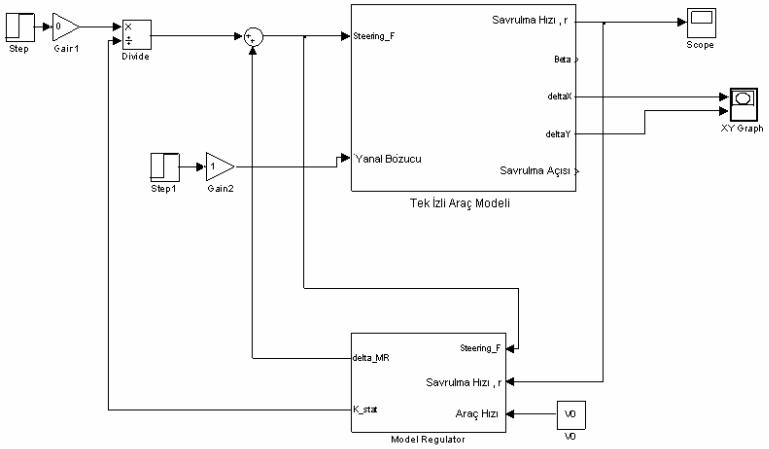
Şekil 2.5 : Savrulma stabilizasyonu kontrolcüsü [24].

Burada δ_d direksiyon girişi, G_n istenilen direksiyon girişi – savrulma hızı transfer foksiyonu, G_a direksiyon girişi – savrulma hızı transfer fonksiyonu, G_d bozucu etki transfer fonksiyonu, r savrulma hızı, M_d bozucu etki ve Q da (2.1)'deki gibi birinci dereceden ve kazancı boylamsal hıza göre ayarlanan bir filtredir.

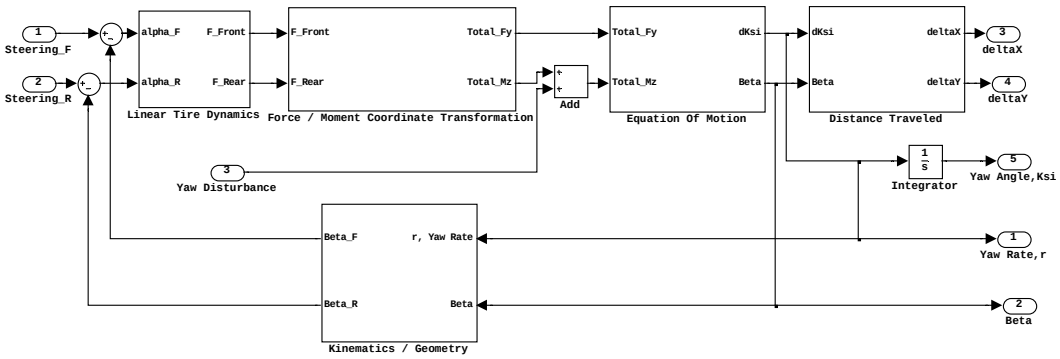
$$Q = \frac{1}{\tau_Q s + 1}, \qquad G_n = \frac{K_n(v)}{\tau_n s + 1} \tag{2.1}$$

2.1.3. Hızlı Kontrolcü Prototiplendirme

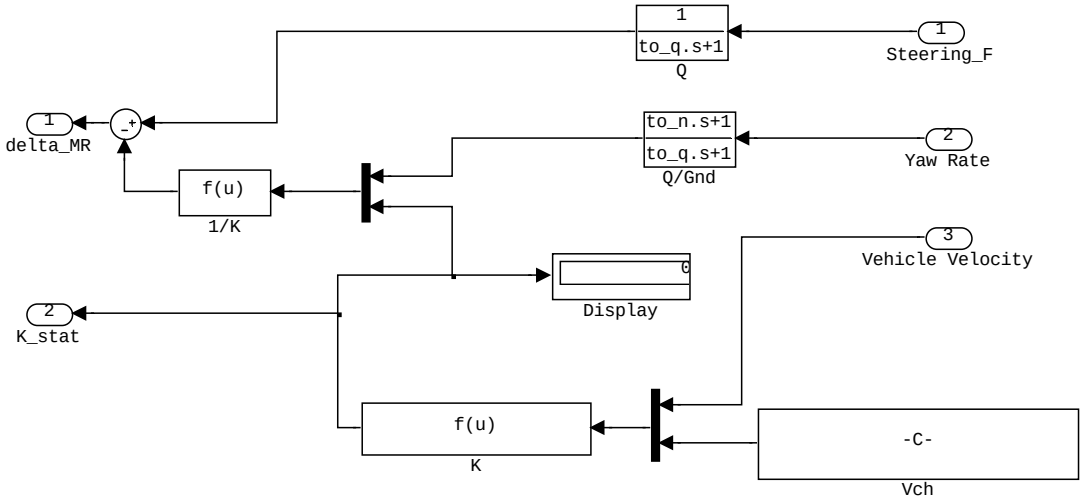
Burada hızlı kontrolcü prototiplendirme araçları olarak Matlab/Simulink ve dSPACE sistemleri kullanılacaktır. Kontrolcü ve kontrol edilen sistem, blok diyagramlar ve hazır bloklar yardımıyla Simulink ortamında kolay bir şekilde modellenebilir. Şekil 2.6, Şekil 2.7 ve Şekil 2.8'de kontrolcü ve kontrol edilen sistemin Simulink modeli verilmiştir.



Şekil 2.6 : Kontrolcü ve kontrol edilen sistemin Simulink modeli.

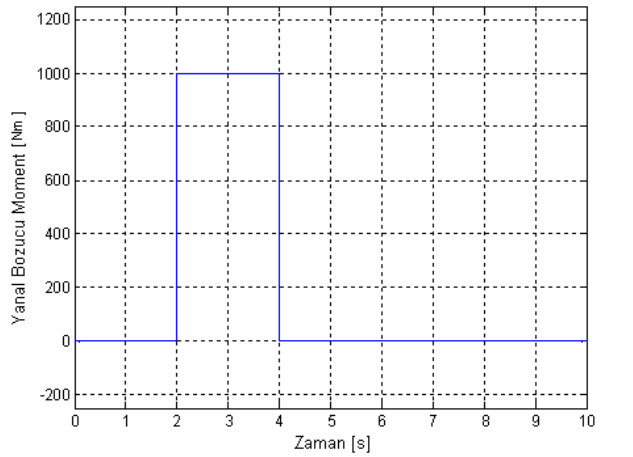


Şekil 2.7 : Tek İzli Araç modeli.

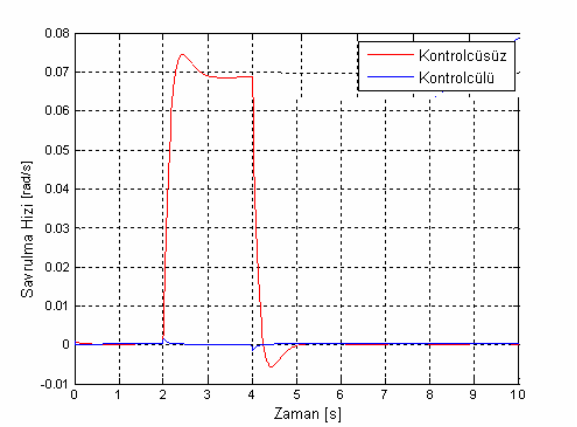


Şekil 2.8 : Model Regülatörü.

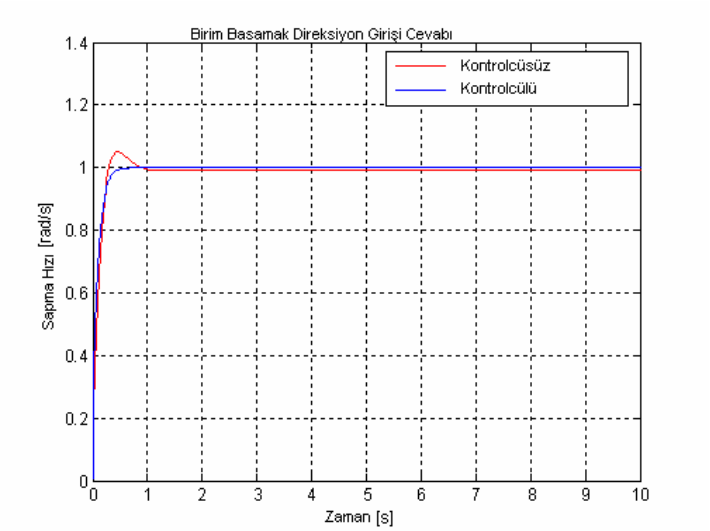
Simulink ortamında modeller kurulduktan sonra gerçek zamanlı olmayan simülasyonlar ile kontrolcünün performansı geliştirilir. Şekil 2.9 ve Şekil 2.10'da aracın yanal dinamiğindeki bozucu etkiye karşı sistemin kontrolcülü ve kontrolcüsüz durumdaki cevapları verilmiştir. Şekil 2.11'de de birim basamak direksiyon girişine karşılık gelen sistem cevabı verilmiştir. Şekil 2.9 ve 2.10'den de görüldüğü gibi kontrolcü aracın yanal dinamiğini iyileştirmekte ve tek izli araç modeli üzerinde başarılı bir şekilde çalışmaktadır. Bundan sonra aracın dinamiklerini çok daha iyi temsil edecek, aktüatör ve sensör dinamiklerinin dahil edildiği yüksek serbestlik dereceli bir araç modeli kurularak kontrolcünün performansı gerçek zamanlı olmayan simülasyonlar ile Simulink ortamında test edilir.



Şekil 2.9 : Simulink ortamında sisteme uygulanan bozucu etki, M_d.



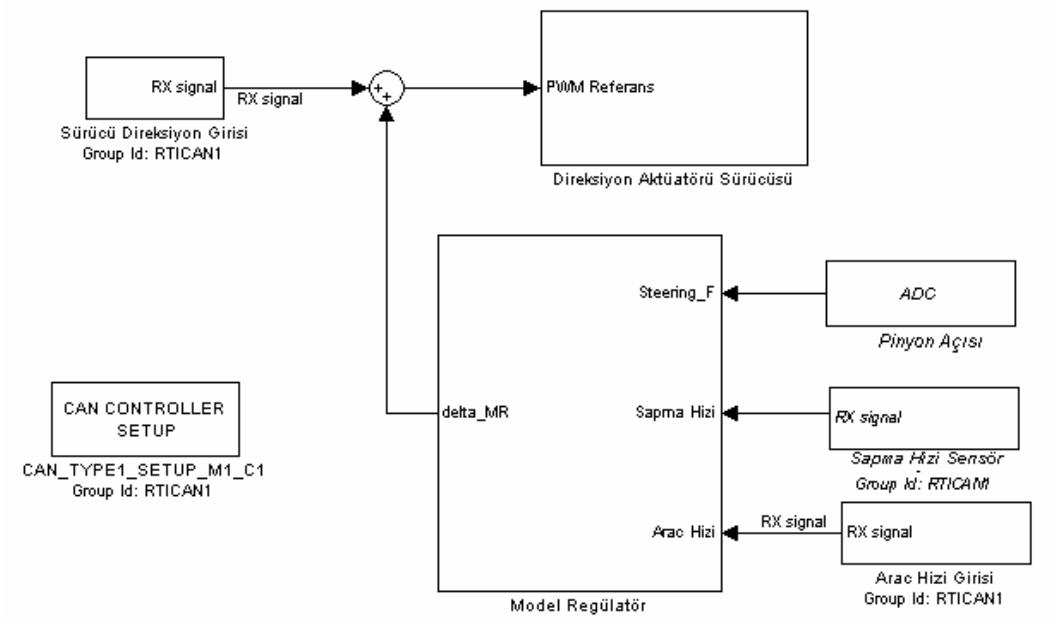
Şekil 2.10 : Bozucu etkiye karşı sistemin cevabı.



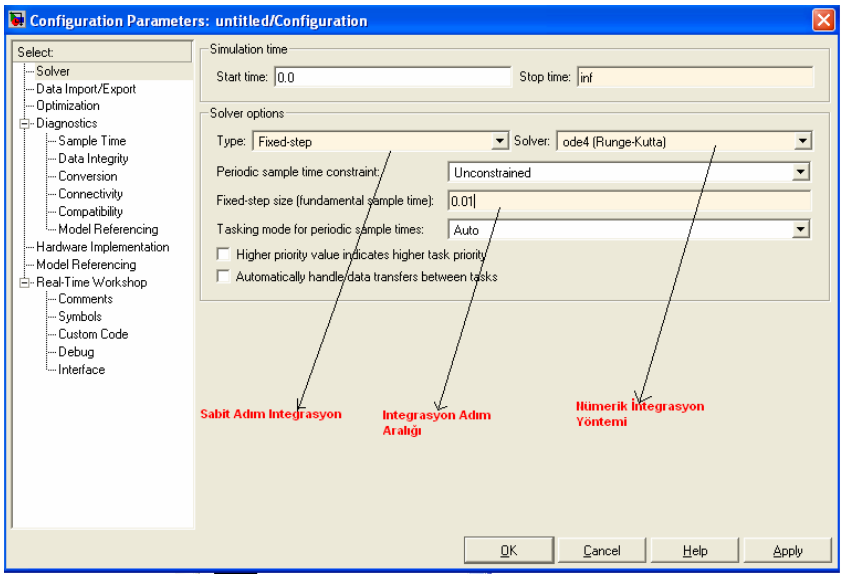
Şekil 2.11 : Sistemin birim basamak cevabı.

Gerçek zamanlı olmayan simülasyonlar ile kontrolcünün performansı test edildikten ve kontrolcünün iyi çalıştığı tespit edildikten sonra, bu kontrolcünün gerçek zamanlı

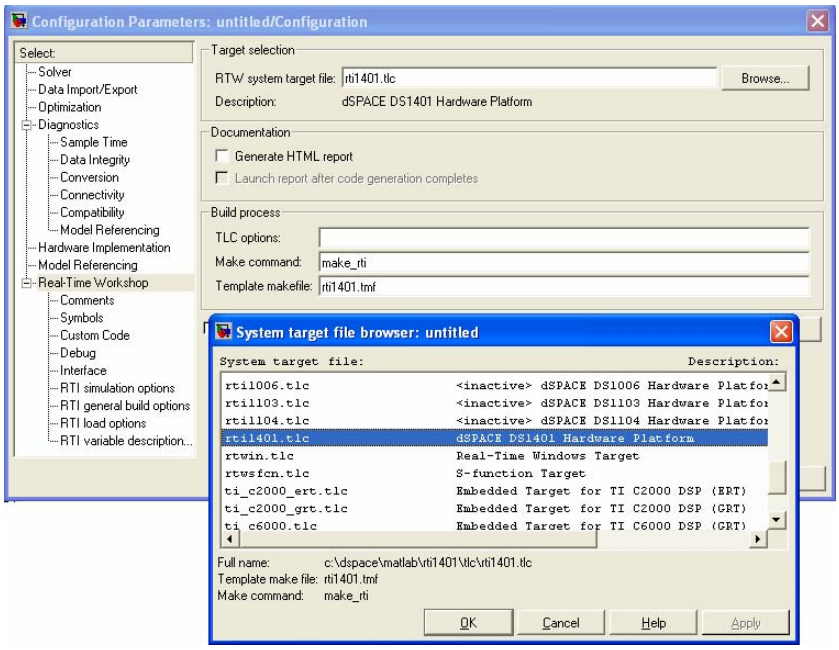
olarak çalıştırılacağı ortama aktarılması safhasına geçilir. Bu örnekte kontrolcünün gerçek zamanlı çalıştırılacağı ortam olarak dSPACE MicroAutoBox seçilmiştir. Bu safhada tasarlanan kontrolcü modeli üzerinde bir takım değişiklikler yapılır. Kontrolcü gerçek ortamda çalıştırıldığında, ihtiyaç duyacağı sinyalleri sanal sistem modeli yerine gerçek donanımdan temin edebilmesi için sensör girişlerinin ve üretilen kontrol sinyalleri ile aktüatörleri sürebilmesi için de aktüatör çıkışlarının kontrolcü modeline eklenmesi gerekir. Şekil 2.12'de bu amaçla değiştirilmiş kontrolcü modeli verilmiştir. Ayrıca geliştirilen modelin gömülü sistem üzerine prototiplendirilebilmesi için Şekil 2.13'de görüldüğü "Simulation → Configuration Parameters → Solver" menüsünden sabit adımlı integrasyon algoritması ve integrasyon adım aralığı seçilmelidir. Gerçek zamanlı gömülü sistem olarak dSPACE MicroAutoBox seçildiğinden "Real-Time Workshop → Browse" menüsünden de "DS1401 Hardware Platform" seçilmelidir.



Şekil 2.12 : Gerçek zamanlı çalıştırılmak üzere değiştirilmiş kontrolcü modeli.

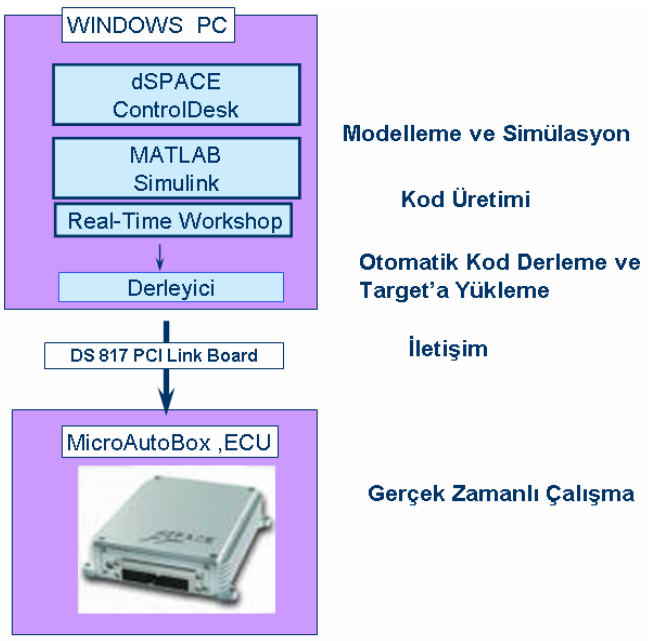


Şekil 2.13 : Sabit adımlı integrasyon ayarları.



Şekil 2.14 : Gerçek zamanlı sistem olarak DS1401 seçimi.

Real Time Workshop araç seti kullanılarak Simulink modelinden otomatik olarak C kodu üretilebilmektedir. Üretilen bu C kodu dSPACE tarafından desteklenen bir derleyici aracılığıyla derlenerek gerçek zamanlı olarak çalıştırılacağı MicroAutoBox üzerine yüklenir ve gerçek zamanlı performansı incelenir. Bu arada bütün bu kod üretimi, kodun derlenmesi ve yüklenmesi safhaları, eğer önceden gerekli ayarlamalar yapıldıysa, Simulink ortamında bir tıklama ile otomatik olarak yapılmaktadır. Şekil 2.15'de bu süreç akış diyagramı şeklinde gösterilmiştir.



Şekil 2.15 : Matlab/Simulink ve dSPACE ile hızlı kontrolcü prototiplendirme.

Hızlı kontrolcü prototiplendirme süreci tamamlandıktan ve kontrolcünün gerçek zamanlı sistem üzerindeki performansı incelendikten sonra kontrolcünün gerçekte çalıştırılacağı elektronik kontrolcü üzerine yüklenmesi safhası gelmektedir. Bazı tasarımcılar, kontrolcünün gerçekte çalıştırılacağı ortam için gerekli kodları elle yazma yolunu seçmektedirler. Bunun birinci nedeni, otomatik kod üreteçlerinin ürettikleri kodun boyutunun büyük olduğunu düşünmeleri ve daha optimum boyutta kod elde etmek istemeleridir. İkinci nedeni ise, hızlı kontrolcü prototiplendirme çalışmalarını yaptıkları platform ile kontrolcünün gerçekte çalışacağı platformun birbirinden farklı olması ve dolayısıyla kodun yeniden yazılması gerekliliğidir. Yalnız günümüzde optimum boyutta kod üretebilen otomatik üreteçler geliştirilmektedir. Ayrıca hızlı kontrolcü geliştirme sürecinde derleme aşamasından önce koda müdahele edilerek de üretilen kod üzerinde istenen değişikliklerin yapılması mümkündür. İkinci problem de, kontrolcünün gerçekte çalıştırılacağı elektronik kontrol ünitesine benzer bir hızlı kontrolcü prototiplendirme platformu kullanılarak çözülebilir. Zaten hızlı kontrolcü prototiplendirme sistemlerini geliştirenler bu problemleri dikkate almakta ve prototiplendirme araçlarını ona göre geliştirmektedirler. Örneğin, genelleme olmamakla birlikte, Bosch elektronik kontrol ünitelerinde sıkça Motorola işlemcilerinin kullanılması dikkate alınarak, dSPACE prototiplendirme platformlarında Motorola işlemcilerine, mesela MPC565, yer vermektedir. Bu sayede V şemasındaki seri üretim amaçlı kod geliştirme safhasını da hızlı kontrolcü prototiplendirme safhasına dahil etmek mümkün olmaktadır.

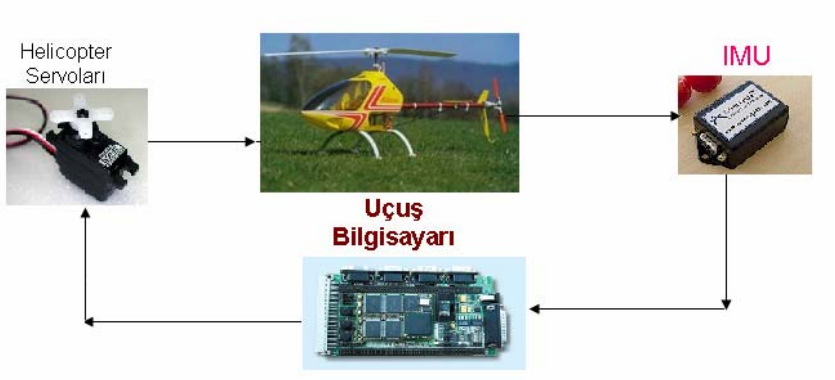
Kontrolcü geliştirme sürecinin bundan sonraki aşamalarında ise, Bölüm 4'te bahsedilecek olan donanım içeren simülasyon testlerinin yapılması, araç üzerinde testlerin yapılması yer almaktadır.

Görüldüğü gibi hızlı kontrolcü prototiplendirme sayesinde tasarım süreci otomatikleştirilmiş ve dolayısıyla geliştirme süresi oldukça kısalmıştır.

Tasarlanan bu savrulma engelleyici kontrolcünün Bölüm 6'da bahsedilecek olan Yol Taşıtı Simülatörü üzerinde donanım içeren simülasyon ile başarılı bir şekilde çalıştığı test edilmiştir.

2.2. Hızlı Kontrolcü Prototiplendirme Uygulaması – 2

Bu uygulamada da insansız hava aracı olarak model helikopter için yunuslama, yalpa ve sapma durumlarını kontrol edecek ve helikopterin stabilizasyonunu artıracak bir kontrolcünün tasarım evreleri anlatılacaktır. Şekil 2.16'da kapalı çevrim kontrol diyagramı verilmiştir.



Şekil 2.16 : Model helikopter durum kontrolcüsü.

2.2.1. Kontrolcünün Nitelikleri ve İşlevleri

Tasarlanacak kontrol sisteminin yapması gereken temel işlev, ataletsel ölçüm sisteminden (IMU) sapma, yalpa ve yunuslama açısal hız bilgilerini okuyarak helikopteri dengede tutacak kontrol sinyallerini üretmek ve ilgili helikopter servolarını sürmektir. Helikopter durumlarının gerçek zamanlı olarak takip edilmesi ve servoların da gerçek zamanlı olarak sürülmesi gerektiğinden, kontrolcünün prototiplendirileceği sistem gerçek zamanlı çalışan ve tutarlı bir sistem olması istenmektedir.

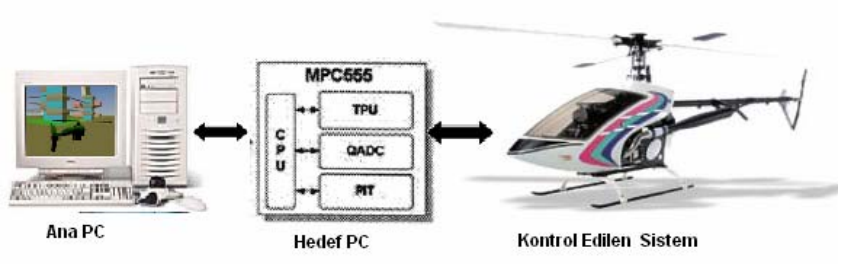
2.2.2. Kontrolcü ve Kontrol Edilen Sistemin Modellenmesi

İlk olarak kontrol edilen sistemin matematiksel modeli kurulur. Model helikopter modelinde temel olarak gövde, rotor, kuyruk rotoru, dikey stabilizatör, Bell stabilizatör, vb. bileşenlerin modelleri yer almaktadır. Model helikopter aktüatörleri ise, palaların hücum açısını değiştirerek ana rotor tarafından üretilen kaldırma miktarını kontrol eden kollektif servosu; helikopterin sağa – sola ve ileri – geri hareketini kontrol eden saykılik servoları ve helikopterin sapma açısını kontrol eden kuyruk servosu olmak üzere toplam dört tanedir.

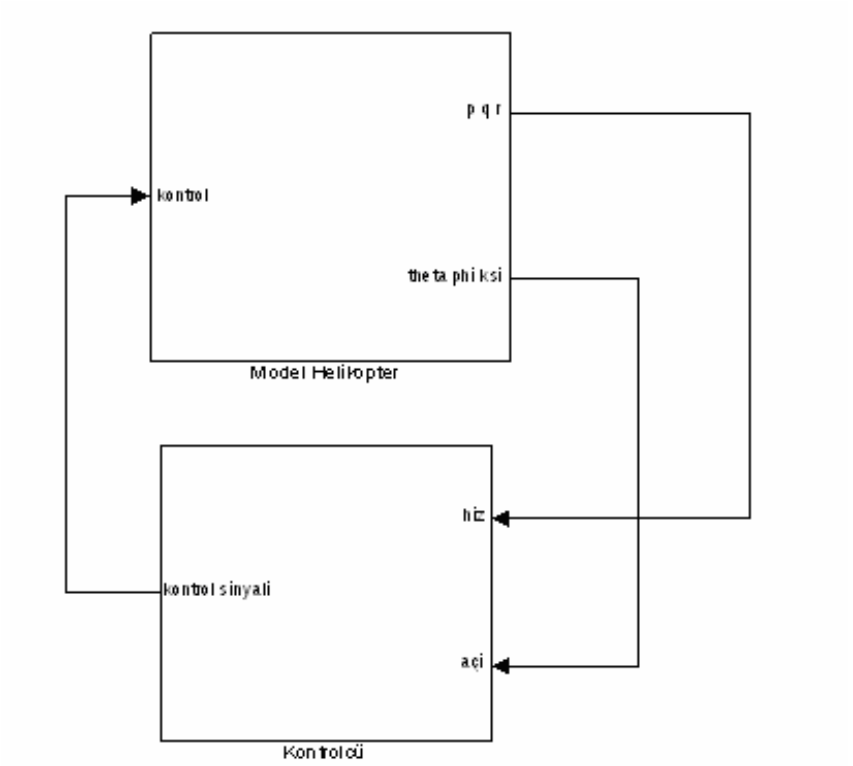
Gerçekte çok daha karmaşık ve çok girişli – çok çıkışlı kontrol sistemleri tasarlanmasına rağmen, helikopterin ilgili kontrol bileşenlerine göre eğrisel helikopter modelinden doğrusal modeller elde edilerek her bir model için doğrusal kontrolcüler de tasarlanabilir. Daha sonra bu doğrusal kontrolcülerin hepsi birlikte kullanılarak eğrisel helikopter modeli üzerinde simülasyonları yapılıp performansı incelenebilir.

2.2.3. Hızlı Kontrolcü Prototiplendirme

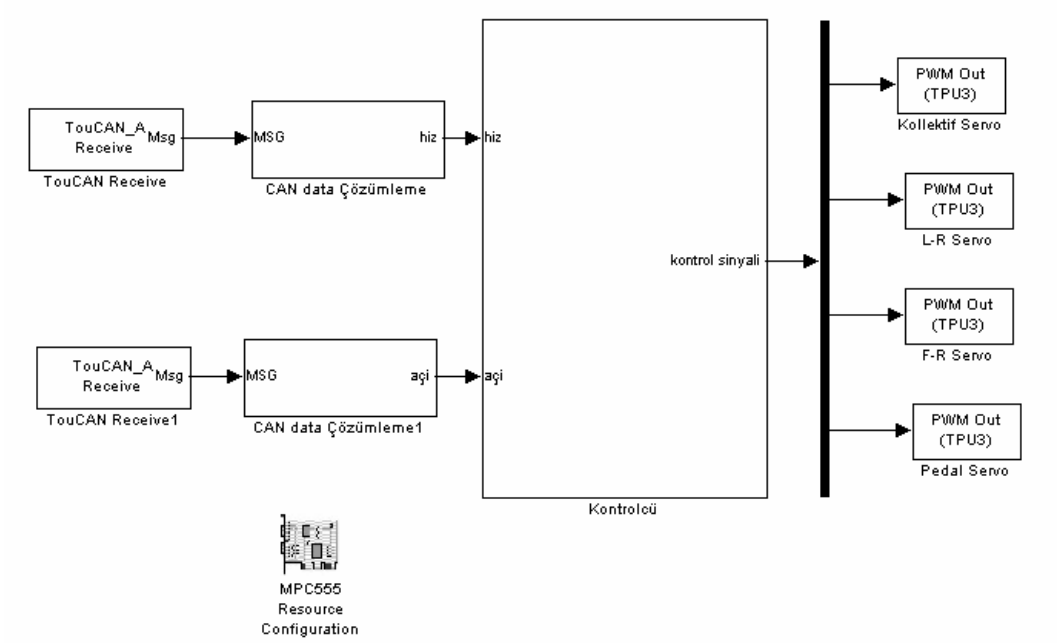
Bu örnekte hızlı kontrolcü prototiplendirme platformları olarak Matlab/Simulink ve Embedded Target For Motorola MPC555 seçilmiştir. MPC555 32-bit, çalışma frekansı 40MHz ve üzerinde CAN'den dijital - analog I/O'ya bir çok kontrol uygulamasında ihtiyaç duyulan bileşenleri barındırmaktadır ve bu örnekte de olduğu gibi bir uçuş kontrol bilgisayarı olarak kullanabilir. Şekil 2.17'de hızlı kontrolcü prototiplendirme donanımı gösterilmiştir. Buna göre Ana PC, üzerinde hızlı kontrolcü prototiplendirme araçlarının yer aldığı bilgisayar ve Hedef PC de gerçek zamanlı çalışan kontrol ünitesidir. Şekil 2.18'de de Simulink ortamındaki kontrolcü ve kontrol edilen sistem modelleri verilmiştir. Helikopter durum kontrolcüsünün tasarımında sapma, yalpa ve yunuslama açısal hız bilgileri kontrol girişi olarak kullanılmakta, bu değişkenlere göre model helikopter aktüatörleri için kontrol sinyalleri üretilmektedir.



Şekil 2.17 : Model helikopter hızlı kontrolcü prototiplendirme donanımsal diyagramı.



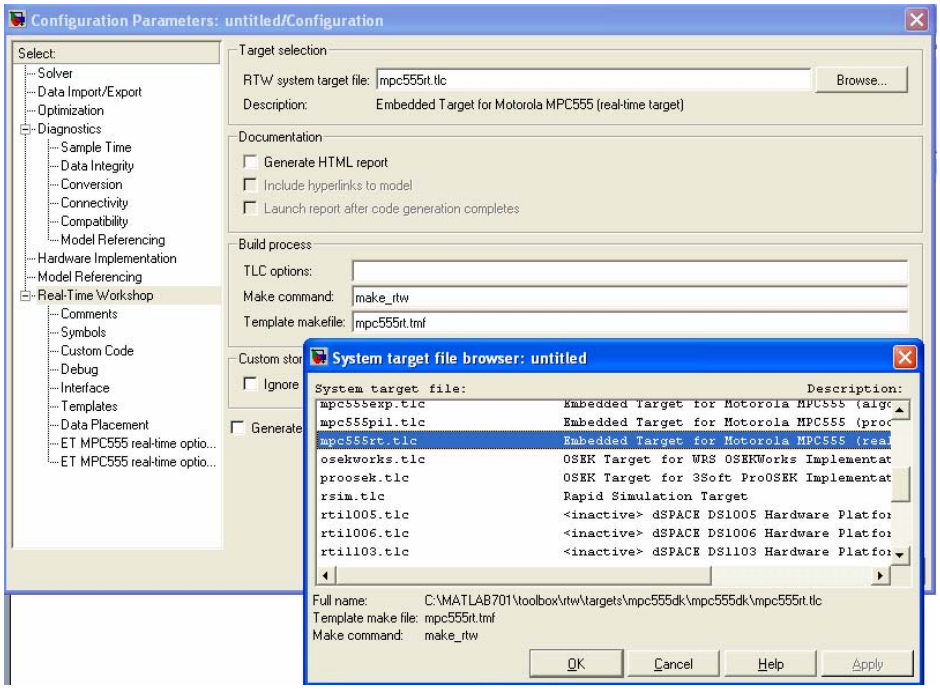
Şekil 2.18 : Kontrolcü ve kontrol edilen sistemin Simulink modeli.



Şekil 2.19 : MPC555 üzerinde gerçek zamanlı olarak çalışan model.

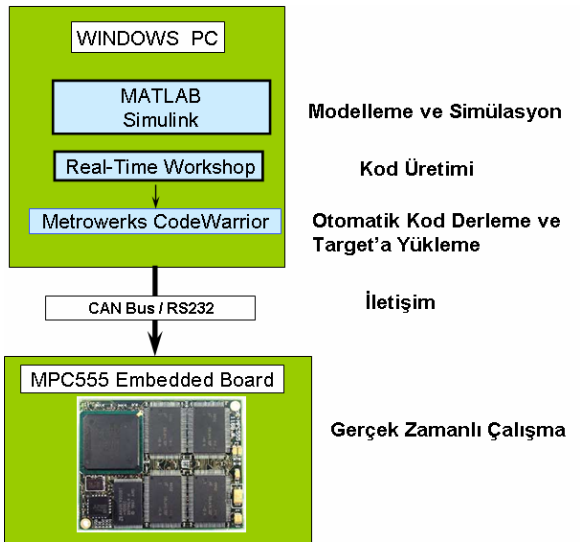
Gerçek zamanlı olmayan simülasyonlar ile kontrolcünün performansı test edildikten sonra Simulink modelinin gerçek zamanlı çalışabilecek hale getirilmesi gerekmektedir. Buna göre açısal hız bilgilerini IMU'dan okumak için CAN protokolü

bloklarını ve servoları sürmek için de dijital çıkış PWM modüllerini modele eklemek gerekmektedir. Ayrca Şekil 2.13'de gösterildiği gibi Simulink'de kullanılan çözücünün değiştirilmesi, sabit adımlı integrasyona müsade eden bir algoritmanın kullanılması ve simülasyon adımının belirlenmesi gerekir. Şekil 2.19'da gerçek zamanlı çalışabilir hale getirilmiş kontrolcü modeli verilmiştir.Stabilizasyon kontrolcüsünün gerçek zamanlı çalıştırılacağı gömülü sistem olarak Motorola MPC555 seçildiğinden dolayı Şekil 2.20'deki gibi "Configuration Parameters → Browse" menüsünden Embedded Target For Motorola MPC555 seçilir.



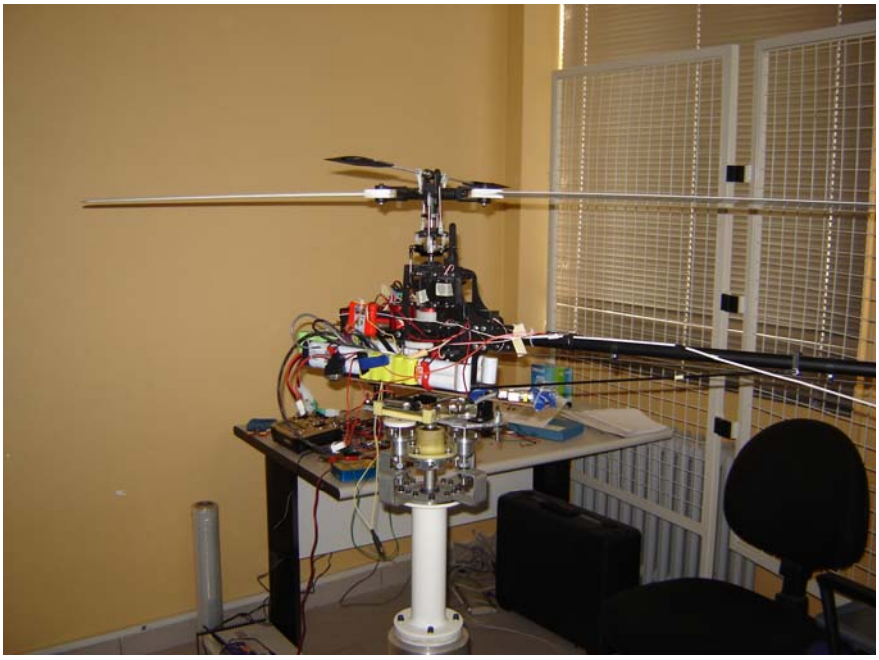
Şekil 2.20 : Gömülü sistem olarak Embedded Target For MPC555 seçimi.

Bundan sonraki aşama, Şekil 2.19'deki kontrolcünün kontrol ünitesine prototiplendirilmesidir. Ek 1'de Embedded Target For Motorola MPC55 için Matlab/Simulink ortamında yapılması gereken ayarlar belirtilmiştir. Real Time Workshop araç seti kullanılarak Simulink modelinden kontrolcünün kodu otomatik olarak üretilir. Daha sonra uygun bir derleyici , burada Metrowerks CodeWarrior, kullanılarak gerçek zamanlı olarak çalıştırılacağı elektronik kontrol ünitesine prototiplendirilir. Şekil 2.21'de model helikopter için hızlı kontrolcü prototiplendirme akış diyagramı verilmiştir.



Şekil 2.21 : Model helikopter hızlı kontrolcü prototiplendirme akış diyagramı.

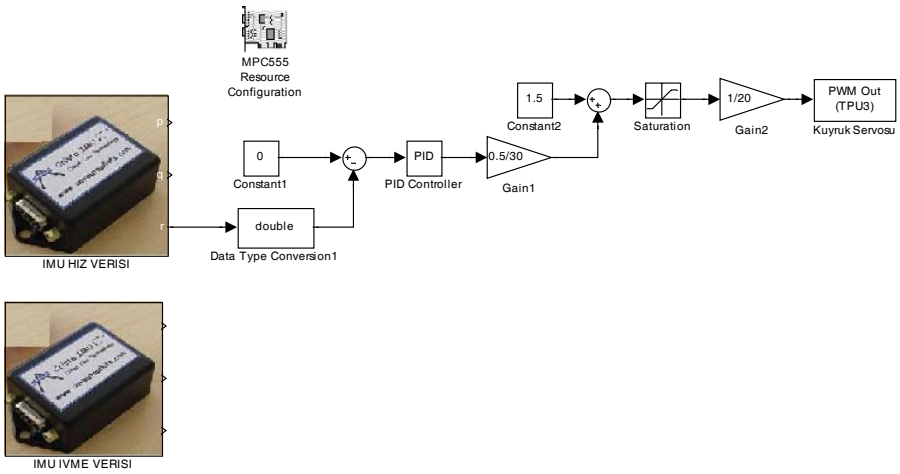
Model helikopter için tasarlanan kontrolcülerin laboratuar ortamında güvenli bir şekilde test edilebilmesi için İTÜ MEKAR laboratuarı bünyesinde İTÜ ROTAM'ın destekleri ile Şekil 2.22'deki platform geliştirilmiştir.



Şekil 2.22 : Model helikopter kontrolcüsü test platformu.

Bu platform yunuslama, yalpa, savrulma ve dikey yönde hareket ekseni olmak üzere toplam 4 serbestlik derecesine sahip bir platformdur. Helikopterin savrulma ekseninin kontrolü bu platform üzerinde başarılı bir şekilde test edilmiştir. Savrulma

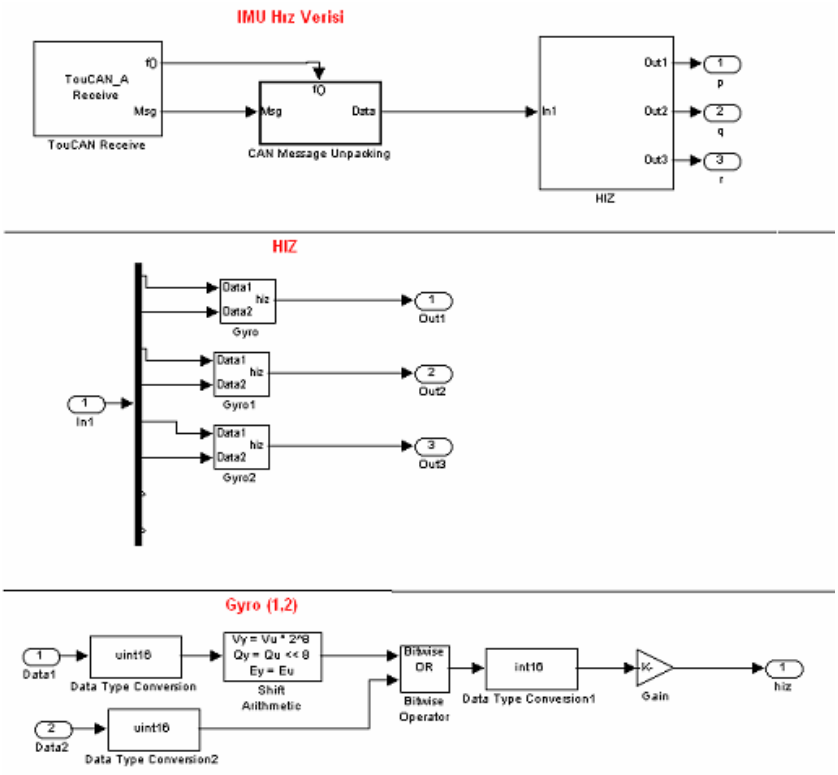
ekseni için uçuş bilgisayarı üzerinde çalıştırılan kontrolcüye ait Simulink modeli Şekil 2.23'de verilmiştir.



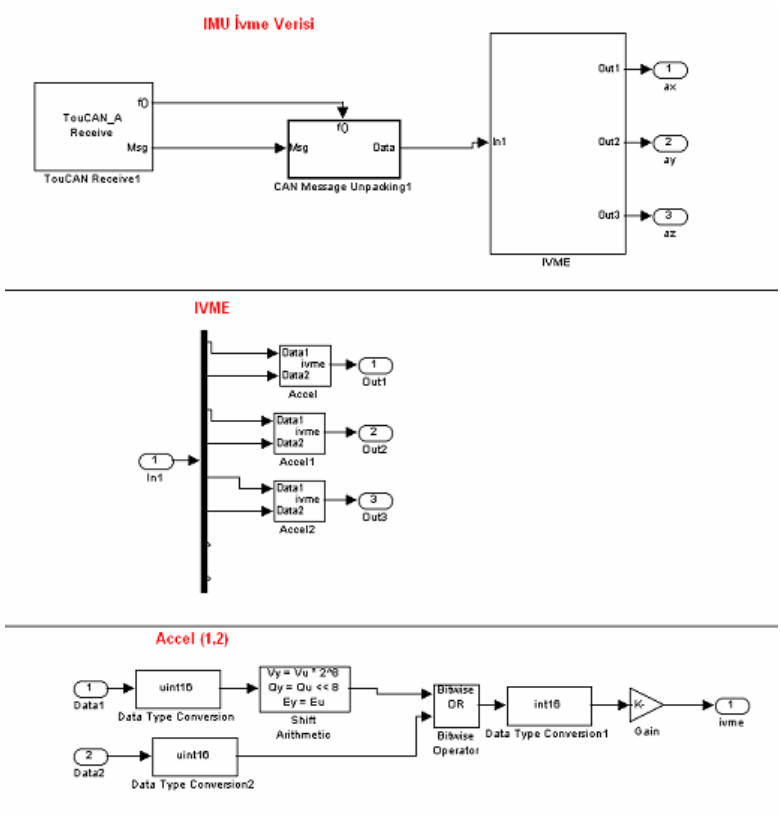
Şekil 2.23 : Vario model helikopter savrulma ekseni stabilite kontrolcüsü.

Helikopter üzerine yerleştirilen Cloud Captech firmasına ait IMU ile yalpa, yunuslama ve savrulma hızları ile X, Y, Z eksenlerindeki ivmeleri okumak mümkündür. Bu sensörden veriler hem RS232, hem de CAN protokolü aracılığıyla okunabilmektedir. Bu çalışmada IMU verileri CAN üzerinden okunmuştur. Uçuş bilgisayarı üzerinde IMU verilerini okumak için kullanılan Simulink modelleri Şekil 2.24 ve Şekil 2.25'de verilmiştir.

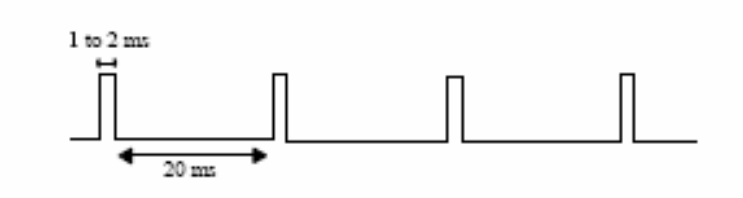
Kullanılan helikopter Vario firmasına ait elektrikli bir helikopterdir. Helikopter üzerinde Futaba firmasına ait S9452 kodlu helikopter servoları yer almaktadır. Helikopterin kontrolü bu servolar aracılığıyla sağlanmaktadır. Şekil 2.26'da gösterildiği gibi servolar periyodu 14 – 20 ms aralığında olması gereken PWM sinyalleri ile sürülmektedir. 0 – 90° arasındaki hareketi PWM sinyalinin darbe genişliği değiştirilerek kontrol edilmektedir. Buna göre 1 ms'lik darbe genişliği servonun minimum konumunu, 1.5 ms'lik darbe genişliği orta konumunu ve 2 ms'lik darbe genişliği de maksimum konumunu almasını sağlar. Gerekli PWM sinyali, uçuş kontrol bilgisayarı olarak kullanılan Şekil 2.27'deki Phytec firmasına ait Motorola MPC555 tabanlı gömülü sistem üretmektedir.



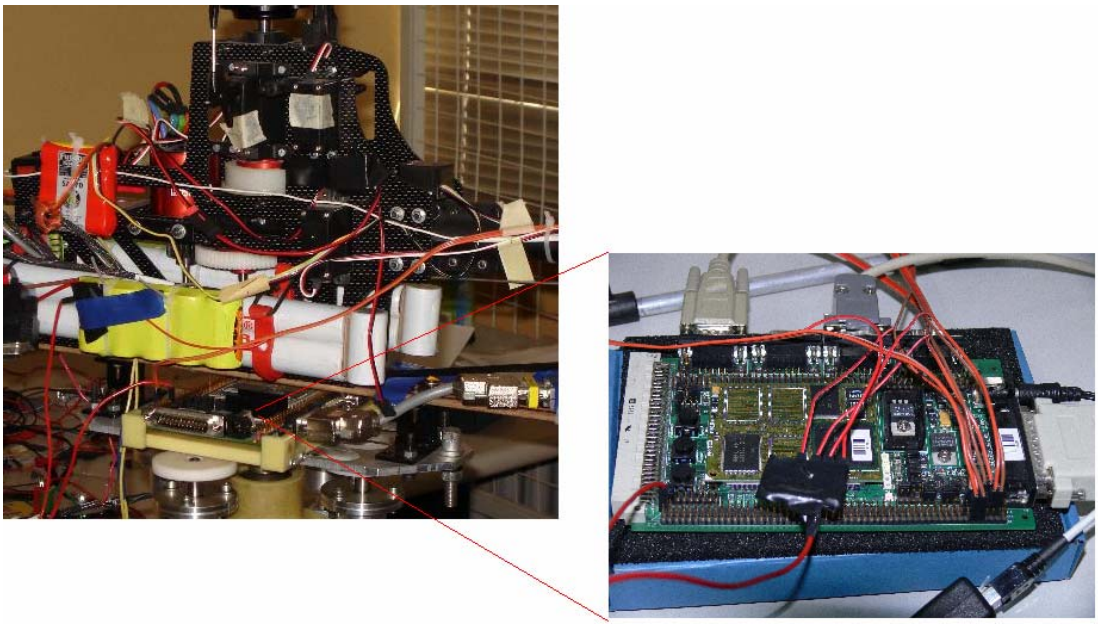
Şekil 2.24 : Helikopter açısal hızlarını CAN üzerinden okuyan Simulink modeli.



Şekil 2.25 : Helikopter ivmelerini CAN üzerinden okuyan Simulink modeli.



Şekil 2.26 : Futaba servolar için PWM kontrol sinyali.



Şekil 2.27 : Phytec firmasına ait MPC555 tabanlı gömülü sistem, uçuş bilgisayarı.

3. GERÇEK ZAMANLI SİMÜLASYON

Kontrolcü tasarım sürecinde, uygulamada gerçek zamanlı olarak çalıştırılacağından dolayı, tasarlanan kontrolcünün gerçek zamanlı performansının da incelenmesi gerekmektedir. Bunun için gerçek zamanlı simülasyonlara olanak veren platformlar, işletim sistemleri ve donanımlar kullanılmaktadır. Daha sonraki bölümde bahsedilecek olan donanım içeren simülasyonlarda olduğu gibi, tasarlanan kontrolcünün gerçekte kontrol edeceği sistem ile gerçek zamanlı simülasyonları yapılır. Örneğin bir uçuş simülatöründe pilot girişlerinden dolayı ve bir yol taşıtı simülatöründe de sürücü direksiyon girişlerinden dolayı simülasyonların gerçek zamanlı olarak çalışması gerekmektedir. Bunun için kontrolcünün, gerçek zamanlı çalışmaya olanak veren bir sistem üzerine veya gerçekte çalıştırılacağı elektronik kontrol ünitesine ya da mikrokontrolöre prototiplendirilmesi gerekmektedir.

3.1. Gerçek Zamanlı Sistemler ve Gerçek Zaman

Gerçek zamanlı sistem, belirli bir işlevi veya görevi önceden belirlenmiş bir sürede ve doğru bir şekilde gerçekleştirmesi gereken sistemlerdir. Bu işlev örneğin 1 milisaniye zaman aralıklarıyla sensör girişlerinin okunması, yorumlanması ve aktüatörlerin sürülmesi olabilir. Dolayısıyla gerçek zamanlı sistemler için bir zaman kısıtlaması söz konusudur.

Bu zaman kısıtlamasının durumuna göre gerçek zamanlı sistemler “katı gerçek zamanlı sistem” ve “esnek gerçek zamanlı sistem” olarak ikiye ayrılmaktadır. Gerçek zamanlı sistemler için, “katı” ve “esnek” sıfatları zaman sınırlarının tutturulması ile ilgilidir. Buna göre bir esnek gerçek zamanlı sistemde, önceden belirlenen zaman sınırlarının tutturulması istenir. Buna karşılık, katı gerçek zamanlı sistemlerde belirlenen zaman sınırlamalarının mutlaka tutturulması gerekir. Aksi durum istenmeyen olumsuz sonuçların doğmasına neden olabilir. Örneğin insan gözünün algılama hızından daha hızlı bir şekilde belirli zaman aralıklarıyla resim karelerini görüntülemesi istenen bir video oynatıcıyı düşünürsek, belirlenen sürenin aşılması halinde bu durum video görüntüsünün bozulması olarak algılanır ve izleyicinin hoşnutsuzluğuna neden olur. Yani gecikmenin doğuracağı ciddi bir tehlike

söz konusu değildir. Dolayısıyla esnek gerçek zamanlı bir sistemdir ve zaman gecikmelerine katlanılabilir. Diğer esnek gerçek zamanlı sistem örneği olarak sıklıkla kullandığımız ATM para çekme makinalarını düşünürsek, sistemin belirli zaman aralıklarında kullanıcının girişlerine cevap vermesi istenir. Haberleşme ağından veya dahili nedenlerden kaynaklanan gecikmelerin olmaması istenir ama katlanılabilir. Sadece ilgili bankanın sağladığı sistemin performansının düşük olması olarak algılanır ve müşteri memnuniyetsizliğine neden olur. Buna karşılık bir otopilot sistemi ele alındığında durum değişmektedir. Otopilotun önceden belirlenen sürelerde uçağın durumunu izlemesi, ilgili sensörleri okuması ve uçağı dengede tutacak kontrol sinyallerini üreterek ilgili aktüatörleri sürmesi gerekmektedir. Kontrol sistemindeki en ufak bir gecikme uçağın dengesini yitirmesine ve istenmeyen bir durumun oluşmasına neden olabilir. Otopilot sisteminin zaman sınırlamasına uyması ve gecikme söz konusu olmaması gerekir. Dolayısıyla katı gerçek zamanlı sistem olarak adlandırılır. Benzer şekilde bir Fly-By-Wire sisteminin, kullanıcının girişlerine hızlı bir şekilde cevap vermesi ve kontrol sinyallerini uçağın ilgili kontrol yüzeylerine iletmesi gerekmektedir. Sistemdeki gecikme pilotun kontrolü kaybetmesine, uçağın dengesini yitirmesine ve daha kötü istenmeyen durumlara neden olabilir. Dolayısıyla katı gerçek zamanlı bir sistemdir. Veya bir füze savunma sisteminde saldırı füzesini imha etmek için gönderilen füzenin koordinatlarının, saldırı füzesinin koordinatlarına göre güncellenmesi gerekmektedir. Sistemdeki bir gecikme füzenin sapmasına neden olabilir. Dolayısıyla bu sistem, katı gerçek zamanlı bir sistemdir.

Gerçek zamanlı sistemler daha ziyade ya bir gerçek zamanlı gömülü sistem ya da kullanıcı girişlerine cevap veren reaktif gerçek zamanlı sistem olarak karşımıza çıkmaktadır. Reaktif sistemlerden kastedilen, kullanıcı girişi ile aktifleşen sistemlerdir. Örneğin pilotun butona basması ile ateşlenen füze sistemi reaktif sistemdir. Gömülü sistemler ise, başlı başına bir bilgisayar olmayan daha genel bir sistem içinde belirli işlevi olan mikrokontrolör tabanlı sistemlerdir. Günümüzde otomobillerde, uçaklarda, helikopterlerde, uydularda, füzelerde ve daha bir çok sistemde gömülü sistemler yer almaktadır. Örneğin araç motor kontrol sistemi, ABS - ASR kontrol sistemi, direksiyon kontrol sistemi, araç güvenlik kontrol sistemleri, otopilot sistemleri birer gömülü sistemdir.

3.2. Gerçek Zamanlı Simülasyon Sistemleri

Bir kontrolcünün gerçek zamanlı simülasyonlarının gerçekleştirebilmesi için özel donanımlara ve yazılımlara ihtiyaç vardır. Özellikle katı gerçek zamanlı sistemler için

gerekli olan zaman sınırlamalarının tutturulması hızlı bilgisayar sistemlerini gerekli kılmaktadır. Günümüzde sıklıkla kullandığımız Linux ve MS Windows benzeri gerçek zamanlı olmayan işletim sistemleri üzerinde bu gerçek zaman taleplerinin karşılanması her zaman için mümkün olmamaktadır. Örneğin MS Windows işletim sistemini ele alırsak; aynı anda birden çok işlevi yerine getirecek şekilde tasarlanmış bir işletim sistemi olduğundan dolayı, tasarlanan kontrolcünün çalıştırılması işlevini askıya alıp daha öncelikli başka işlevleri yerine getirmeye başlayabilir ve dolayısıyla kontrolcü için gerekli sınırlı zamanların aşılmasına neden olabilir. Bundan dolayı kontrolcülerin gerçek zamanlı olarak çalıştırılması için QNX, VxWorks, RT-Linux, xPC Target gibi gerçek zamanlı çalışmaya olanak veren işletim sistemleri kullanılmaktadır. Bu işletim sistemleri sadece kontrolcünün yapması gereken işlevleri yerine getirecek şekilde kullanılabilmekte ve bu sayede kontrolcü için gerekli zaman sınırları tutturulabilmektedir.

Kontrolcü tasarımında gerçek zamanlı simülasyonlara olanak veren bir çok platform mevcuttur. Bunlara örnek olarak xPC Target, dSPACE, LabView, Dymola, Easy 5 verilebilir. Bu sistemler gerçek zamanlı simülasyonlara ilaveten birer hızlı kontrolcü prototiplendirme platformu olarak da kullanılabilmektedir.

3.2.1. dSPACE Sistemi

dSPACE, daha ziyade otomotiv uygulamaları için, ama havacılık, uzay, endüstriyel otomasyon sistemlerini de kapsayacak şekilde değişik sektörler için de , çeşitli boyutlarda ve ihtiyaçlara göre çeşitli konfigürasyonlarda gerçek zamanlı simülasyon sistemleri geliştirmektedir. dSPACE sistemi, djital sinyal işleyici (DSP) ve mikrokontrolör tabanlı hızlı donanımlar ve bunları destekleyen yazılımlar sayesinde hızlı kontrolcü prototiplendirme, otomatik kod üretme ve gerçek zamanlı donanım içeren simülasyonların oldukça başarılı bir şekilde gerçekleştirildiği sistemlerdir. Üzerinde bulunan güçlü donanımlar sayesinde bir çok gerçek zamanlı sistemden çok daha hızlı ve başarılı simülasyonlara olanak vermektedir. Bununla birlikte ControlDesk, ConfigurationDesk, MotionDesk, AutomationDesk gibi paket yazılımlar sayesinde gerçek zamanlı simülasyonların gözlenmesi, yönetilmesi ve veri kaydetme oldukça kolaylaştırılmıştır. Yine bu programlar sayesinde gerçek zamanlı simülasyonlar için başarılı görsel arayüzler geliştirilebilmektedir.

dSPACE sistemi MATLAB/SIMULINK ile birlikte çalıştırılmaktadır. Simulink ortamında modellerin geliştirilebilmesi için RTILIB adı verilen kütüphaneler geliştirilmiştir. Buna göre blok diyagramlar yardımıyla modeller kurulmakta ve gerçek

zamanlı olmayan simülasyonlar yapılmakta, daha sonra otomatik kod üreteci ve dSPACE RTI gerçek zamanlı arayüzü sayesinde bu modellerden oldukça başarılı ve optimum boyutta kod üretilebilmekte ve tasarlan kontrolcüler gerçek zamanlı olarak çalıştırılabilmektedir.

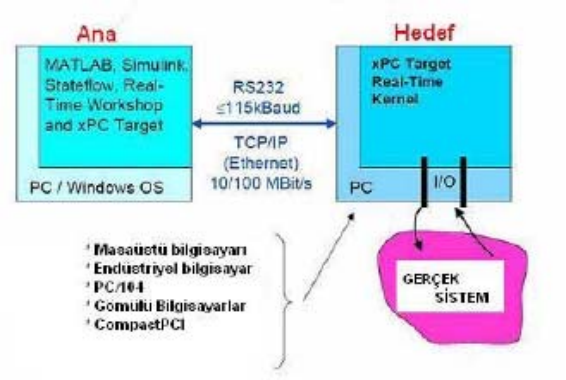
Şekil 3.1 : dSPACE sistemi [25].

dSPACE çözümleri, DS1103, DS1005, DS2210, DS2211 gibi genişleme üniteleri üzerinde modüler yapıda olabileceği gibi DS1401/1501/1505/1506 MicroAutoBox ve RapidPro System gibi portatif ürünler şeklinde de kullanılabilir. Bu ürünler ve üzerlerinde yer alan dijital / analog giriş – çıkış, kodlayıcı, CAN, LIN gibi kontrolcü tasarımında gerekli arayüzler sayesinde dSPACE, hızlı kontrolcü prototiplendirme ve gerçek zamanlı donanım içeren simülasyonlar için oldukça yararlı bir platformdur. Ayrıca müşteri taleplerine göre özel modüllerin geliştirilmesi, ürünlerin esnek konfigürasyonu ve geçmiş büyük projelerdeki güvenirliliği özellikle otomotiv alanında dSPACE ürünlerinin sıklıkla tercih edilmesine neden olmaktadır. Buna karşılık dSPACE sistemleri oldukça pahalı sistemlerdir.

3.2.2. xPC Target ve xPC TargetBox

xPC Target, Mathworks firması tarafından geliştirilen MATLAB/SIMULINK tabanlı bir sistemdir. Şekil 3.2'de xPC Target'ın genel yapısı gösterilmiştir.

xPC Target, ana ve hedef olmak üzere iki bilgisayar sisteminden oluşmaktadır. Ana bilgisayar, üzerinde MATLAB/SIMULINK ve xPC Target programlarının hızlı kontrolcü prototiplendirme sistemi olarak yer aldığı MS Windows işletim sisteminin çalıştığı bir masaüstü bilgisayarıdır. Hedef bilgisayar ise, PC uyumlu, minimum bilgisayar bileşenlerinin yeterli olduğu ve üzerinde xPC Target gerçek zamanlı işletim sisteminin çalıştırıldığı bir bilgisayardır.



Şekil 3.2 : xPC Target genel yapısı [26].

xPC Target tabanlı gerçek zamanlı simülasyon sisteminde, MATLAB/SIMULINK ile modellenen bir sistemin veya tasarlanan kontrolcünün xPC Target üzerinde gerçek zamanlı olarak çalıştırılabilmesi için ilk olarak bu modelden C kodunun üretilmesi gerekmektedir. Real Time Workshop kullanılarak Simulink modelinden C kodu üretilir ve bir derleyici aracılığıyla bu kod derlenir. Bağlantı türünün seçimine göre, RS232 veya TCP/IP, hedef bilgisayar ile bağlantı kurulur ve kod hedef bilgisayara yüklenir. Daha sonra kod hedef bilgisayarda gerçek zamanlı olarak çalıştırılır ve gömülü sistemimiz yapmasını kodladığımız işlevleri yerine getirir.

xPC TargetBox ise, yine Mathworks firması tarafından özellikle gerçek zamanlı donanım içeren simülasyonlar için geliştirdiği, üzerinde CAN, dijital – analog giriş – çıkış, LAN, kodlayıcı girişleri gibi bir çok gerçek zamanlı uygulama tarafından ihtiyaç duyulan opsiyonlara sahip olan ve xPC Target gerçek zamanlı işletim sisteminin çalıştığı gömülü bir sistemdir. Şekil 3.3'de xPC TargetBox'un görünüşü verilmiştir.

Şekil 3.3 : xPC TargetBox [26].

4. DONANIM İÇEREN SİMÜLASYON

"Donanım İçeren Simülasyon" deyimi İngilizce'deki "Hardware-In-The-Loop, HIL" deyimine karşılık olarak kullanılmaktadır. Her ne kadar Türkçe literatürde "Çevrimde Donanım Simülasyonu" deyimini kullananlar olsa da bu tez kapsamında birinci deyim tercih edilmektedir.

Donanım içeren simülasyon, karmaşık gerçek zamanlı gömülü sistemlerin testlerinde ve geliştirilmesinde yaygın olarak kullanılan bir tekniktir. Donanım içeren simülasyonların amacı, gerçek zamanlı gömülü sistemlerin geliştirilmesi ve test edilmesinde kullanışlı bir platform sağlamaktır. Donanım içeren simülasyon, kontrol edilen sistemi de test platformuna dahil ederek efektif bir platform sağlar. Kontrol edilen sistem, ilgili bütün dinamiklerinin matematiksel ifadeleriyle geliştirme ve test evrelerine dahil edilir ve bu matematiksel temsil, "sistem simülasyonu" olarak adlandırılır [26].

4.1. Donanım İçeren Simülasyonlara Neden İhtiyaç Duyulur ?

Donanım içeren simülasyonlar günümüzde bir çok sektörde gömülü sistemlerin geliştirilmesinde yaygın olarak kullanılmaktadır. Yeni bir kontrol sistemi geliştirilirken ve son prototipin geliştirilmesinden önce kontrol işlevlerinin doğru çalışıp çalışmadığının test edilmesi gerekir. Bu testlerin yapılması bazen mümkün olmayabilir, mümkün olsa bile oldukça maliyetli olabilir. Örneğin bir uçak veya helikopter için kontrol sistemi tasarlandığını düşünürsek, uygulamaların ve testlerin doğrudan hava aracı üzerinde yapılması hem maliyetli hem de tehlikeli olabilir. Kontrolcünün çalışmaması veya yanlış çalışmasından dolayı hava aracı düşebilir ki, bu da maddi manevi kayıplar demektir. Bunun yerine kontrol edilen sistemin dinamiklerini gerçekçi olarak yansıtan bir simülasyon modeli geliştirilebilir ve gerçek uçuş kontrol sistemi donanımı ve yazılımı bu simülasyona dahil edilebilir. Bu sayede kontrol sisteminin testleri yerde laboratuar ortamında yapılabilir. Otomobil için yeni bir kontrolcü geliştirilirken her yeni gelişmede araç ile yol testlerine çıkmak gerekmez. Bir prosesteki robot için geliştirilen kontrol sisteminin her defasında robot üzerinde birebir uygulamalı olarak denenmesi de bu yaklaşımla önlenir.

Bir kontrol sisteminin geliştirilmesi ve test sürecindeki verimlilik kriteri, maliyet, zaman ve güvenlik faktörlerini içeren bir formülle belirlenir. Maliyet faktörü, bu süreç esnasındaki geliştirme araçları ve işçilik maliyetlerini kapsar. Geliştirme ve test süresi, ürünün ticari hüviyet kazanması için geçen zamanı etkiler. Donanım içeren simülasyonların kullanımını zorunlu kılan belirli koşullar vardır. Bunlar, geliştirme sürecinde sınırlı zaman ve kesin ürün bitiş tarihleri, kontrol edilen sistemlerin yüksek maliyeti ve kontrol sisteminde insan faktörüdür. Bu faktörler aşağıdaki gibi kısaca özetlenebilir [27] :

- **Sınırlı Zaman:** Sınırlı ve kesin zaman kısıtlamalarından dolayı, kontrol siteminin test edilebilmesi için bir prototip ürününün beklenilmesine zaman olmayabilir. Böyle durumlarda kontrol edilen sistemin geliştirilmesi sürecine paralelel olarak donanım içeren simülasyonlar ile kontrol sistemleri de geliştirilir. Yani yeni bir sistem geliştrildiğinde onun kontrolcüsü de büyük oranda eş zamanlı olarak geliştirilmiş olur. Örneğin otomobil için yeni bir motor geliştirildiğinde, donanım içeren simülasyonlar sayesinde motor kontrol ünitesi de neredeyse bitmiş olur.

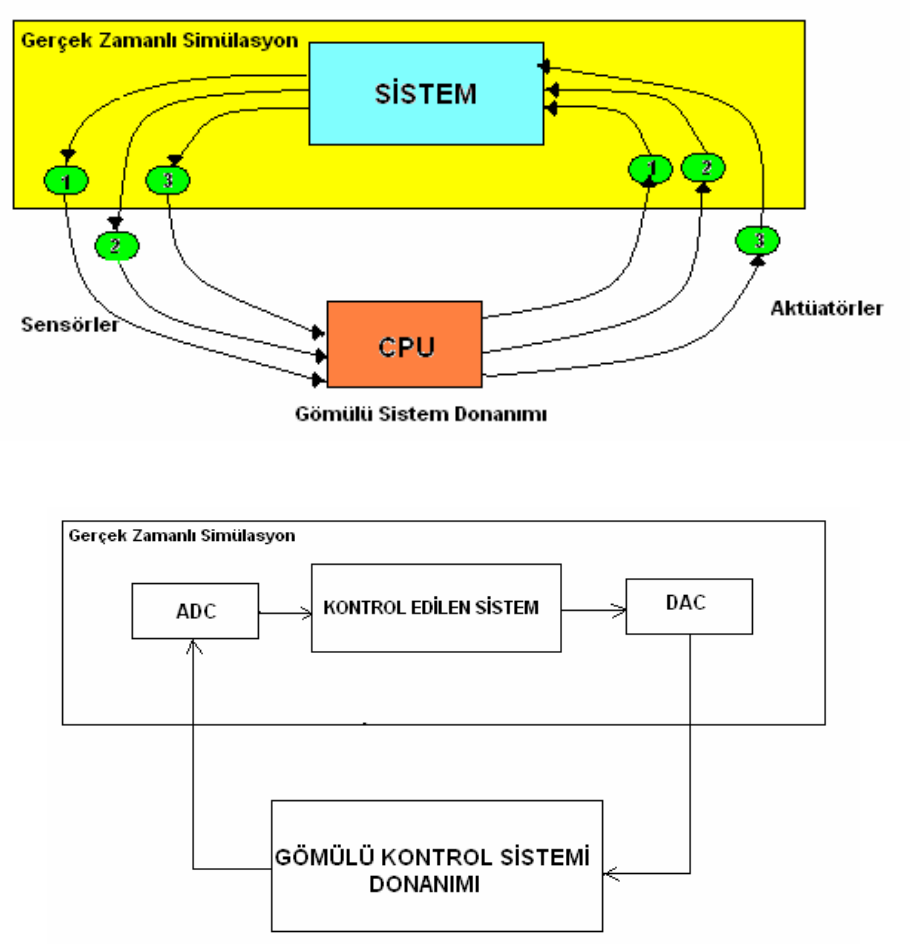
- **Yüksek Sistem Maliyeti:** Kontrol sisteminin test edilebilmesi için en kolay çözüm, kontrol sistemini doğrudan kontrol edilen sisteme bağlamaktır. Fakat kontrol edilen sistem oldukça pahalı olabilir. Bunun yerine güçlü ve gerçekçi bir gerçek zamanlı simülatörünü yapmak ve kontrol sistemini bu simülator üzerinde donanım içeren simülasyonlar ile test etmek daha ucuz bir çözüm olabilir. Örneğin uçaklar için jet motorları geliştiren bir firmayı düşünürsek, milyonlarca dolar değerindeki motorlar üzerinde FADEC kontrol sisteminin test edilmesi için donanım içeren simülasyonların kullanılması daha ucuz bir çözüm olabilir.

- **İnsan Faktörü:** Eğer kontrol işlevine insan da dahil olursa, kontrol sisteminde insan faktörü de incelenmelidir. Bunun için pilot içeren simülasyonlar kullanılmaktadır. Örneğin Fly-By-Wire uçuş kontrol sisteminin geliştirilmesini ele alalım: Fly-By-Wire uçuş kontrol sistemi, uçağın kontrol çubukları ile kontrol yüzeyleri arasındaki mekanik bağlantıları ortadan kaldırır. Kontrol giriş isteği sensörler aracılığıyla algılanır, ilgili yüzeylere iletilir ve elektrik motorları kullanılarak kontrol çubuklarına kuvvet uygulanır. Fly-By-Wire uçuş kontrol sisteminin davranışı kontrol algoritmaları ile belirlenir. Algoritmanın parametrelerindeki bir değişiklik, belirli bir kontrol girişine karşılık sistemin cevabını etkileyebilir. Yine benzer şekilde,

algoritmanın parametrelerindeki değişiklik kontrol çubuklarına uygulanan kuvveti de etkileyebilir. Gerçek parametereler kişiye özeldir. Dolayısıyla optimum parametre değerlerinin elde edilebilmesi için pilot içeren simülasyonların yapılması gerekir.

4.2. Donanım İçeren Simülasyon Nasıl Yapılır ?

Şekil 4.1'de donanım içeren simülasyonların genel yapısı verilmiştir. Donanım içeren simülasyonlarda kontrol edilen sistemin matematiksel modeli mümkünse sensör ve aktüatör dinamikleri de dahil edilerek modellenir ve bu sanal sistem simülasyonu gerçek zamanlı olarak çalıştırılır. Şekil 4.1'den de görüleceği gibi bazı aktüatör ve



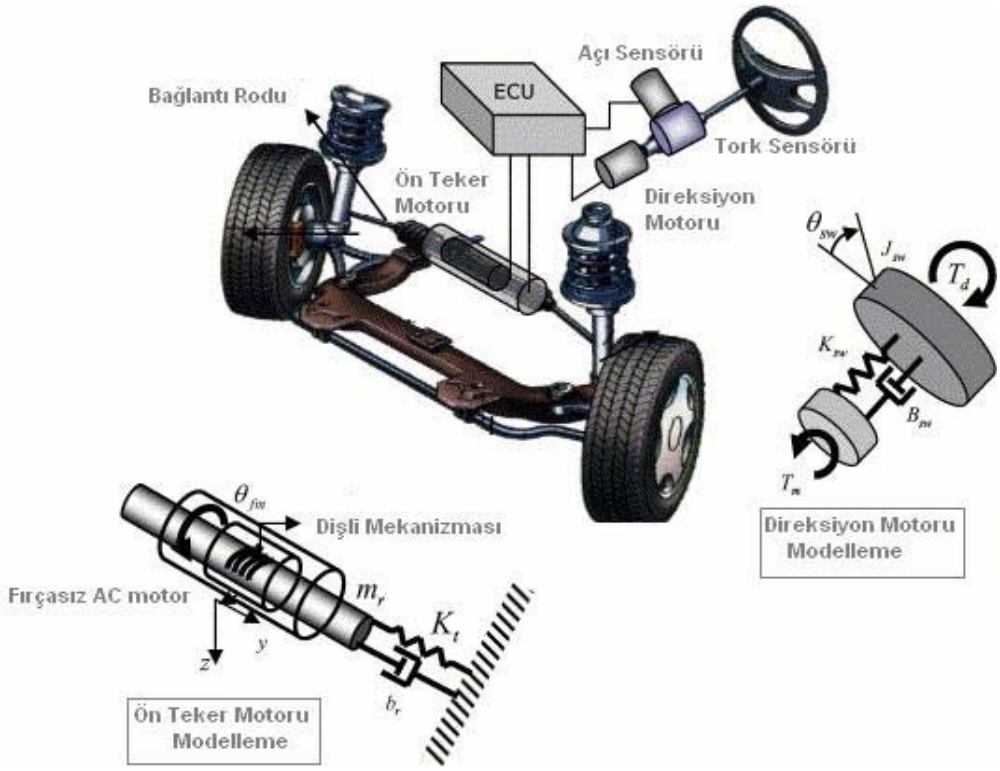
Şekil 4.1 : Donanım içeren simülasyonların genel yapısı [28].

sensörlerin sanal modelleri yerine gerçek donanımlar da simülasyona dahil edilebilir. Test edilen kontrol sistemi de gömülü kontrol ünitesi üzerine prototiplendirilmiştir ve kontrol edilen sistemin gerçek zamanlı sanal simülasyonuna sensörler ve aktüatörler aracılığıyla bağlanmıştır. Bu simülasyon yapısı donanım içeren simülasyon olarak

adlandırılmaktadır. Kontrol edilen sistemin gerçek zamanlı simülasyonu için, üzerinde gömülü kontrol sistemi ile iletişimi sağlamak için gerekli giriş çıkış birimlerini bulunduran, gerçek zamanlı çalışabilcek hızda ve güçte donanımlara ihtiyaç duyulmaktadır. Daha önceki bölümlerde gerçek zamanlı simülasyonların nasıl yapıldığı ve nasıl kontrolcü prototiplendirildiği konularından bahsedilmişti.

4.3. Donanım İçeren Simülasyon Örneği – 1

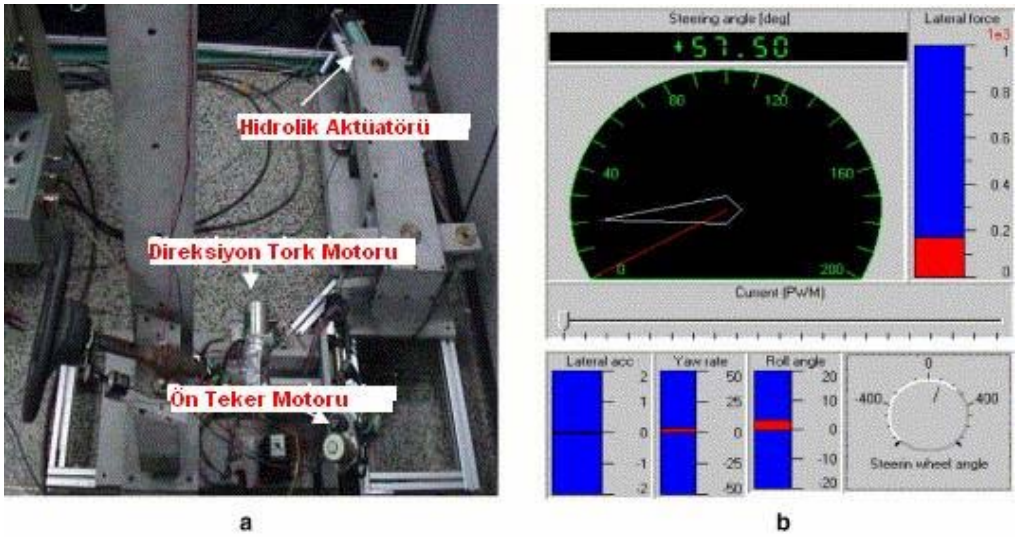
Bu örnekte otomobiller için Steer-by-Wire sisteminin donanım içeren simülasyonlar ile geliştirilmesinden bahsedilecektir. Steer-by-Wire sisteminde direksiyon ile ön tekerler arasındaki mekanik bağlantı ortadan kaldırılmakta ve elektrik kabloları ile gerekli bağlantı sağlanmaktadır. Şekil 4.2'de Steer-by-Wire sistemi gösterilmiştir.



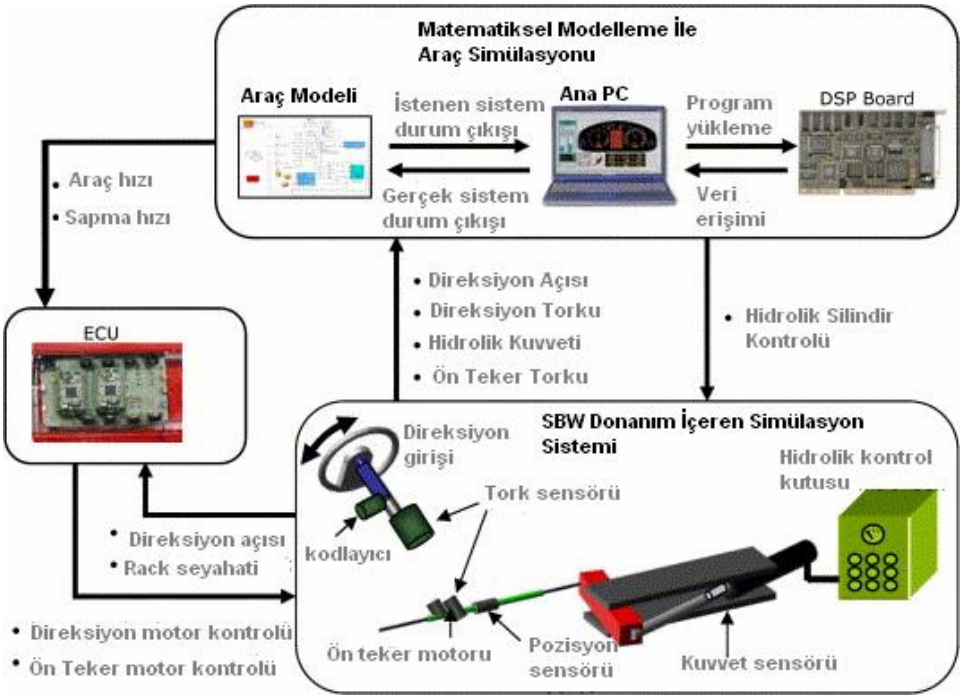
Şekil 4.2 : Steer-by-Wire sisteminin genel yapısı [29].

Steer-by-Wire sistemi, ön tekerlerin ve dolayısıyla aracın dönmesi için gerekli torku üreten ön teker motor sistemi ve yol etkilerinin sürücüye hissettirilmesi için kullanılan direksiyon motor sisteminden oluşur. Direksiyon motor sistemi, direksiyon ve motordan oluşur. Direksiyon motor sistemi üzerinde bir kodlayıcı ve bir de tork sensörü bulunmaktadır. Ön teker motor sistemi ise, bir bağlantı elemanı, bir

fırçasız DC Motor ve rack'ın seyahatini ölçmek için kullanılan bir kodlayıcıdan oluşur.



Şekil 4.3 : Donanım içeren simülasyon donanımı ve yazılımı [29].



Şekil 4.4 : Donanım içeren simülasyon sistemi blok diyagramı [29].

Şekil 4.3 ve Şekil 4.4'de sırasıyla donanımı ve yazılımı ve blok diyagramı verilen donanım içeren simülasyon sistemi Park vd [29] tarafından geliştirilmiştir. Donanım içeren simülasyon sistemi bir elektronik kontrol ünitesi, bir hidrolik sistem ve bir gerçek zamanlı kontrolcüden oluşur. Aracın tam matematiksel modeli geliştirilmiş ve

sanal modeldeki direksiyon sistemi Şekil 4.3'deki gibi donanım içeren simülasyon sistemi ile değiştirilmiştir. Direksiyon motor kontrolü ve ön teker motor kontrolü için PID tipinde kontrol algoritmaları geliştirilmiştir. Donanım içeren simülasyon sisteminde bir prototip elektronik kontrol ünitesi, ECU, geliştirilmiş ve kontrol algoritmaları bu kontrol sistemi üzerinde çalıştırılmıştır. ECU, kodlayıcı aracılığıyla direksiyon açısını, pozisyon sensörü aracılığıyla rack'ın seyahatini, araç simülasyonundan da araç hızı ve sapma hızını okumakta ve direksiyon motorunu ve ön teker motorunu sürmektedir. Direksiyon döndürüldüğünde gerçek araçta oluşan yanal kuvvetleri simülasyon sisteminde gerçeklemek için de hidrolik sistem kullanılmaktadır. Öncelikle aracın hızı ve direksiyon açısı ile Pacejka tekerlek modeli kullanılarak yanal kuvvet hesaplanmaktadır. Hidrolik sistem üzerinden de kuvvet ölçümü yapılmaktadır. Tasarlanan kayma kipli kontrolcü ile hidrolik sistemin kuvveti ile hesaplanan kuvvetin aynı olması sağlanmaktadır.

Bu donanım içeren simülasyon sistemi sayesinde Steer-by-Wire sisteminin geliştirilmesi ve testleri saha uygulamalarına gerek kalmaksızın laboratuar ortamında başarılı bir şekilde gerçekleştirilebilir. Bir çok kontrol senaryosu çok kolay ve hızlı bir şekilde test edilebilir ve direksiyon sistemi için optimum kontrol sistemi tasarlanabilir. Daha da önemlisi bu donanım içeren simülasyon sistemi aracılığıyla kontrolcü geliştirme süreci, gerçek araç ve saha testlerini içeren kapsamlı süreç ile zaman, maliyet ve güvenlik açısından karşılaştırıldığında çok büyük kazanımlar elde edildiği görülecektir. Yazarlar ayrıca gerçek araç verileri ile donanım içeren simülasyon sisteminin geçerliliği ve başarımını makalelerinde detaylı bir şekilde sunmuşlardır [28].

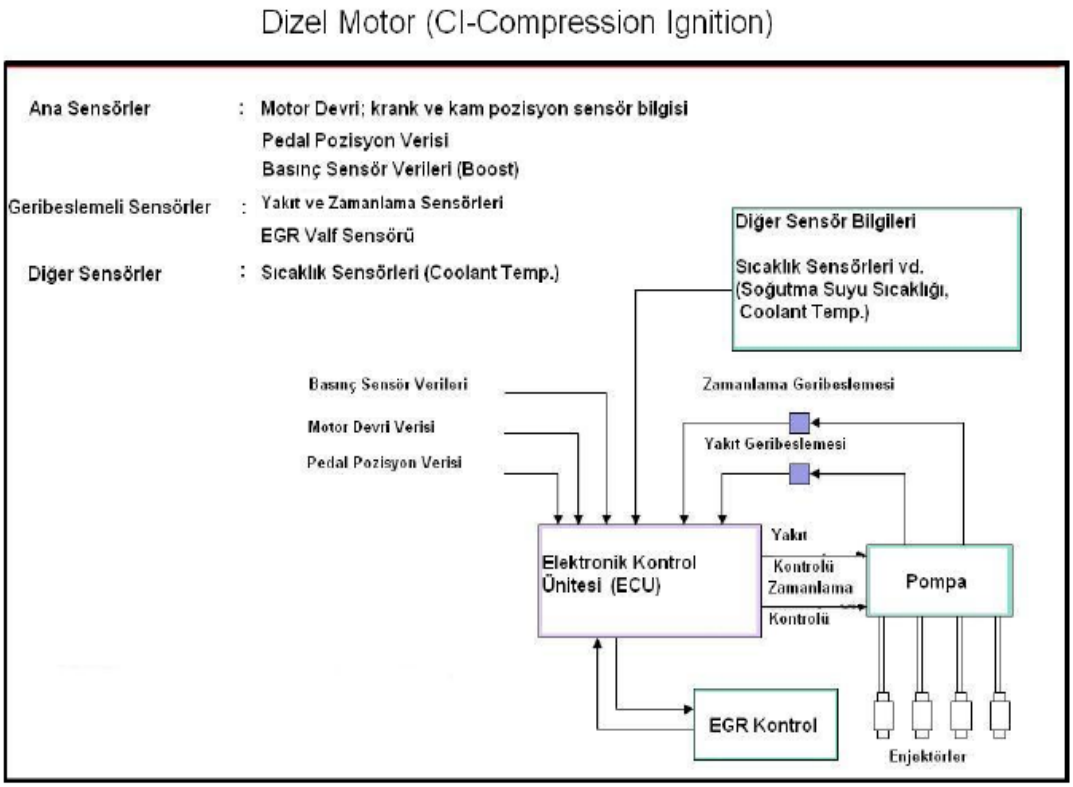
4.4. Donanım İçeren Simülasyon Örneği – 2

Bu örnekte de bir içten yanmalı motor elektronik kontrol ünitesinin gerçek zamanlı donanım içeren simülasyonlar ile testi hakkında bilgi verilecektir. Bu konu bağlamında İTÜ MEKAR kapsamında yapılan Çebi vd [30,31] çalışmalarına atıf yapılacaktır. Daha detaylı bilgi için ilgili kaynaklara bakılabilir.

Motor kontrol ve yönetim sistemleri motor gücü, yakıt tüketimi ve emisyon seviyelerini sensör girişlerini kullanarak denetler. Ayrıca olası sensör hatalarını ve diğer hataları da kontrol ederek gerekli diyagnostik sinyallerini oluştururlar. Yeni bir motorun geliştirilmesi esnasındaki zaman daraltıcı etmenlerden biri de elektronik kontrol ünitesindeki katsayı tablolarının kalibrasyonu ve tüm ECU yazılımının

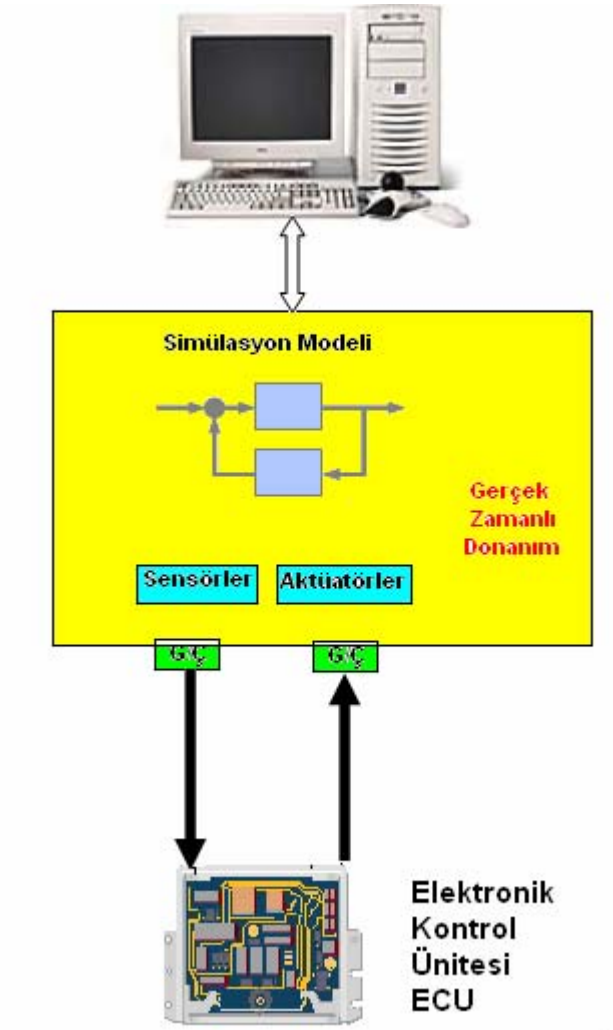
hatalara karşı sınanmasıdır. Test odasında ya da yolda yapılan testlerde tüm sensör hatalarının test edilmesine olanak yoktur, olsa bile maliyetli ve zaman alan işlerdir. Dolayısıyla otomotiv sektöründe motor elektronik kontrol ünitesinin sınanmasını hızlandırma amaçlı olarak donanım içeren simülasyonlar kullanılmaktadır [31]. Zaten içten yanmalı motor ECU testleri, donanım içeren simülasyonların ilk ve yaygın olarak kullanıldığı alanlardan biridir.

ECU, motor kontrol sistemindeki işlevini tam olarak yerine getirebilmek için motor devri, krank ve kam durumu, hava miktarı, motor sıcaklığı, motor yükü, gaz pedalı açısı gibi bilgilere ihtiyaç duymaktadır. ECU bu bilgileri kullanarak enjeksiyon zamanlamasını ve gerekli yakıt miktarını ayarlar [31]. Şekil 4.5'te ECU'nun işlevi bir diyagram üzerinde verilmiştir.



Şekil 4.5 : Bir dizel ECU'sunun işlevsel diyagramı [31].

ECU'nun donanım içeren simülasyonlarda test edilmesi için Şekil 4.6'daki yapının oluşturulması gerekir. Buradaki amaç elektronik kontrol ünitesini sanki gerçek bir araç üzerinde gerçek bir motor ile çalışıyormuş gibi kandırmaktır. Bunun için ECU'yu doğrudan gerçek bir motora bağlamaktansa, motorun bir simülasyon modeli kullanılmaktadır. Bu simülasyon modeline sensör ve aktüatör dinamikleri doğrudan eklenebileceği gibi gerçek donanımlar da simülasyona dahil edilebilir. Buna göre



Şekil 4.6 : ECU donanım içeren simülasyon test diyagramı.

ECU'nun ihtiyaç duyduğu Şekil 4.5'de belirtilen sensör bilgileri, sinyaller, simülasyon modelinde sanal olarak oluşturulabilir ve bu sayede ECU kandırılabilir.

ECU'nun ihtiyaç duyduğu sinyallerden kam ve krank sinyallerinin simülasyon ortamında oluşturulması hızlı ve güçlü, gerçek zamanlı olarak çalışabilen donanımları gerektirmektedir. Dolayısıyla simülasyon modelinin gerçek zamanlı olarak çalıştırılabileceği donanım seçilirken, maksimum motor hızı için gerekli kam ve krank sinyallerini üretebilmesi ve doğru ve tutarlı bir şekilde çalışabilmesi aranan en temel kriterler olmalıdır.

Çebi vd. [30,31] yaptıkları çalışmalarında bu kriterler ışığında dSPACE 1103 ve 2210, National Instruments, xPC TargetBox gibi donanımsal çözümleri incelemiş ve yukarıdaki kriterler ışığında bu gerçek zamanlı donanım içeren simülasyonlara müsade eden sistemleri karşılaştırmışlardır. Ayrıca simülasyonlarda gerçek yol

verileri kullanılmış ve ECU'nun hata simülasyonları da yapılmıştır. Yapılan simülasyonlarda motor modeli olarak literatürde MVEM, Mean Value Engine Model, olarak adlandırılan motor modeli kullanılmıştır.

5. HELİKOPTER SİMÜLATÖRÜ

Günümüzde helikopterler gerek sivil, gerekse askeri amaçlı olarak değişik görevlerde kullanılmaktadır. Uzun süre havada asılı vaziyette kalabilmesi, yerden doğrudan dikey yönde kalkış yapabilmesi gibi üstün özellikleri helikoptere ayrı bir önem katmaktadır. Dolayısıyla helikopterler, arama-kurtarma, hasta nakil ve kısa mesafe uçuşların temel aracıdır.

Helikopterler uçaklara göre çok daha karmaşık ve kontrol edilebilmesi çok daha zor hava araçlarıdır. Genel olarak helikopter dinamiği kararsızdır. Bu durum bilhassa askı halindeki bir helikopter için geçerlidir. Helikopteri kararlı hale getiren saykılik kumandaları ile sürekli olarak girişleri değiştiren pilottur. Pilotun da sistemi kararlı hale getiremeyeceği durumlarda kararlılık artırıcı sistemlere ihtiyaç duyulmaktadır. Bu nedenle helikopterin gerçekçi bir dinamik modelinin kurulması, bu modeller üzerinde kontrol uygulamalarının geliştirilmesi ve pilotlu gerçek zamanlı simülasyonlarının yapılması gerekir. Bu amaçla gerçek zamanlı helikopter simülatörleri kullanılmaktadır. Bu simülatörler ile bir helikopterin gerçek zamanlı simülasyonu yapılabilmekte, helikopter için geliştirilen kontrol algoritmaları ve otopilot sistemleri test edilebilmektedir. Hatta helikopterler için değişik uçuş ortam koşulları ve görevler oluşturulabilmektedir. Ayrıca pilot kontrol çubukları da simülatöre dahil edilerek gerçek zamanlı pilotlu donanım içeren simülasyonlar yapılabilmektedir.

Geliştirilen kontrol algoritmalarının gerçek helikopter üzerinde çalışıp çalışmayacağının test edilebilmesi ve pilotun kontrol girişleri ile simülasyona dahil olmasından dolayı, simülasyon ortamının gerçek zamanlı olarak çalışması gerekir. Günlük hayatımızda yoğun olarak kullandığımız Linux, Windows, MAC OS, vb. işletim sistemlerinin gerçek zamanlı çalışmaya elverişli olmamasından dolayı, gerçek zamanlı çalışabilecek sistemlere ihtiyaç duyulmaktadır. RTLAB, VxWorks, RTLinux, xPC Target, QNX gibi gerçek zamanlı çalışabilen bir işletim sistemine ve bunları destekleyen donanımlara ihtiyaç vardır.

Helikopter simülasyonları temelde pilot eğitimleri ve uçuş kontrol sistemlerinin tasarlanması amacıyla kullanılmaktadır. Günümüzde bu amaçla geliştirilmiş bir çok

helikopter simülatörü mevcuttur. ALSIM'ın geliştirmiş olduğu Şekil 5.1'deki simülatör gerçek zamanlı olarak çalışabilen, C programlama dili ile programlanmış, özel bir platform ve kokpit sistemi üzerine temellendirilmiş bir uçuş simülatörüdür [32].

Şekil 5.1 : ALSIM uçuş simülatörü [32].

Bihrle Applied Research Inc.'nin geliştirmiş olduğu D-Six tabanlı, gerçek zamanlı olarak çalışabilen, Matlab/Simulink ve D-Six Target tabanlı çalışan ve donanımsal platformlar ve avyonikler ile desteklenebilen uçuş simülatörü Şekil 5.2'de gösterilmiştir [33]. Bunun dışında Flightlab firmasının geliştirdiği HeliFlight [34] ve ASELSAN'ın uçuş simülatörleri örnek olarak gösterilebilir.

Şekil 5.2 : D-Six uçuş simülatörü [33].

Helikopter simülatörleri, içerdikleri yazılım, donanım ve gerekli avyonik'lerden dolayı oldukça pahalı sistemlerdir. Dolayısıyla böyle bir simülatörün geliştirilmesi oldukça maliyetli olduğu gibi, uçuş kontrol sistemlerinin gelştirilmesi için bireysel çalışmalar yapan bir akademisyen ve tasarım ve kontrol mühendisleri için uygun değildir. Daha düşük maliyetli, gerçek zamanlı, pilotlu helikopter simülatörünün sahip olması

gereken asgari bileşenleri içeren simülatörlere ihtiyaç duyulmaktadır. Bu tez kapsamında geliştirilen helikopter simülatörü bu amaçla kullanılabilecek bir simülatördür.

5.1. Helikopter Simülatörü Tasarımı

Bu tez kapsamında düşük maliyetli gerçek zamanlı ve pilotlu bir helikopter simülatörü ortamı geliştirilmiştir. Şekil 5.3'de pilotlu simülasyon ortamı gösterilmiştir.

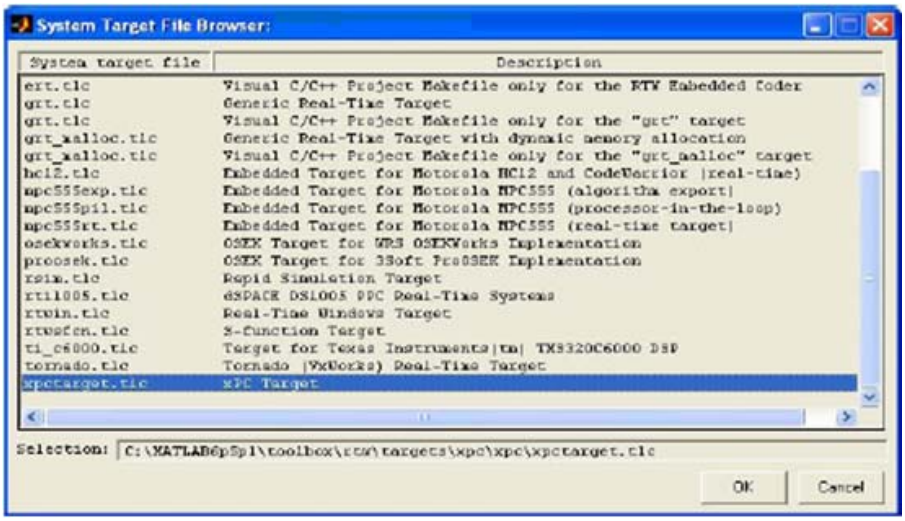
Şekil 5.3 : Pilotlu simülasyon ortamı.

Burada sunulan simülatör, MATLAB/SIMULINK, xPC TargetBox, FlightGear, Microsoft SideWinder ForceFeedback Joystick üzerine kurulmuş gerçek zamanlı çalışabilen bir uçuş simülatörüdür.

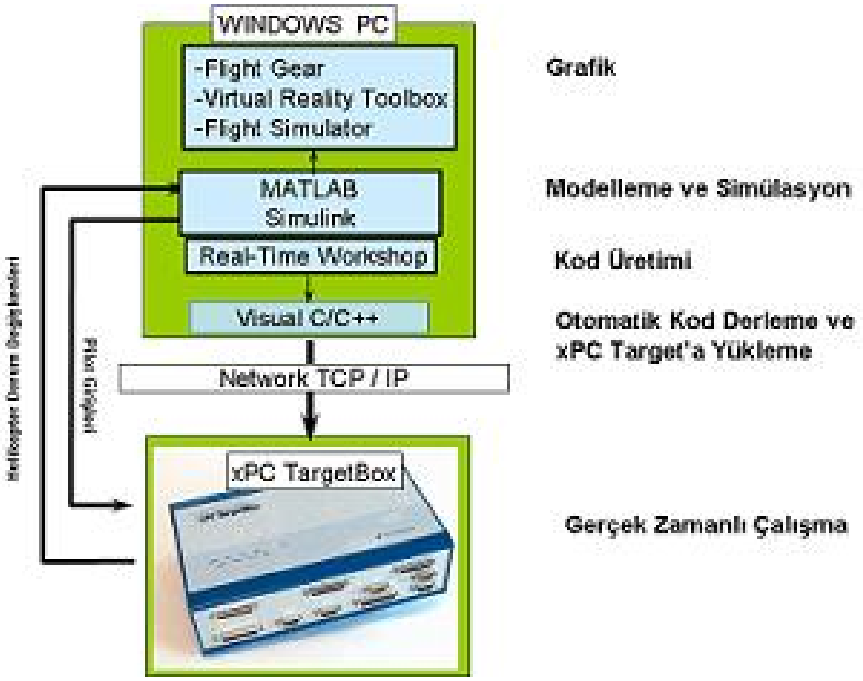
Simülatörde kullanılan helikopter modeli, MATLAB/SIMULINK ortamında geliştirilmektedir. Geliştirilen bu helikopter modelinin gerçekçi olup olmadığını, gerçek helikopterden alınan validasyon test verilerini kullanarak teyit etmek gerekir.

MATLAB/SIMULINK ortamında geliştirdiğimiz modellerimizi ve kontrolcülerimizi simülatör ortamında test etmek için, öncelikle helikopter modelinden ve tasarlanan kontrolcülerden Real Time Workshop kullanılarak C kodu üretilir. Tabiki bunun için öncelikle, geliştirilen model üzerinde gerçek zamanlı olmayan simülasyonlardan farklı olarak bir takım düzenlemeler yapılır. Buna göre sabit adımlı nümerik çözücülerin seçilmesi, gerçek zamanlı simülasyon için sabit adım örnekleme

süresinin belirlenmesi, gerçek zamanlı sistemler içerisinden de xPC Target seçeneğinin seçilmesi gibi simülasyon parametreleri ayarlanır. Şekil 5.4'de MATLAB/SIMULINK içerisinde seçilebilecek gerçek zamanlı simülasyon ortamlarının bir kısmı gösterilmiştir. Daha sonra üretilen bu C kodu derlenip xPC TargetBox ortamına aktarılır. Şekil 5.5'de de modelleme, otomatik kod üretimi ve hedef bilgisayara yükleme safhalarını gösteren simülatör akış şeması verilmiştir.

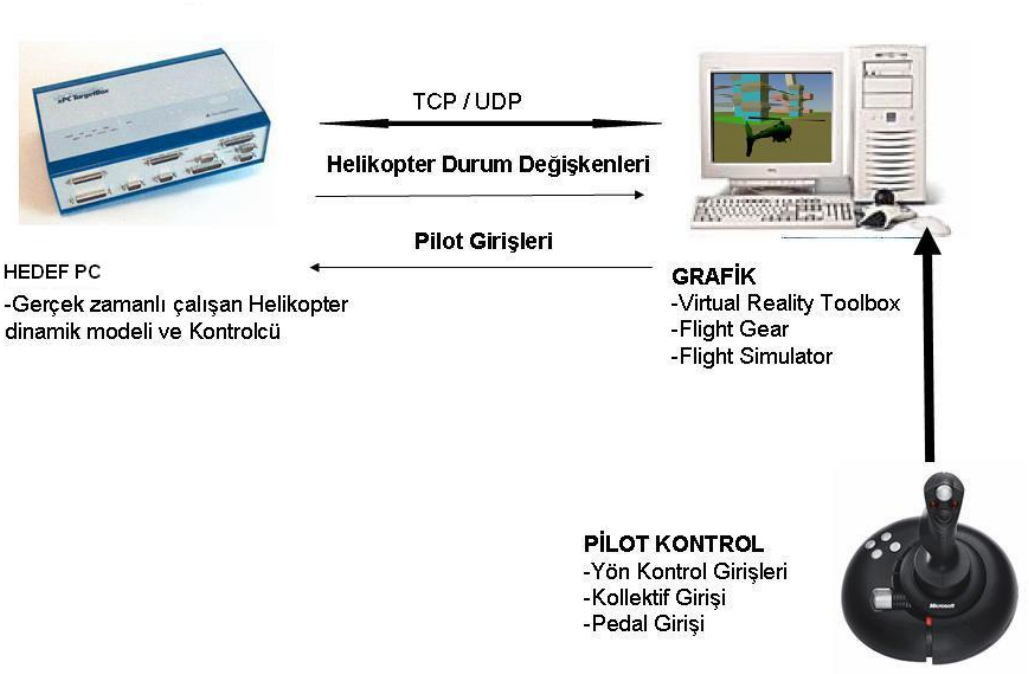


Şekil 5.4 : Gerçek zamanlı simülasyon ortamları ve xPC Target.

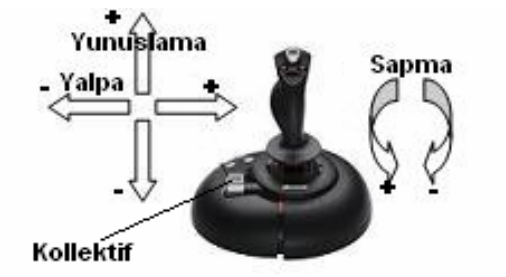


Şekil 5.5 : Simülatör akış diyagramı.

Daha önceki bölümlerde de bahsedildiği gibi xPC Target tabanlı gerçek zamanlı simülasyonlarda bir hedef bilgisayar, bir de ana bilgisayar yer almakta idi. xPC Target tabanlı bu simülatörde dinamik modeller, gerçek zamanlı simülasyon ortamı, hedef olarak kullanılan xPC TargetBox üzerinde çalıştırılmaktadır. Hedef bilgisayar üzerinde çalışan gerçek zamanlı helikopter modelinden yunuslama, sapma ve yalpa açıları ile hızları, konum bilgisi gibi durum değişkenleri alınır. Hedef bilgisayara ise pilot girişleri gönderilir. Hedef bilgisayar ile ana bilgisayar arasındaki iletişim ethernet üzerinden UDP/IP protokolü kullanılarak sağlanır. Uçtan uca ethernet hattı, 100Mbit/s iletişim hızı ile gerçek zamanlı iletişim imkanı sağlamaktadır. Şekil 5.6'da geliştirilen simülatörün genel yapısı verilmiştir.



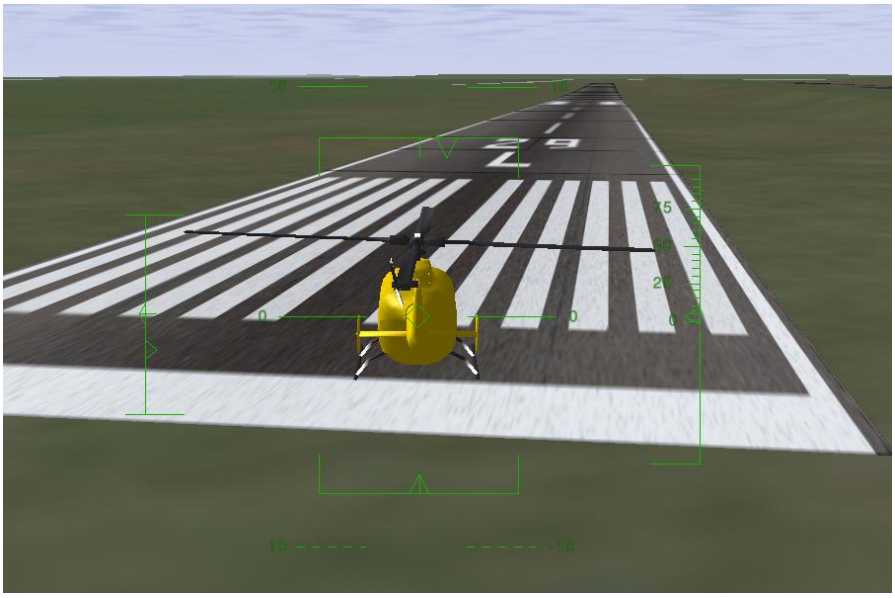
Şekil 5.6 : Simülatörün genel yapısı.



Şekil 5.7 : Microsoft SideWinder Joystick ve simülatörde karşılık gelen yönler.

Simülatörde pilot girişleri, ana bilgisayara bağlı Microsoft SideWinder ForceFeedback Joystick kullanılarak sağlanmaktadır. Bu joystick, 2 doğrusal ve bir de dönme olmak üzere toplam 3 eksen, bir kelebek girişi ve 8 buton ile istenen bütün pilot girişlerini sağlayabilmektedir. Joystick girişleri, ethernet üzerinden UDP/IP protokolü ile gerçek zamanlı olarak helikopter modeline girilir. xPC Target işletim sistemi USB Joystick'i desteklemediğinden, joystick ana bilgisayara bağlanmış ve pilot girişleri ethernet üzerinden gönderilmiştir. Şekil 5.7'de joystick hareketlerini ve helikopterde karşılık geldiği kontrol girişleri belirtilmiştir.

FlightGear programı normalde oyun amaçlı geliştirilmiştir. Yalnız program açık kod olduğundan, üzerinde değişiklikler yapmak mümkündür. Simülatörde FlightGear programı, helikopter durum değişkenlerini UDP/IP protokolü ile almaktadır. Bunun için programın ethernet üzerinden veri alacak şekilde ayarlanması gerekmektedir. Şekil 5.8'de simülatör görsellik arayüzünden bir enstantane gösterilmiştir.

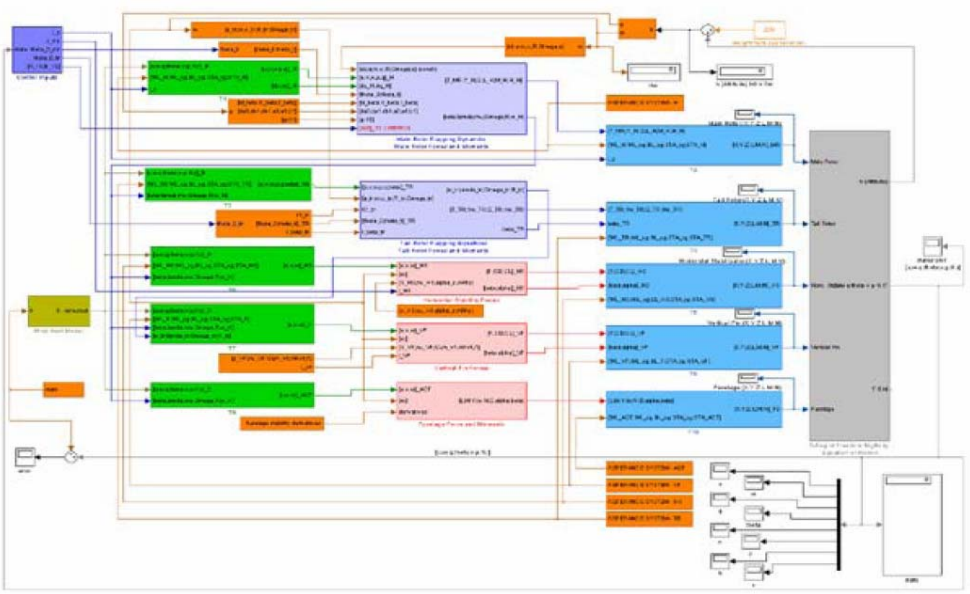


Şekil 5.8. Simülatör görsellik ortamından bir görüntü.

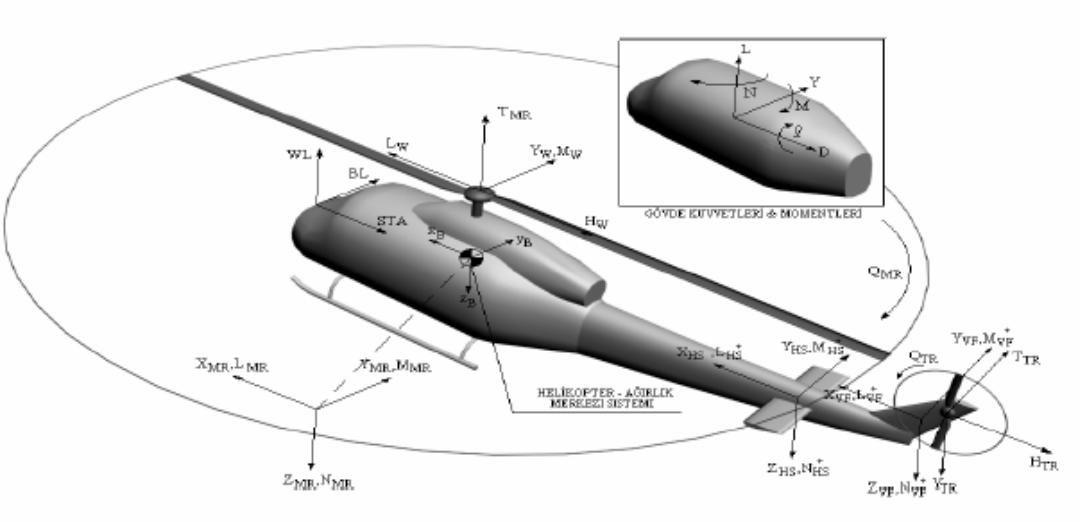
Geliştirilen simülatörde MATLAB/SIMULINK ortamında modellenmiş ya da dinamiği C++ ile kodlanmış herhangi bir helikopterin simülasyonu yapılabilir. Şu anda simülatör ortamında MATLAB/SIMULINK aracılığıyla Abdülhamitbilal [34] tarafından modellenmiş Bell205 helikopter modeli kullanılmaktadır. Kullanılan Bell205 helikopterine ait Simulink modeli Şekil 5.9'da verilmiştir.

Simülatörde kullanılan helikopter modeli, ana rotor, kuyruk rotor, yatay ve dikey stabilizatör ve gövde bileşenlerinden oluşmaktadır. Bu sistemlerin her birinin ayrı

ayrı dinamik ifadeleri çıkarılmış, Matlab/Simulink ortamında modellenmiş ve helikopterin ağırlık merkezi koordinat sistemi referans alınarak tüm sistemlerdeki kuvvet ve moment değişkenleri tespit edilmiştir. Şekil 5.10'da helikopter üzerindeki kuvvet ve moment ifadeleri gösterilmiştir. Burada $()_{MR}$ ana rotora ilişkin değişkenleri ; $()_{TR}$ kuyruk rotoruna ilişkin değişkenleri; $()_{VF}$ dikey stabilizatöre ilişkin değişkenleri; $()_{HS}$ yatay stabilizatöre ilişkin değişkenleri ifade etmektedir.



Şekil 5.9 : Doğrusal olmayan Bell205 Simulink modeli [35].



Şekil 5.10 : Helikoptere etkiyen kuvvetler ve momentler [35].

Bütün bu kuvvet ve moment bileşenleri dönüşüm matrisleri ile helikopter ağırlık merkezine taşınarak X,Y,Z eksenlerinde helikoptere etkiyen toplam kuvvet ve

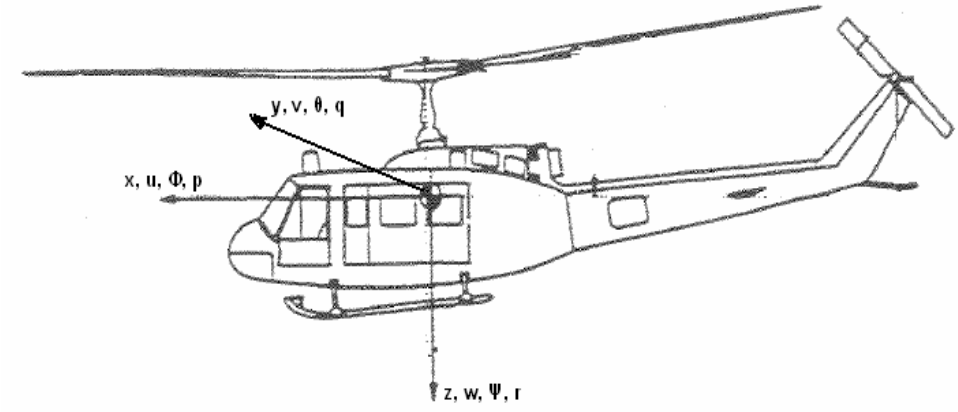
moment ifadeleri elde edilmiş ve hareket denklemleri bulunmuştur. Helikopterin her biri bileşenine ait kuvvet ve moment ifadeleri [35,36,37]'de detaylı bir şekilde bulunabilir.

Helikopter, üç tanesi döner hareket , üç tanesi de yer değiştirme için olmak üzere toplam altı serbestlik derecesine sahiptir. Şekil 5.11'de ağırlık merkezi eksen takımı üzerinde helikopter değişkenleri verilmiştir. Burada x, y, z hareket eksenleri; u, v, w yer değiştime hızları ; Φ, θ, Ψ Euler açıları; p, q, r de açısal hızlardır. Genel olarak Newton-Euler hareket denklemleri,

$$m\frac{dv}{dt} = F \qquad \textbf{(5.1)}$$

$$\frac{d(I\omega)}{dt} = M \qquad \textbf{(5.2)}$$

şeklindedir. $F = [X\ Y\ Z]^T$, helikopterin ağırlık merkezine etkiyen toplam kuvvetler ve $M = [L\ M\ N]^T$ toplam momentlerdir. Bir helikopterde ağırlık merkezine etkiyen toplam kuvvet ve momentler ana rotor, kuyruk rotoru, yatay ve dikey stabilizatörler, yer çekimi ve aerodinamik etkilerden ortaya çıkar.



Şekil 5.11 : Ağırlık merkezi referans sistemi ve helikopter değişkenleri.

Diğer taraftan genel olarak dinamik denklemler,

$$m\dot{v} + m(\omega x v) = F \qquad \textbf{(5.3)}$$

$$I\dot{\omega} + (\omega x I\omega) = M \qquad \textbf{(5.4)}$$

şeklinde ifade edilir. Yukarıdaki dinamik denklemlerde **v** = [u v w]^T gövde hızları ve **ω** =[p q r]^T açısal hızlarıdır. Denklem (5.3)'den helikopterin üç referans eksenindeki yer değiştirme hareketine ilişkin denklemler elde edilir. Bu hareket denklemleri,

$$\dot{u} = (-wq + vr) + X / m \quad \textbf{(5.5)}$$

$$\dot{v} = (-ur + wp) + Y / m \quad \textbf{(5.6)}$$

$$\dot{w} = (-vp + uq) + Z / m \quad \textbf{(5.7)}$$

şeklindedir. Benzer şekilde (5.4) denkleminden de (5.8), (5.9) ve (5.10) döner hareket denklemleri elde edilir.

$$\dot{p} = -qr(I_{yy} - I_{zz}) / I_{xx} + L / I_{xx} \quad \textbf{(5.8)}$$

$$\dot{q} = -pr(I_{zz} - I_{xx}) / I_{yy} + M / I_{yy} \quad \textbf{(5.9)}$$

$$\dot{r} = -pq(I_{xx} - I_{yy}) / I_{zz} + N / I_{zz} \quad \textbf{(5.10)}$$

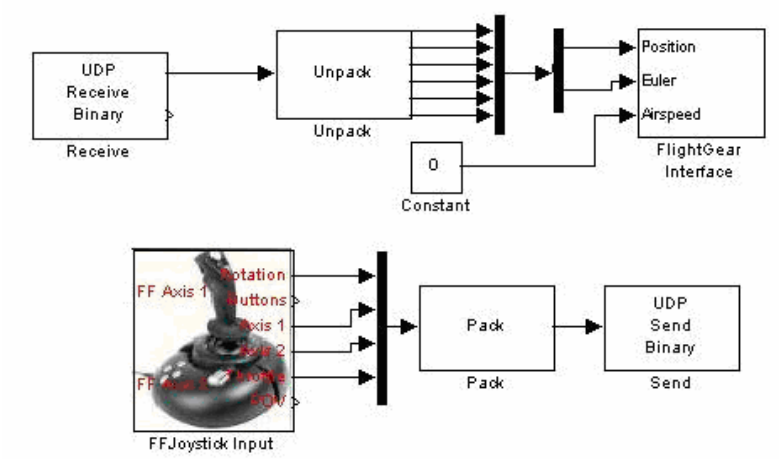
Burada $\mathbf{I_{xx}}$, $\mathbf{I_{yy}}$, $\mathbf{I_{zz}}$ sırasıyla x, y ve z eksenleri etrafıdaki ataletlerdir. (5.5) - (5.10) denklem takımları, etkiyen kuvvet ve momentlere karşılık helikopterin hareketini temsil etmektedir. Bu kuvvet ve momentler helikopter kontrol girişlerinin ve durum değişkenlerinin bir fonksiyonudur. Buna göre bu denklem takımları genel olarak (5.11)'deki gibi doğrusal olmayan diferansiyel denklem ile ifade edilebilir.

$$\dot{x} = f(x, u, t) \quad \textbf{(5.11)}$$

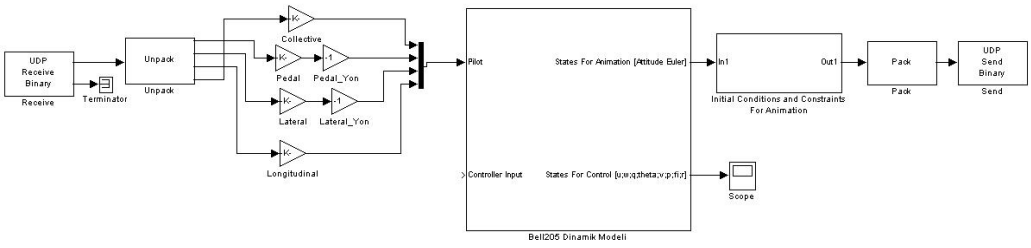
Burada **x** = [u, v, w, Φ, θ, Ψ, p, q, r]^T ve **u** = [A_{1c}, B_{1c}, θ_{0MR}, θ_{0TR}]^T 'dir. A_{1c} ve B_{1c} saykılik , θ_{0MR} kollektif ve θ_{0TR} de pedal girişleridir. **u** vektörü doğrudan rotor sistemindeki açılardır. Gerçekte kokpitten girilen pilot girişleri bir dizi mekanik veya elektromekanik sistemler aracılığıyla rotor sistemine iletilmektedir. Asıl kontrol giriş vektörü **δ** = [δ_{boy}, δ_{yan}, δ_{kol}, δ_{ped}]'dir. **δ**/**u** = **k** kumanda mekanizmasının kazançlarıdır.

Gerçek zamanlı simülatörde kullanılan Microsoft SideWinder Force Feedback Joystick ile pilot girişleri girilmekte ve helikopterin ana rotor yunuslama açısı, kuyruk rotoru yunuslama açısı, tabla asamblesi yalpa açısı ve tabla asamblesi yunuslama açısı değiştirilerek helikopter sistemi kontrol edilmektedir.

Pilot girişlerini joystick'den okumak için C++ programlama dili kullanılarak bir Simulink S-Function oluşturulmuştur. xPC Target UDP bileşeni ile joystick girişleri xPC TargetBox'a gönderilir. Şekil 5.12'de da bu amaçla kullanılan Simulink modeli verilmiştir. Ayrıca yine bu model aracılığıyla yine UDP bileşeni sayesinde xPC TargeBox'dan helikopter durum değişkenleri alınarak FlightGear ortamına gönderilir.



Şekil 5.12 : xPC Target, Joystick ve FlightGear iletişimi için Simulink modeli.



Şekil 5.13 : xPC TargetBox üzerinde gerçek zamanlı çalışan Simulink modeli.

xPC TargetBox üzerinde gerçek zamanlı olarak çalışan helikopter dinamik modeli de, görselliğin sağlandığı ana bilgisayara durum değişkenlerini ethernet üzerinden gerçek zamanlı olarak göndermektedir. Yine bunun için xPC Target UDP bileşeninden faydalanılmaktadır. Şekil 5.13'de xPC TargetBox üzerinde gerçek zamanlı olarak çalışan Simulink modeli verilmiştir.

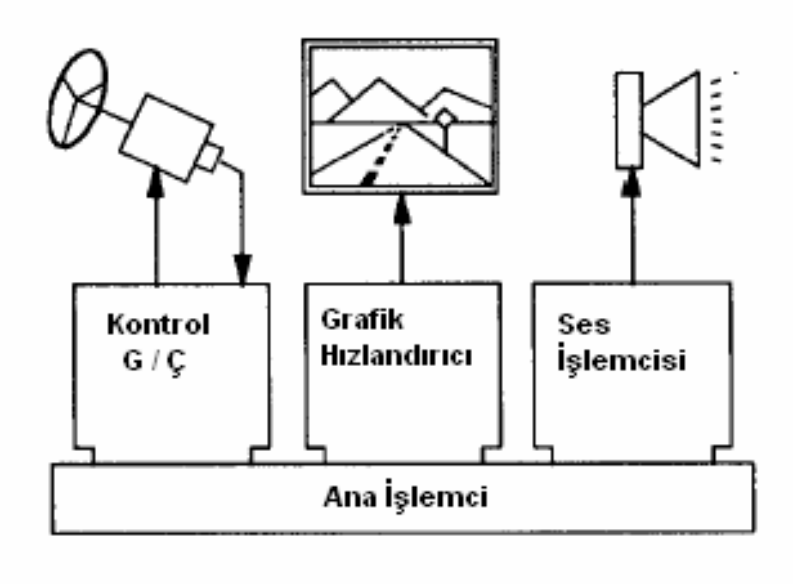
6. YOL TAŞITI SİMÜLATÖRÜ

Yol taşıtı simülatörleri günümüzde özellikle sürücü eğitimlerinde, otomotiv sektöründe araç kontrol sistemlerinin geliştirilmesi ve testlerinde ve sürücü davranışlarının belirlenmesinde yaygın olarak kullanılmaktadır. Buna karşılık bu tez kapsamında daha ziyade araç kontrol sistemlerinin geliştirilmesinde donanım içeren simülasyon sistemlerinin bir parçası olarak kullanılan taşıt simülatörleri dikkate alınmaktadır.

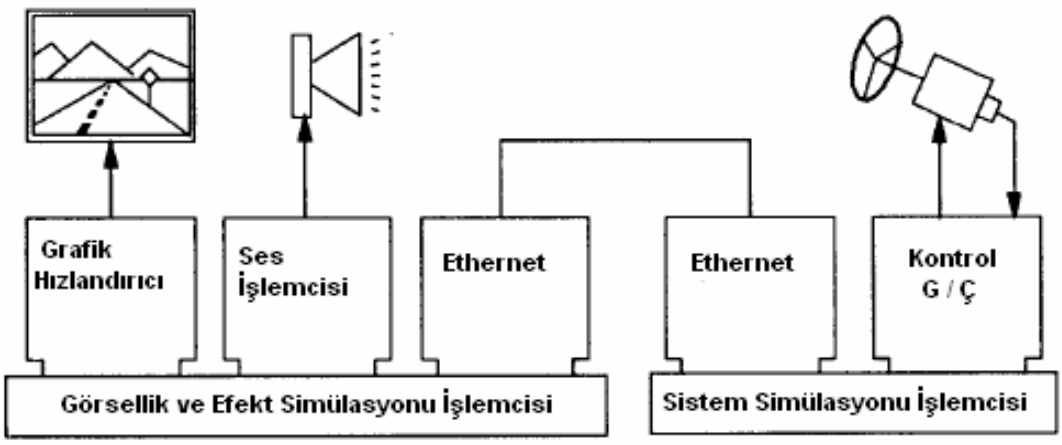
Daha önceki bölümlerde bahsedildiği gibi donanım içeren simülasyonlar ve simülatörler gömülü kontrol sistemlerinin geliştirilmesinde önemli bir role sahiptir. Geliştirilen kontrol sistemlerinin test edilmesi için, gerek maliyet ve gerekse zaman gibi kontrol sistemi tasarım kısıtlamalarından dolayı, doğrudan gerçek araç üzerine bağlamak yerine, kontrol edilen sistemin, yol taşıtı, gerçekçi bir sanal sistem simülasyonunun kullanılması ile yapılan donanım içeren simülasyonlar kullanılır. Eğer tasarlanan kontrol sisteminde , örneğin Elektronik Stabilite Programı (ESP), X-by-Wire sistemleri, sürücünün etkisi ve sürücü girişleri söz konusu ise sürücünün de bir şekilde simülasyon sistemine dahil edilmesi gerekmektedir. Gerçek araç üzerinde çalışacak olan kontrol sisteminde direksiyon girişi, gaz, fren ve debriyaj pedalı girişleri, vites girişi gibi veriler de kontrol parametresi olarak yer almakta ise bu girişlerin de donanım içeren simülasyonlarda dışarıdan alınması gerekmektedir. İnsanın da kontrol sistemine dahil olduğu bu simülasyonlara, İngilizce “Man-InThe-Loop, MIL” deyiminin karşılığı olarak “İnsan İçeren Simülasyonlar” adı verilmektedir. Gerçek zamanlı pilot kontrol girişlerinden dolayı, bu simülasyonların gerçek zamanlı çalışması gerekmektedir. Bu da gerçek zamanlı çalışmaya imkan veren yazılım ve donanım sistemlerini gerektirir.

Gerçek zamanlı donanım içeren yol taşıtı simülatörleri genel olarak üç farklı yapıda olabilir. Bunlardan birincisi, araç dinamiğinin ve görselliğin tek bir bilgisayar üzerinde çalıştırıldığı, bütün donanımın bu simülasyon sistemine entegre edildiği Şekil 6.1’deki yapıdır. İkincisi ise, araç dinamiğinin bir bilgisayar üzerinde, görsellik ve diğer etkenlerin de ayrı bir bilgisayar üzerinde çalıştırıldığı Şekil 6.2’deki simülatör yapısıdır. Son olarak çok daha gelişmiş simülatörlerde olduğu gibi, araç dinamiğinin

bir bilgisayar üzerinde, görselliğin ise daha büyük görüş açısı elde etmek için birden çok bilgisayar ve dolayısıyla birden çok ekran üzerinde sağlandığı Şekil 6.3'deki dağınık yapı yer almaktadır. Dağınık yapılı simülatörlerde bilgisayarlar arasındaki iletişim genellikle ethernet hattı üzerinden sağlanmaktadır.

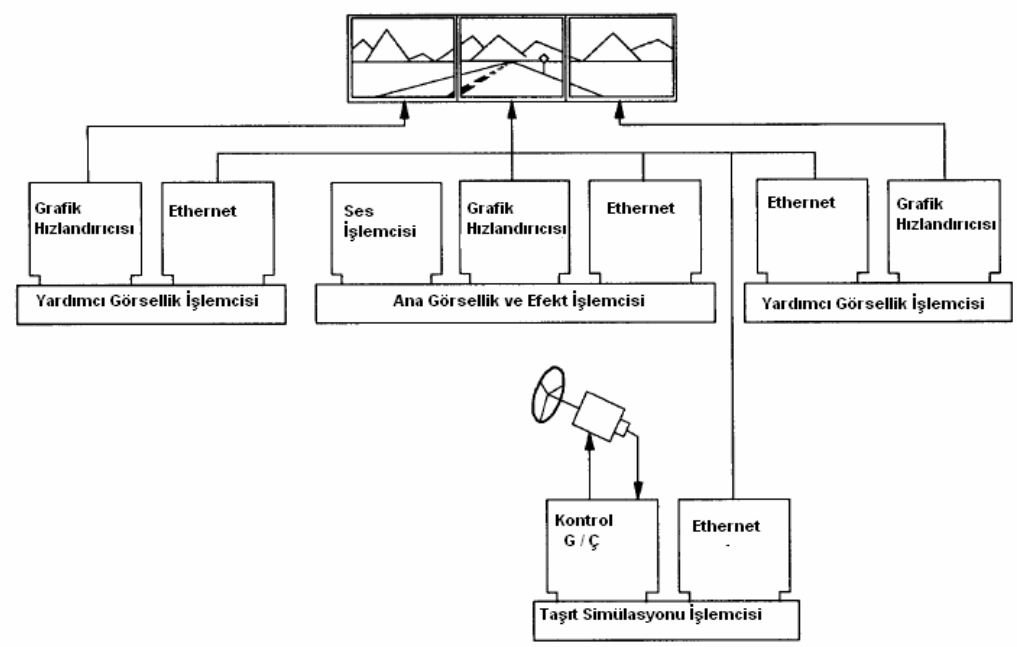


Şekil 6.1 : Tek bilgisayar üzerinde taşıt simülatörü [38].



Şekil 6.2 : İki bilgisayar temelli taşıt simülatörü [38].

Bu üç farklı sabit yol taşıtı simülatörü yapısı dışında, daha gerçekçi olması amacıyla simülatörlerin hareketli platformlar üzerine inşa edildiği yapılar da mevcuttur. Böyle bir platformda gerçek sürüş esnasında araçta oluşan açısal ve doğrusal eksenlerdeki hareketler elektrik motoru veya hidrolik tahrikli hareket mekanizmaları ile simülasyona dahil edilmektedir. Böylece sürücüye görsellik ve ses efektleri dışında gerçek araç kullanırken birebir üzerinde oluşan dinamik etkiler hissettirilir.



Şekil 6.3 : Dağınık yapılı yol taşıtı simülatörü [38].

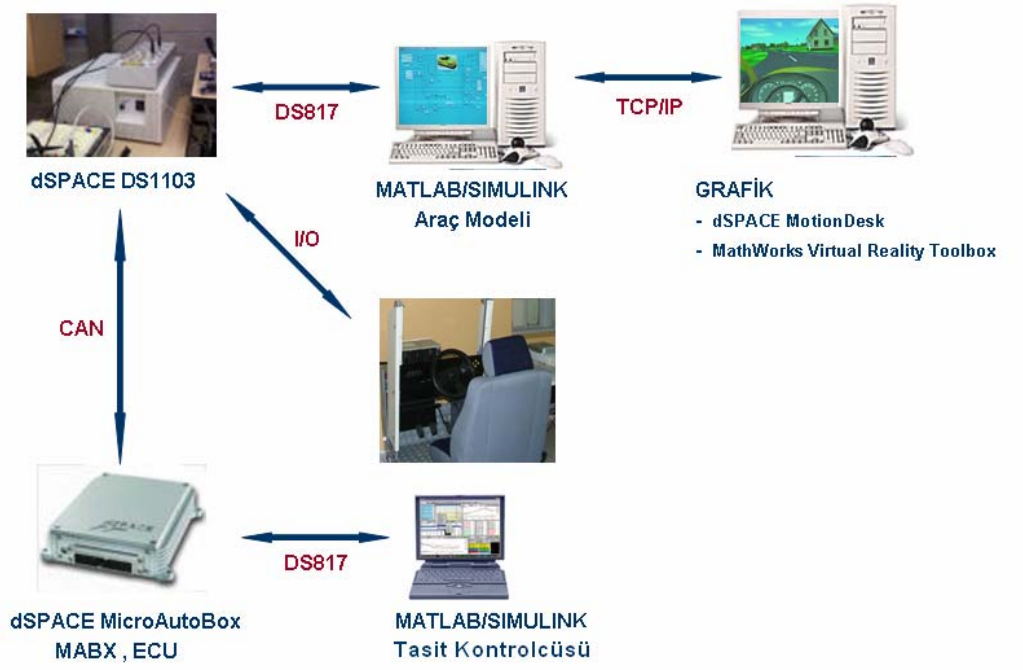
Şekil 6.4 : Vehicle Dynamics Expo 2004'de sergilenen simülatör [39]

6.1. Yol Taşıtı Simülatörü Tasarımı

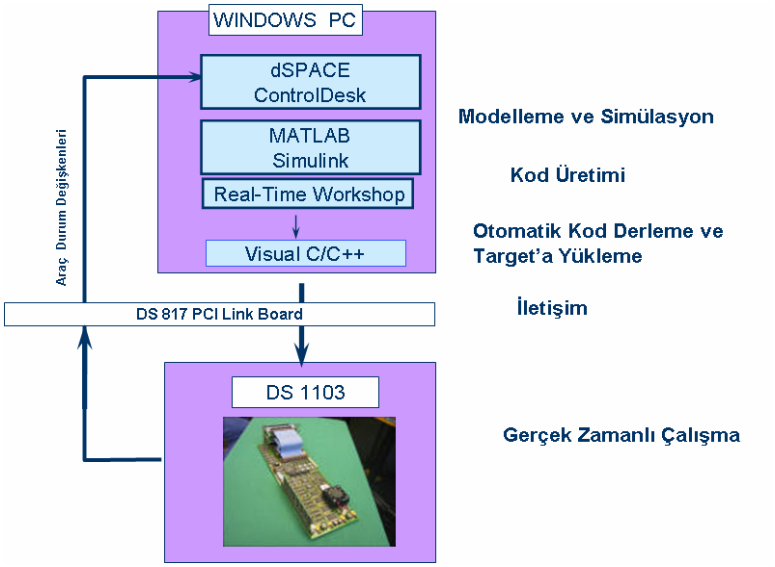
Bu tez kapsamında geliştirilen yol taşıtı simülatörü gerçek zamanlı bir simülatördür ve Şekil 6.5'de genel yapısı verilmiştir. Simülatör donanımı, sürücü girişleri, gaz, fren, debriyaj, direksiyon, vites donanımlarını içeren gerçek bir araç sürücü platformu; taşıt dinamiğinin gerçek zamanlı olarak çalıştırıldığı dSPACE DS1103 sistemi ve bu sistemin protiplendirilmesi için kullanılan bir bigisayar; taşıt kontrol

sistemlerinin gerçek zamanlı olarak çalıştırıldığı dSPACE MicroAutoBox ve bu sistemin prototiplendirilmesi için kullanılan bir bilgisayar; Ve görselliğin sağlandığı bir bilgisayardan oluşur.

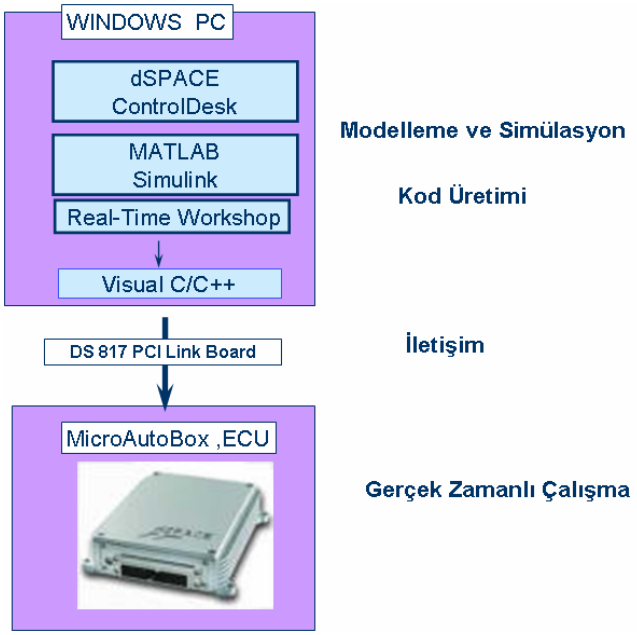
Simülatörde hızlı kontrolcü prototiplendirme ve gerçek zamanlı simülasyonlar için MATLAB / SIMULINK ve dSPACE sistemleri kullanılmaktadır. Sistem dinamik modelleri Simulink ortamında hızlı ve kolay bir şekilde geliştirilmektedir. MATLAB yazılımı ile entegre olan Real Time Workshop kullanılarak bu modellerden otomatik olarak C kodu üretilebilmektedir. Daha sonra üretilen bu kod derleyiciler ile derlenerek gerçek zamanlı olarak çalıştırılacağı sistem üzerinde prototiplendirilmektedir. Eğer Simulink ortamında gerekli ayarlamalar önceden yapıldıysa, daha önceki bölümlerde de bahsedildiği gibi bu kod üretimi ve hedef sisteme yüklenmesi işlemleri bir tıklama ile saniyeler içerisinde gerçekleştirilmektedir. Şekil 6.6 'da taşıt dinamiğinin modellenmesi ve gerçek zamanlı olarak çalıştırılacağı sistem üzerine prototiplendirilmesini gösteren akış diyagramı verilmiştir. Şekil 6.7'de ise geliştirilen ve bu simülatör üzerinde donanım içeren simülasyonlar ile doğruluğu ve işlerliği test edilecek olan taşıt kontrol sisteminin gerçek zamanlı olarak çalıştırılacağı sistem üzerinde prototiplendirilmesini gösteren akış diyagramı verilmiştir.



Şekil 6.5 : Yol taşıtı simülatörü genel yapısı.



Şekil 6.6 : Araç dinamiği prototiplendirme ve simülasyon akış diyagramı.



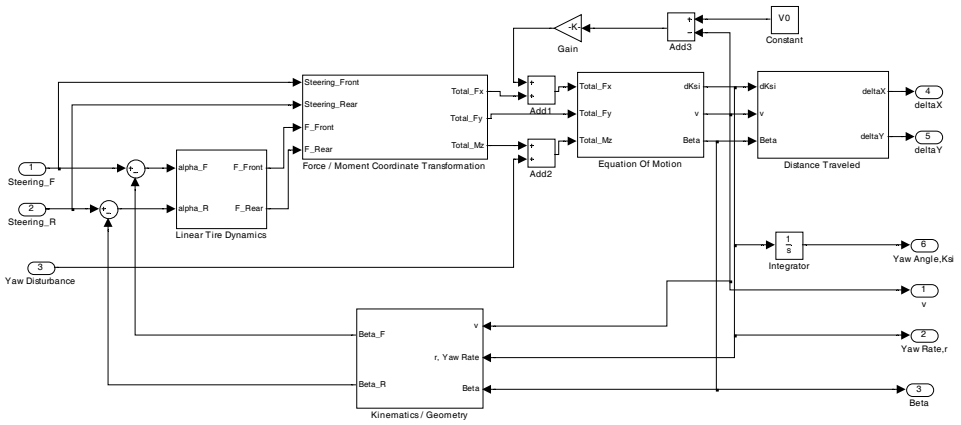
Şekil 6.7 : Taşıt kontrol sistemi prototiplendirme akış diyagramı.

Araç dinamiği DS1103 üzerine, tasarlanan araç kontrol sistemleri de MicroAutoBox üzerine prototiplendirildikten sonra gerçek zamanlı olarak çalıştırılır ve kontrol sisteminin testi için donanım içeren simülasyonlar gerçeklenir. Gerçek zamanlı olarak çalışan sistem simülasyonu ile kontrol sistemi arasındaki iletişim CAN prokolü ile sağlanmaktadır. CAN, 1 Megabit hız ile gerçek zamanlı veri iletişimine olanak sağlamaktadır. Ayrıca simülatörde CAN kullanılmasının diğer bir nedeni, bu protokolün otomotivde araçlar üzerinde yaygın olarak kullanılmasıdır. Bu sayede kontrol sistemi gerçek giriş / çıkış arayüzleri ile test edilmiş olmaktadır.

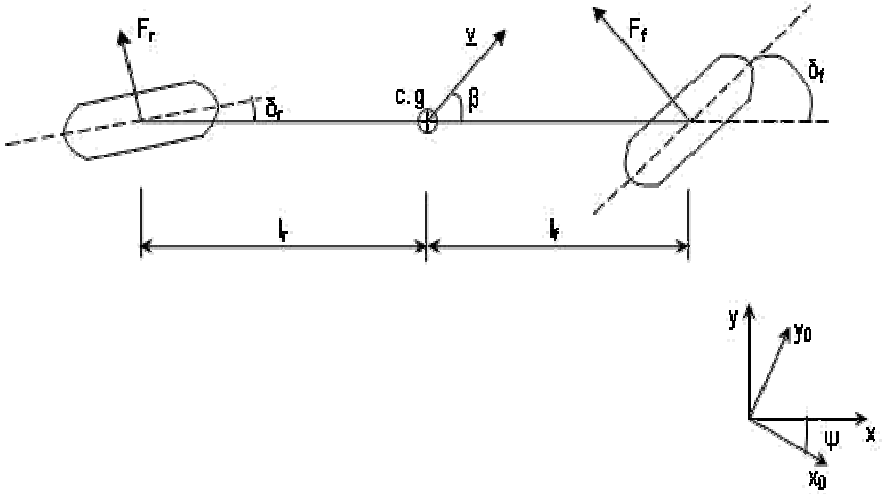
Simülatörde görsellik ise, dSPACE MotionDesk aracılığıyla sağlanmaktadır. MotionDesk üzerinde yeni görsel arayüzler geliştirilebilmekte ve ayrıca vrml uzantılı grafik animasyonlar da görselliğe dahil edilebilmektedir. Görsellik için gerekli araç değişkenleri, araç dinamiğinin gerçek zamanlı olarak çalıştığı DS1103 sistemi üzerinden DS817 kartı ile alınır ve görselliğin sağlandığı bilgisayara ethernet üzerinden gönderilir. Daha önce de söylendiği gibi uçtan uca ethernet 100 Mbit /sn veri iletişim hızı ile bu amaç için yeterlidir.

Simülatörde sistem simülasyonunda kullanılan taşıt dinamiği olarak Matlab/Simulink ortamında modellenmiş ve dSPACE sistemi üzerinde gerçek zamanlı olarak çalışabilen herhangi bir dinamik model kullanılabilir. Buradaki simülatörde şimdilik tek izli araç modelleri kullanılmış ve bu donanım içeren gerçek zamanlı simülatör yapısı savrulma engelleyici kontrolcülerin testlerinde kullanılmıştır. Geliştirilebilecek daha yüksek dereceli taşıt dinamiği modelleri ile istenilen araç kontrol sistemlerinin testleri de yapılabilir. Örnek olarak bu simülatör üzerinde, daha önce hızlı kontrolcü prototiplendirme bölümünde örnek olarak anlatılmış savrulma engelleyici kontrolcü, donanım içeren simülasyonlar ile başarılı bir şekilde test edilmiştir. Bu simülatör sayesinde geliştirilen kontrol sistemlerinin doğruluğu ve işlerliği saha uygulamalarına çıkmaksızın daha önce laboratuar ortamında test edilebilmektedir.

Aşağıda kullanılan tek izli araç modeline ilişkin dinamik denklemler verilmiş ve Şekil 6.8'de de MATLAB/SIMULINK ortamında modellenmiş tek izli araç modeli verilmiştir.



Şekil 6.8 : Tek izli araç modeli.



Şekil 6.9 : Tek izli araç modeli diyagramı [23].

Şekil 6.9'da tek izli araç modelinin bazı parametreleri gösterilmiştir. Ön ve arka tekerlekler için yanal kayma açıları,

$$\alpha_f = \delta_f - \beta_f$$
$$\alpha_r = \delta_r - \beta_r \qquad \textbf{(6.1)}$$

şeklindedir. δ_f ve δ_r sürücü tarafından tekerleğe verilen yönelim açıları ve β_f ve β_r de teker hızı ile şasi arasındaki açılardır. Yanal tekerlek kuvvetleri (6.2)'deki gibidir.

$$F_f = F_f(\alpha_f)$$
$$F_r = F_r(\alpha_r) \qquad \textbf{(6.2)}$$

Tekerlek modelinde hesaplanan tekerlek kuvvetleri şasi koordinat sistemine indirgenirse araç şasisine etkiyen kuvvet ve momentler (6.3)'deki gibi elde edilir.

$$\begin{bmatrix} F_x \\ F_y \\ M_z \end{bmatrix} = \begin{bmatrix} -\sin\delta_f & -\sin\delta_r \\ \cos\delta_f & \cos\delta_r \\ l_f\cos\delta_f & -l_r\cos\delta_r \end{bmatrix} \begin{bmatrix} F_f \\ F_r \end{bmatrix} \qquad \textbf{(6.3)}$$

Ve araç dinamik hareket denklemleri de (6.4)'deki gibidir. Burada β yanal kayma açısıdır.

$$\begin{bmatrix} mv(\dot{\beta}+\dot{\psi}) \\ m\dot{v} \\ J\ddot{\psi} \end{bmatrix} = \begin{bmatrix} -\sin\beta & \cos\beta & 0 \\ \cos\beta & \sin\beta & 0 \\ 0 & 0 & 0 \end{bmatrix} . \begin{bmatrix} F_x \\ F_y \\ M_z \end{bmatrix} \qquad \textbf{(6.4)}$$

Seyahat edilen yol da (6.5)'de verilen denklem takımı ile hesaplanır:

$$\Delta \mathrm{X} = \int_0^{t_1} v\cos(\beta + \psi)dt$$
$$\Delta Y = \int_0^{t_1} v\sin(\beta + \psi)dt \qquad \textbf{(6.5)}$$

Tek izli araç modeline motor modeli dahil edilmez ise Şekil 6.8'den de görüldüğü üzere (6.6)'daki gibi P-tipi bir kontrolcü ile F_x kuvveti değiştirilerek aracın sabit hızla hareketi sağlanmaktadır.

$$\Delta F_x = K(V_{istenen} - V_{gerçek}) \qquad \textbf{(6.6)}$$

7. SONUÇLAR VE ÖNERİLER

Bu çalışmada hızlı kontrolcü prototiplendirme teknikleri ve araçları, donanım içeren simülasyon sistemleri ve gerçek zamanlı simülasyon üzerinde durulmuştur. Gerek otomotiv sektöründe, gerek havacılık ve uzay sektöründe, gerek robotik sektöründe ve gerekse savunma sektöründeki mekatronik sistemlerde gelişen dijital teknolji ile birlikte gömülü kontrol sistemlerinin sayısı arttıkça bu sistemlerin geliştirilmesi sürecinde yukarıdaki simülasyon teknolojilerinin ve tasarım tekniklerinin ne kadar önemli olduğu, zaman ve maliyet bakımından geliştiriciye getirdiği çok büyük kazanımlar üzerinde duruldu. Dolayısıyla sistemlerin ve kontrolcülerin simülasyonları ve mikrokontrolörler üzerine prototiplendirilmesi için elle kod yazmaktansa, grafik arayüzler üzerinde model tabanlı olarak sistem modellerinin geliştirilmesine ve otomatik kod üretimine olanak veren hızlı kontrolcü prototiplendirme araçlarının kullanılması tavsiye edilmiştir. Ayrıca bu amaç için çok daha gelişmiş ve yaygın olarak kullanılan bir çok gömülü sistemi de kapsayan hızlı kontrolcü prototiplendirme araçlarının ve çok daha optimum boyutta kod üretebilen otomatik kod üreteçlerinin geliştirilmesi önerilmektedir. Geliştirilen kontrol sistemlerini, gerçek araçlar ve sistemler üzerinde bir takım saha uygulamalarında test etmektense, kontrol edilen sistemin dinamiğini yansıtan gerçekçi bir sistem modeli kurarak, laboratuar ortamında saha uygulamalarına çıkmaksızın gerçek zamanlı donanım içeren simülasyonlar ile test etmenin, hem bu sistemlerin ve testlerinin maliyeti, hem de zaman bakımından daha mantıklı olduğunu belirtmek gerekir.

Ayrıca bu tez kapsamında bir düşük maliyetli gerçek zamanlı pilotlu helikopter simülatörü ortamı geliştirilmiştir. Bu simülatör üzerinde dinamiği MATLAB/SIMULINK ile modellenmiş veya C ile kodlanmış herhangi bir helikopterin simülasyonu yapılabilir. Simülatörlerde en önemli bileşen şüphesiz görsel arayüzdür. Dolayısıyla gelecekte bu simülatör için çok daha güzel, profesyonel ve kapsamlı görsel arayüzler geliştirilebilir. Şu anda simülatörde kullanılan Bell205 helikopterine ait dinamik modele ilaveten değişik helikopter modellerini içeren bir kütüphane geliştirilebilir ve helikopter görsellikleri ile simülatöre dahil edilebilir. Gerçek

helikopter kokpiti ve aviyonikler üzerine geliştirilen bu simülatör ortamı aktarılabilir ve bu sayede çok daha profesyonel ve işlevsel bir helikopter simülatörü elde edilebilir.

Bunun dışında bir de gerçek zamanlı yol taşıtı simülatörü geliştirilmiştir. Bu araç simülatörü üzerinde değişik otomotiv uygulamaları ve geliştirilen kontrol sistemleri değişik senaryolar ile çok başarılı bir şekilde test edilebilir. Helikopter simülatöründe oduğu gibi, şu anda sadece çok sınırlı ve MotionDesk tabanlı olan görsel arayüz daha da geliştirilebilir, yeni yollar ve ortamlar oluşturulabilir. Bunun yanında simülatörde kullanılan araç modellerinin sayısı, değişik araçları da kapsayacak yüksek serbestlik dereceli dinamik modeller ile arttılabilir.

Hızlı kontrolcü prototiplendirme kapsamında model helikopter için geliştirilen stabilite artırıcı kontrolcüler üzerinde biraz daha çalışılabilir. Burada birbirleriyle etkileşim halinde olan eksenler birbirinden bağımsız olarak düşünülmüş ve her bir eksen için birer PID tipi kontrolcü ile yetinilmiştir. Ayrıca savrulma eksenine ilişkin gerçek helikopter platformu üzerinde de başarılı bir şekilde çalışılan bir kontrolcü geliştirilmiştir. Bundan sonraki süreçte değişik kontrol sistemlerinin tasarlanması üzerine çalışmalar da yapılacaktır. İlaveten tasarlanmış olan kontrolcülerin açık havada uçuş testleri de planlanmaktadır.

KAYNAKLAR

[1] **Shetty, D., Kolk, R.A., Kondo, J., Campana, C.,** 2001. A New Approach to Mechatronics System Design using Hardware in the Loop Simulation, *IEEE/ASME International Conference on Advanced Intelligent Mechtronics Proceedings*, Como, Italy.

[2] **Yoon, M., Lee, W., Sunwoo, M.,** 2005. Development and Implementation of Distributed Hardware-in-the-Loop Simulator For Automotive Engine Control Systems, *International Journal of Automotive Technology*, Vol.6, No.2.

[3] **Zanella, M., Koch, T., Scharfeld, F.,** 2001. Development and Structuring of Mechatronics Systems, Exemplified by the Modular Vehicle X-Mobile, *IEEE / ASME International Conference on Advanced Intelligent Mechatronics Proceedings* , Como, Italy.

[4] **Anakwa, K.N., Cohen, E., Naik, A., Carlton, D., Glen, D., Lopez, J.,** 2001. Tools for Rapid Prototyping of Embbeded Systems, *The 27th Annual Conference of the IEEE Industrial Electronics Society*.

[5] **Wu, X., Figueroa, H., Monti, A.,** 2004. Testing of Digital Controllers Using Real-Time Hardware-in-the-Loop Simulation, *35th Annual IEEE Power Electronics Specialists Conference*, Aachen.

[6] **Papini, M., Baracos, P.,** 2000. "Real-Time Simualtion, Control and HIL with COTS Computing Clusters", *American Institue of Aeronautics and Astronautics Journa*l.

[7] **Kimura, A. , Maeda, I.,** 1996. "Development of engine control system using real time simulator", *Proceedings of the 1996 IEEE International Symposium on CACSD*, Michigan, USA, 157–163.

[8] **Jang, S.H., Park, T.J., Han, C.S.,** 2003. A control of vehicle using Steer-By-Wire System with Hardware in the Loop Simulation System, *Proceedings of the 2003 IEEE / ASME International Conference on Advanced Intelligent Mechatronics (AIM 2003*).

[9] **Hanselmann, H.,** 1996. Automotive Control: From Concept to Experiment to Product , *Proceedings of the 1996 IEEE International Symposium on Computer – Aided Control Sytem Design*, Dearborn, MI.

[10] **Hanselmann, H.,** 1996. Hardware-in-the-Loop Simulation Testing and its Integration into a CACSD Toolset, *The IEEE International Symposium on Computer Aided Control System Design*, Dearborn.

[11] **Ptak, A. , Foundy, K.,** 1998. Real-Time Spacecraft Simulation and Hardware-In-The-Loop Testing, *IEEE Real Time Technology and Applications Symposium*, pp. 230-236.

[12] **Raman, S. , Sivashankar, N. , Milam, W. , Stuart, W. , Nabi, S.,** 1999. Design and Implementation of HIL Simulators for Powertrain Control System Development, *Proceedings of American Control Conference*, San Diego, California.

[13] **Lee, J.C., Suh, M.W.,** 1999. Hardware-In-The-Loop Simulator for ABS/TCS, *Proceedings of IEEE International Conference on Control Applications*, Hawai, USA.

[14] **Temeltas, H., Gokasan, M., Bogosyan, S., Kilic, A. ,** Hardware in the Loop Simulation of Robot Manipulators through Internet In Mechatronics Education, *IEEE Industrial Electronics Society 28th Annual Conference*, Vol. 4, pp. 2617-2622.

[15] **Oh, S.C.,** 2005. Evolution of Motor Characteristics for Hybrid Electric Vehicles Using the Hardware-in-the-Loop Concept., *IEEE Transactions on Vehicular Technology*, Vol.54, No.3, pp. 817-824.

[16] **Wagner, J.R., Keane, J.F.,** 1997. A Strategy to Verify Chassis Controller Software – Dynamics, Hardware, and Automation, *IEEE Transactions on Systems, Man, and Cybernetics – Part A : Systems and Humans*, Vol.27 No.4, pp. 480-493.

[17] **Jeon, J.W., Woo, G.A., Lee, K.C., Hwang, D.H., Kim, Y.J.,** 2003 Developments of ABS Controller for Aircraft with Real-Time HILS System", *IEEE Electrical Machines and Systems*, Vol.2, pp. 498-501.

[18] **Heo, S.J., Park, K., Ahn, H.S.,** 2003. Design and Implementation of Hardware-in-the-Loop Simulator for EHB Systems, *SICE Annual Conference in Fukui University*, Japonya, Vol.1, pp. 612-616.

[19] **Hagiwara, K., Terayama, S., Takeda, Y., Yoda, K., Suzuki, S.,** 2002. Development of Automatic Transmission Control System Using Hardware-In-The-Loop Simulation System, *JSAE Review* 23, pp. 52-59, Elsevier.

[20] **Deppe, M., Robrecht M., Zanella, M., Hardt, W.,** 2001. Rapid Prototyping of RealTime Control Laws for Complex Mechatronics Systems, *IEEE Rapid System Prototyping International Workshop*, pp. 188-193 .

[21] **Michael Vetsch**, 2003. Real-Time Testing, Rapid Controller Prototyping and Hardware-in-the-Loop (HIL), MIADC 2003, MIADC 2003.

[22] **Öztürk, S.,** 2004. Savrulma Engelleyici Kontrolün Gerçekçi Araç Modellerinde Denenmesi, *Yüksek Lisans Tezi*, İTÜ Fen Bilimleri Enstitüsü, İstanbul.

[23] **Yiğit,T.,** 2004. Araç Dinamiği Modellerinin Geliştirilmesi ve Savrulma-Devrilme Engelleyici Kontrolde Kullanımı, *Yüksek Lisans Tezi,* İTÜ Fen Bilimleri Enstitüsü, İstanbul.

[24] **Aksun Güvenç, B., Güvenç, L.,** 2002. The Limited Integrator Model Regulator and Its Use In Vehicle Steering Control, *Turkish Journal of Engineering & Environmental Sciences*, 26, 473 - 482.

[25] ControlDesk, DS1103, MicroAutoBox, www.dspace.de

[26] MATLAB, xPC Target, Real Time Workshop, www.mathworks.com

[27] ADvantage, www.adi.com

[28] **Ledin, Jim,** 2001. "Simualtion Engineering", CMP Books.

[29] **Park, T.J., Han, C.S., Lee, S.H.,** 2005. Development of the electronic control unit for the rack-actuating steer-by-wire using the hardware-in-the-loop simulation system, *Mechatronics* ,15, 899-918, Elsevier.

[30] **Çebi, A., Güvenç, L., Demirci, M., Karadeniz, C.K., Kanar, K., Güraslan, E.,** 2005. A Low Cost, Portable Engine Electronic Control Unit Hardware-In-The-Loop Test System, *Proceedings Of IEEE International Sysmposium on Industrial Electronics* , Vol.1, pp. 293-298, Dubrovnik, Croatia.

[31] **Çebi, A.,** 2005. Motor Kontrol Ünitesinin Gerçek Zamanlı Donanım İçeren Simülasyonlarda Testi, *Yüksek Lisans Tezi*, İTÜ Fen Bilimleri Ensitüsü, İstanbul.

[32] ALSIM Flight Simulator, www.alsim.com

[33] Bihrle Applied Research Inc. , www.bihrle.com

[34] ART, www.flightlab.com

[35] **Abdulhamitbilal, E.,** 2005. Rotorcraft's Mathematical Model Design And Simulation By Matlab – Simulink, *İTÜ ROTAM Teknik Raporu*, **R280-SC-R-04-001**, İstanbul.

[36] **Neshat,A.,** 1995. A High Order Simulation Model For The Bell205 Helicopter, *Master Tezi*, Ottawa-Carleton Instıtute For Mechanical and Aerospace Engineering.

[37] **Peter, D.T., Bruce, E.T., William, A.D., Robert, T.N.C.,** 1981. A Mathematical Model Of A Single main Rotor Helicopter For Piloted Simulation, *NASA Technical Report*, USA.

[38] **Allen, R.W., Rosenthal, T.J., Aponso, B.L., Klyde, D.H., Anderson, F.G., Hogue, J.R., Chrstos, P.J.,** 1998. A Low Cost PC Based Driving Simulator fo Prototyping and Hardware-In-The-Loop Applcations, SAE.

[39] Simulation and Modeling, 2003, www.vehicledynamics-expo.com**.**

[40] **Demirci, M., Güvenç, L. ,** 2005. Düşük Maliyetli Gerçek Zamanlı Pilotlu Helikopter Simülatörü Ortamı Geliştirilmesi, *TOK'05*, İstanbul Teknik Üniversitesi, İstanbul.

[41] **Abdulhamitbilal, E., Caferov, E., Güvenç, L.,** 2005. Doğrusal Olmayan Helikopter Modeli ve Trim Analizi : Asılı Hal, *TOK'05*, İstanbul Teknik Üniversitesi, İstanbul.

EK – A dSPACE DS1103

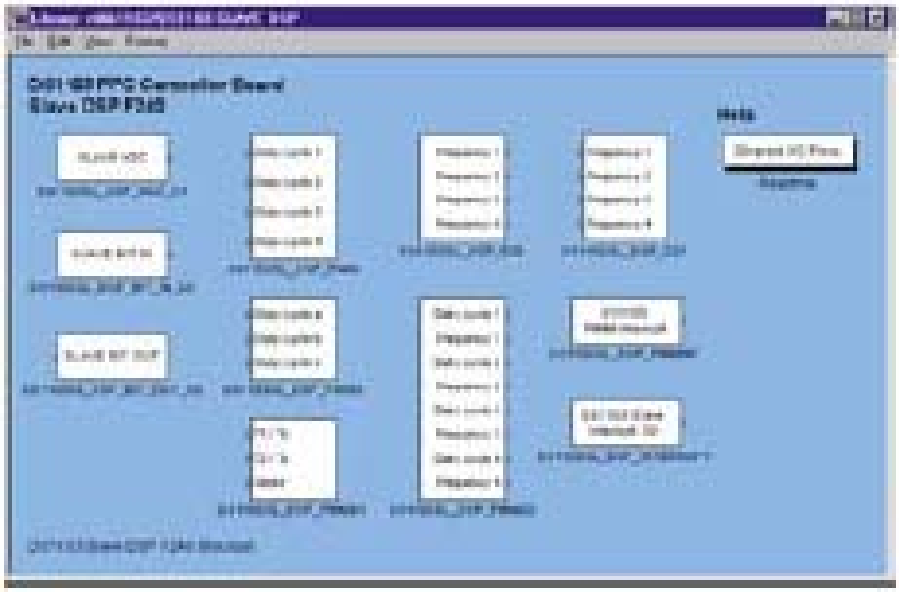
Şekil A.1 : DS1103 donanımı.

- **Bir çok G/Ç opsiyonuna sahip bir sistem**
- **400MHz de çalışan PowerPC 604e içerir**
- **Üzerinde yardımcı kontrolör olarak ekstra G/Ç opsiyonlarına sahip DSP TMS320F240 bulunur**
- **Enkoder, CAN ve seri haberleşme girişleri bulunur**

DS1103 hızlı kontrolcü prototiplendirme uygulamalarında kullanılan oldukça güçlü bir sistemdir. Ana kontrolör olarak kayar noktalı matematiksel işlemlere de olanak veren bir PowerPC 604e bulunur.

Üzerinde 36 adet analog giriş ve 8 adet de analog çıkış bulunur. Ayrıca Dijital G/Ç, PWM üreteci yer alır. G/Ç seçeneklerini artırmak için Texas Instruments'e ait TMS320F240 DSP içerir. Infineon firmasına ait bir CAN kontrolör içerir. G/Ç opsiyonlarının çeşitliliği sayesinde robotik uygulamalarından otomotiv uygulamalarına kadar bir çok alanda yaygın bir şekilde kullanılmaktadır.

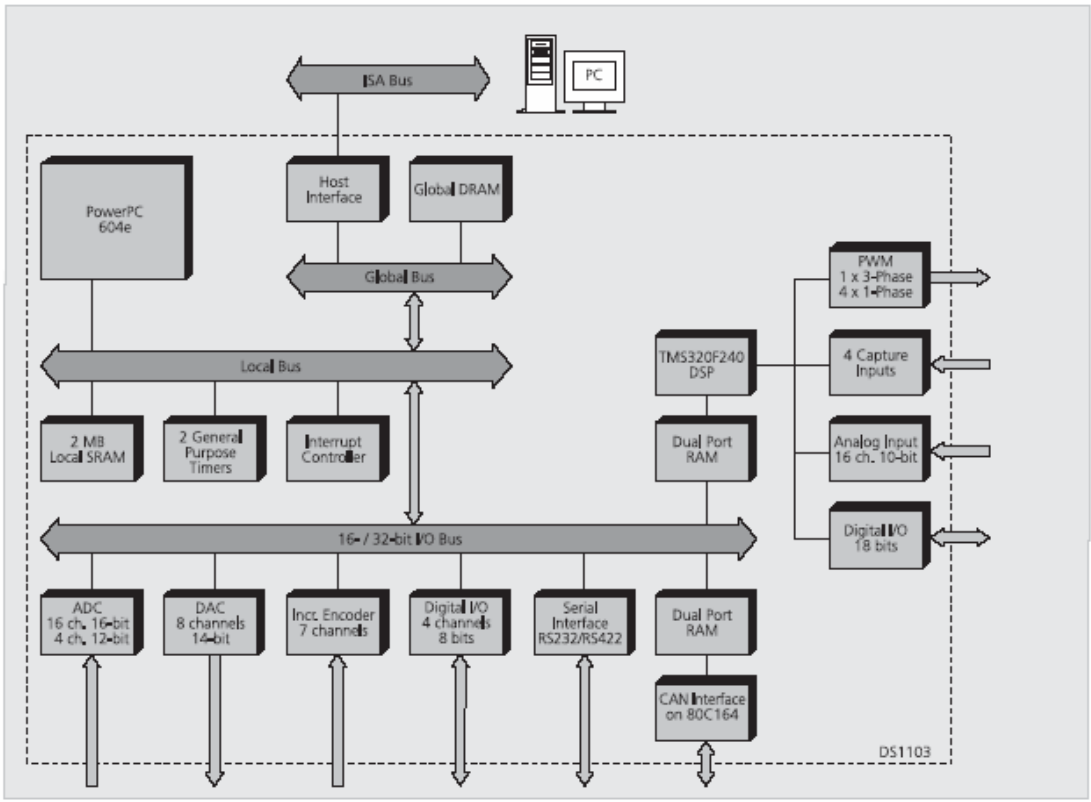
Matlab/Simulink üzerinde dSPACE Real Time Interface sayesinde hızlı kontrolcü prototiplendirme ile kullanılabilmektedir. Real Time Workshop otomatik kod üreteci ile de bu sistem için otomatik olarak C kodu üretilebilmektedir. PC ile DS817 PCI kartı ile haberleştirilmektedir. dSPACE ControlDesk yazılımı ile veri toplamadan sinyallerin görsel arayüzler üzerinde gösterilmesine kadar bir çok işlev yerine getirilebilmektedir.



Şekil A.2 : DS1103 Simulink RTI.

Uygulama Alanları

- Motor Kontrol
- Robotik
- Kontrollü Frekans Dönüştürücüleri
- Konumlandırma sistemleri ve adım motorlar
- 6 eksenli robotlara kadar robot kontrol sistemlerinin tasarımı
- Otomotiv kontrol sistemlerinin hızlı kontrolcü prototiplendirilmesi
- Servohidrolik sistemler
- Aktif titreşim kontrol
- Elektrikli Aktütörler



Şekil A.3 : DS1103 donanım diyagramı.

Ana İşlemci

- PowerPC 604e / 400 MHz
- 3 tamsayı, 1 kayar noktalı işlem ünitesi
- 2 adet işlemci üzerinde zamanlayıcı
- 32 KB komut, 32 KB da veri için önbellek

Zamanlayıcı

- Genel amaçlı 2 zamanlayıcı

Hafıza

- Program hafıza olarak dahili SRAM 2MB hafıza
- Verilerin saklanması ve ana bilgisayar ile haberleşme için 128 MB global DRAM hafıza

Kesme Kontrol Ünitesi

- Ana bilgisayar, CAN, yardımcı DSP, seri haberleşme, artımlı enkoder ve 4 tane de harici kesme mevcuttur
- PWM senkronizasyon girişi

Analog Giriş

- Her biri 4 kanal içeren 4µs örnekleme zamanına sahip 16-bit 4 adet ADC ünitesi ve 4 adet de 12-bit ADC
- +/- 10V voltaj aralığı

Analog Çıkış

- 8 adet 14- bit çıkış
- 5µs yerleşme zamanı
- +/- 10V voltaj aralığı

Artımlı Enkoder Girişi

- 6 kanal dijital giriş
- Dijital gürültü filtreleme
- Max. 1.65 MHz giriş frekansı,
- TC3005H kontrolörü ile analog giriş

Dijital G/Ç

- 4 adet 8-bit dijital G/Ç portu
- Birbirinden bağımsız olarak programlanabilir G/Ç kanallar

Seri Arayüz

- RS232 ve RS422 alıcı/gönderici desteği
- 1Mbaud'a kadar haberleşme hızı

CAN Arayüzü

- Infineon 80C164 mikrokontrolörü tabanlı max. 1MBaud haberleşme hızını destekleyen CAN desteği

Yardımcı DSP Bileşenleri

- Özellikle motor kontrolü için tasarlanmış TI TMS 320F240, 20MHz işlemci
- 3-fazlı PWM çıkışları ve 4 adet tekli PWM çıkışı
- Her biri 8 adet analog giriş içeren 6.6μs örnekleme zamanına sahip 10-bit 2 adet düşük çözünürlüklü ADC
- Her biri birbirinden bağımsız olarak programlanabilen TTL seviyeli 18-bit dijital G/Ç

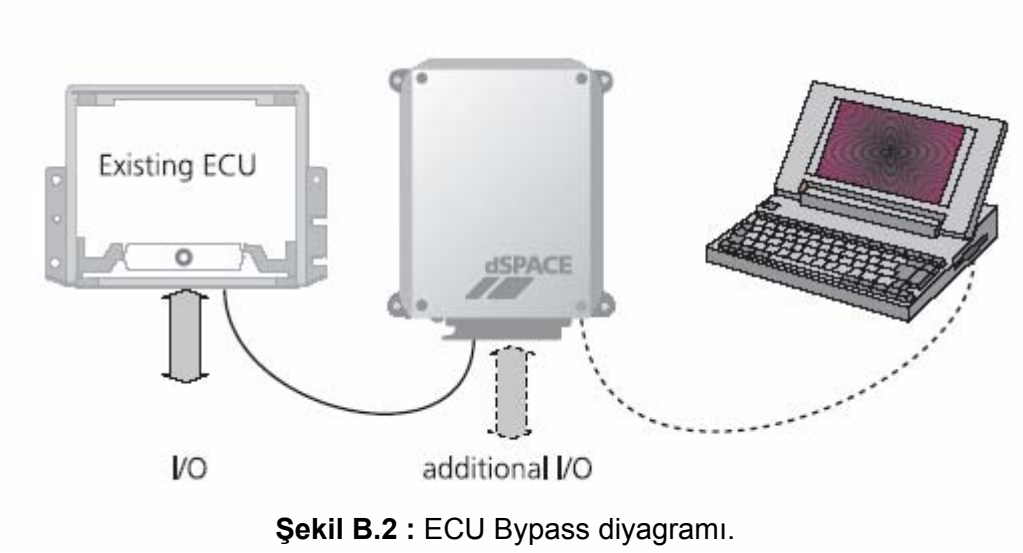
EK – B dSPACE MicroAutoBox (MABX)

Şekil B.1 : DSPACE MicroAutoBox.

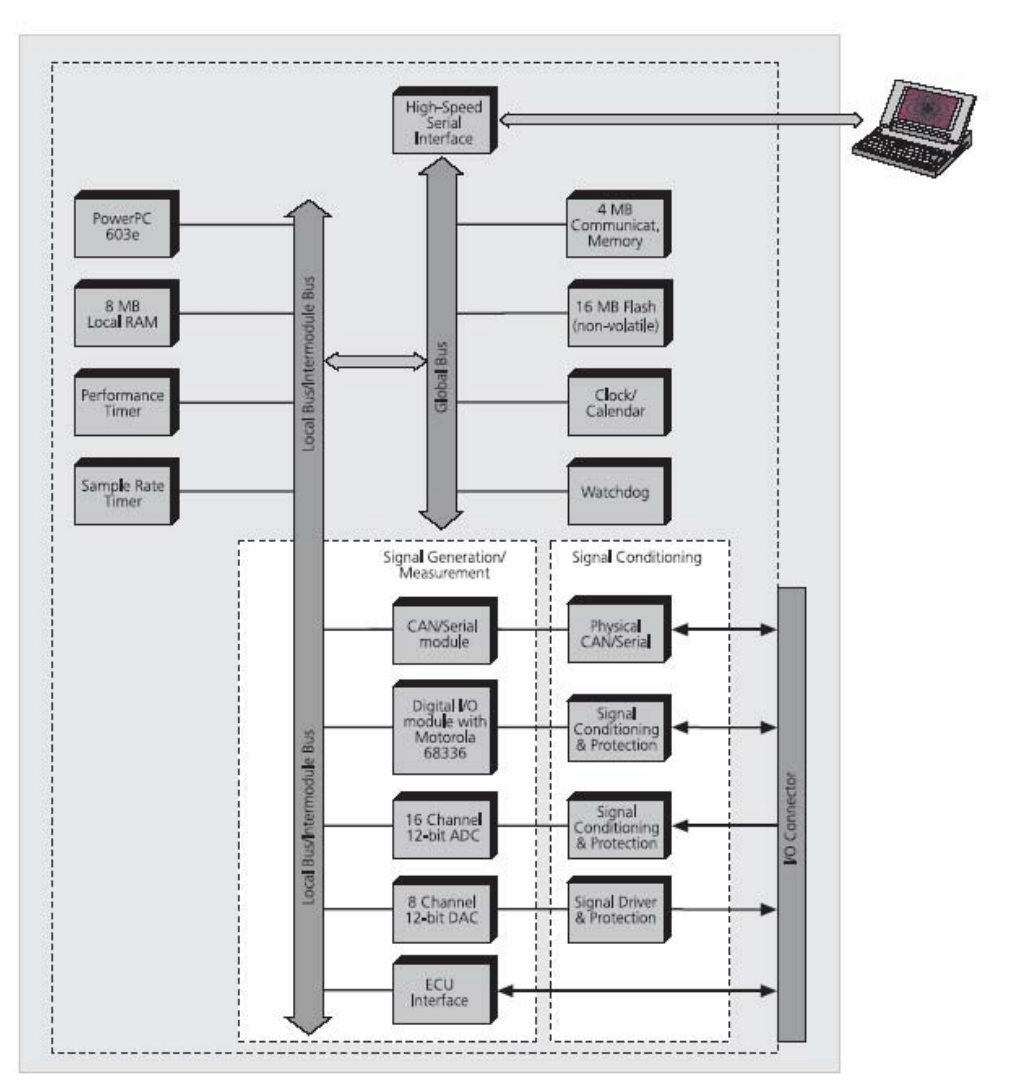
MABX, özellikle otomotiv uygulamalarında gerçek otomotiv ECU'su benzeri amaçlarla kullanılmak üzere geliştirilmiş ve dSPACE gerçek zamanlı sistemlerinin neredeyse bütün işlevlerini bir araya toplamış olan portatif bir sistemdir. Üzerinde otomotivde yaygın olarak kullanılan CAN, LIN ve Flexray haberleşme protokolleri desteklenir.

MABX üzerinde kayar noktalı işlemleri de destekleyen PowerPC 603e bulunur. PC ile PCMCIA ya da PCI DS817 kartları ile haberleşmektedir. dSPACE Real Time Interface sayesinde Matlab/Simulink tabanlı hızlı kontrolcü prototiplendirme yapılabilmektedir. PC ile iletişim sadece program yükleme veya flash hafıza üzerinde kaydedilen verilerin bilgisayara aktarılması için kullanılmaktadır. Bunun dışında tıpkı gerçek araç ECU'sunda olduğu gibi uzun süre tek başına çalıştırılabilir. Eğer program flash hafızaya yüklendiyse PC ile iletişim kurmaksızın MABX'e güç verilmesi ile birlikte çalışmaya başar. Üzerinde 13MB flash hafıza bulundurur. Bu sayede Flight Recorder ile uzun süreli veri kaydı yapılabilmektedir.

Ayrıca gerçek araç ECU'sunun kontrol algoritmalarının kısmen bypass edilerek yerine henüz tasarlanmış kontrolcülerin test edilmesine olanak veren Bypass tabanlı prototiplendirmeyi de destekler.



Şekil B.2 : ECU Bypass diyagramı.



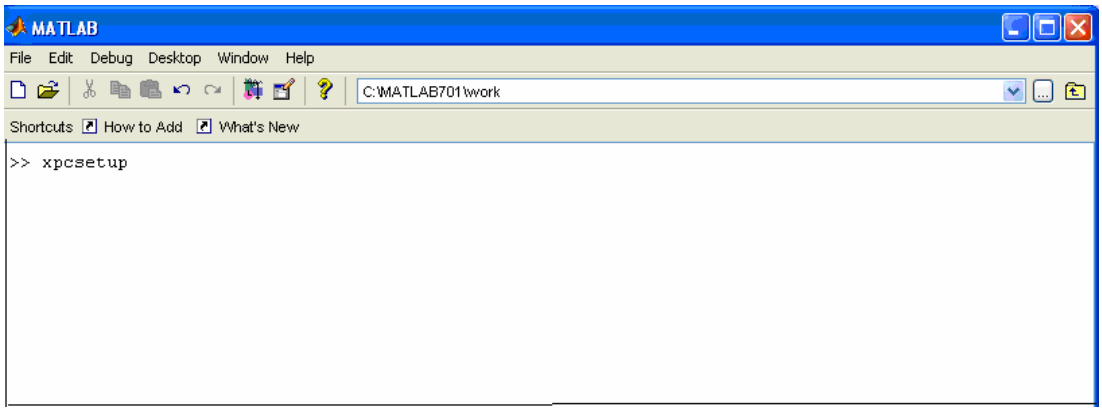
Şekil B.3 : MABX donanım diyagramı.

Tablo B.1 : Giriş/Çıkış Portları ve Veri Haberleşme Arayüzleri.

	MicroAutoBox 1401/1501	MicroAutoBox 1401/1504	MicroAutoBox 1401/1505/1506
CAN Interface	Dual CAN interface In total 2 CAN channels	Two dual CAN interfaces In total 4 CAN channels	
Serial Interface	Serial interface based on CAN processor: 1 x RS232 interface 1 x serial interface usable as K/L-Line or LIN interface	Serial interface based on CAN processor: 1 x RS232 interface 1 x serial interface usable as K/L-Line or LIN interface	Serial interface based on CAN processor: 2 x RS232 interface 2 x serial interface usable as K/L-Line or LIN interface
ECU Interface	Dual-port memory interface, 16 K x 16-bit DPRAM	–	Dual-port memory interface, 16 K x 16-bit DPRAM
FlexRay Interface	–	–	2 slots for FlexRay controllers; currently one controller (2 channels) is supported
Analog Input	16 12-bit channels, 4 to 1 multiplexed, simultaneous sample & hold, 6.7 µs conversion time for all channels*) 0 – 5 V input voltage range	24 12-bit channels, 4 to 1 multiplexed, simultaneous sample & hold, 6.7 µs conversion time for all channels*) 0 – 5 V input voltage range	16 12-bit channels, 4 to 1 multiplexed, simultaneous sample & hold, 6.7 µs conversion time for all channels*) 0 – 5 V input voltage range
Analog Output	8 12-bit channels 0 – 4.5 V output voltage range 5 mA max. sink/source current	–	8 12-bit channels 0 – 4.5 V output voltage range 5 mA max. sink/source current
Digital I/O	Digital I/O on 68336 slave processor, 20MHz, with Time Processor Unit (TPU) 16 discrete inputs 4 shared inputs for frequency or PWM 16 TPU channels 16 shared discrete inputs/outputs 10 discrete outputs, 5 mA output current 4 PWM outputs, 2.5 Hz – 100 kHz PWM frequency, duty cycle 0 – 100%, up to 16-bit resolution I/O software support for different applications		

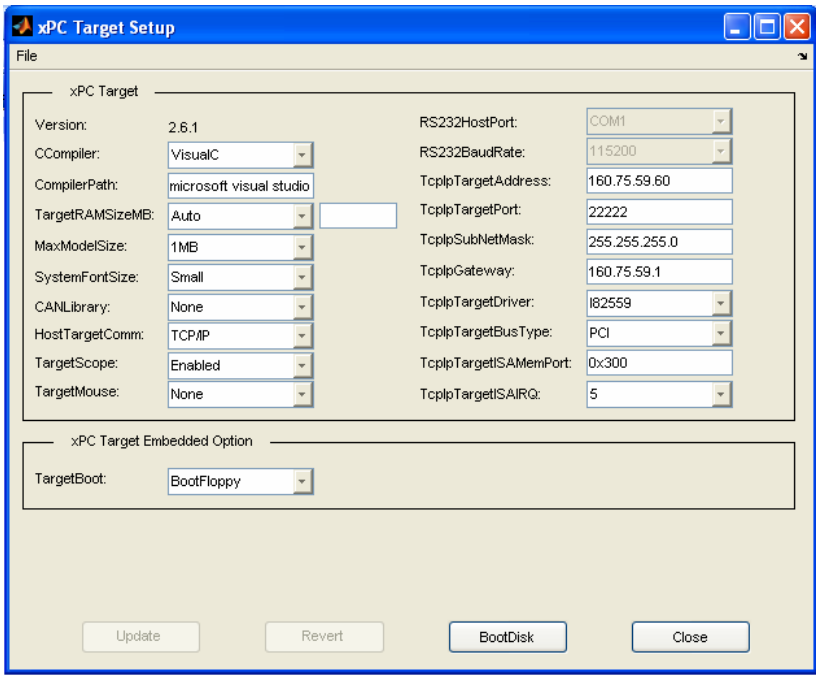
EK – C Matlab/Simulink Üzerinde xPC Target Ayarları

Bölüm 5'te helikopter simülatörü'nde de kullanılan xPC Target gerçek zamanlı sisteminin bahsedilen şekilde hızlı kontrolcü prototiplendirme ve donanım içeren simülasyon testlerinde kullanılabilmesi için aşağıdaki ayarların Matlab/Simulink ortamında önceden yapılması gerekir.



Şekil C.1 : Matlab komut satırı ve *xpcsetup* komutunun çalıştırılması.

Matlab komut satırında *xpcsetup* yazılarak aşağıdaki xPC Target Setup ekranından bu ayarlar yapılabilir.

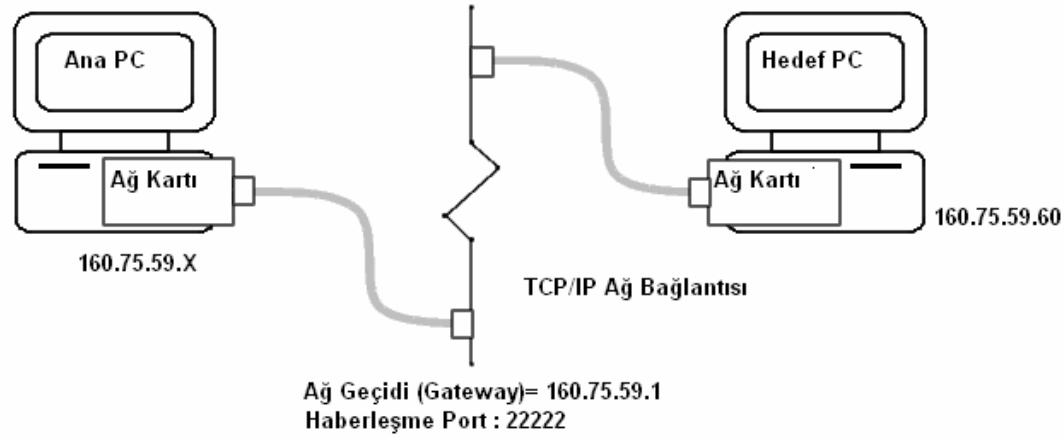


Şekil C.2 : xPC Target Setup menüsü.

xPC Target'da ana bilgisayar ile hedef bilgisayar arasındaki iletişim TCP/IP veya RS232 ile sağlanabilmektedir. Tezde TCP/IP kullanıldığından burada ayarlar buna göre yapılmıştır. Ayrıca burada ana ve hedef sistemler arasındaki fiziksel bağlantının birebir çaprazlama (Peer-to-Peer with cross cable) ile ya da internet/intranet kurulduğu ve hazır olduğu kabul edilmektedir. Bu tezde de olduğu gibi yaygın olarak kullanıldığı şekliyle eğer birebir çapraz kablolama ile bu fiziksel ağ kurulduysa hedef bilgisayara IP atanırken dikkatli olunması ve IP'nin aynı alt ağ'da olmasına dikkat edilmelidir. Aksi takdirde birebir bağlantıda DNS mevcut olmadığından iki sistem birbirini göremeyecektir.

xPC Target Setup ekranından da görüldüğü gibi derleyici olarak Microsoft Visual C ya da Watcom C Compiler'dan birinin ana bilgisayarda kurulu olması gerekir. Bu, Simulink ortamında tasarlanan modellerden xPC Target hedef sistemi için otomatik kod üretildikten sonra kodun derlenmesi için gereklidir.

Eğer hedef bilgisayar olarak xPC TargetBox gibi bir çözüm kullanmak yerine herhangi bir bilgisayar kullanılırsa Ethernet kartı uygunluğu problemi oluşmaktadır. Böyle bir durumda hedef sistem ekranı görüntülenebilirse eğer, "System Halted" hatası görülecektir. Dolayısıyla iletişim TCP/IP ile sağlanıyorsa desteklenen ethernet kartlarından birisinin kullanılması gerekir. SMC 1208BTCard veya Intel Pro/100S şu anda desteklenen PCI Ethernet kartlarıdır.

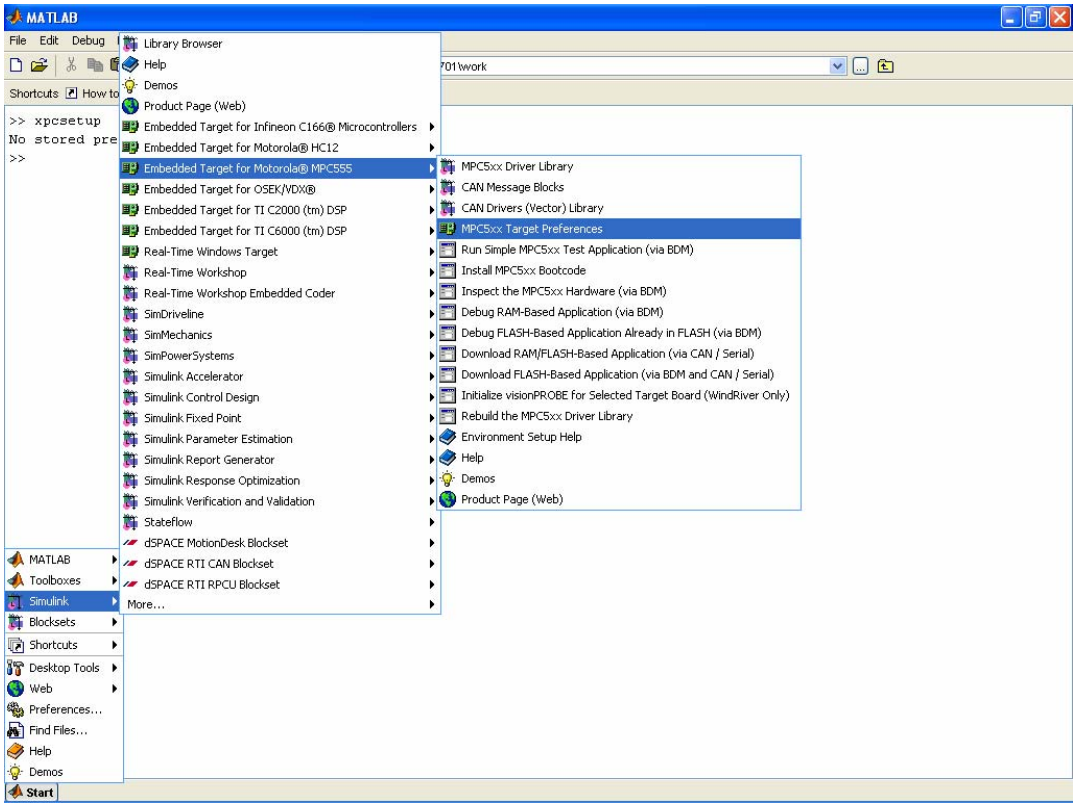


Şekil C.3 : xPC Target işlevsel diyagramı.

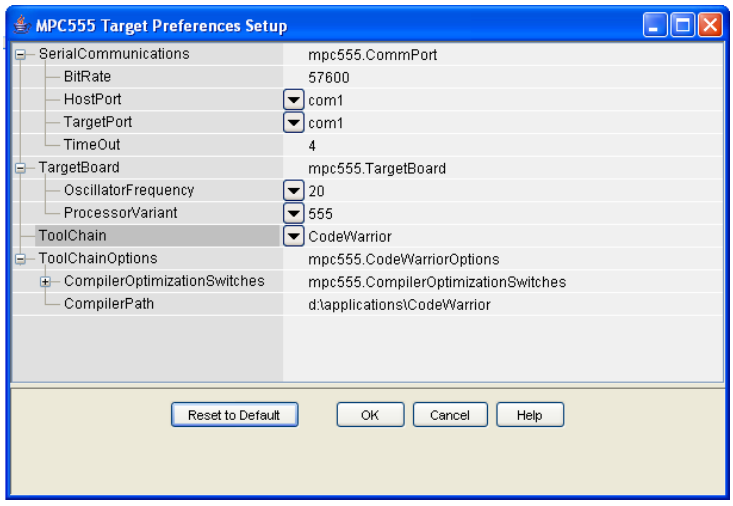
Bu ayarlar ile oluşturulmuş disket ile Hedef PC başlatılırsa ve yukarıda belirtilen uyarılar da dikkate alınırsa gerekli sistem başarılı bir şekilde kurulabilir.

EK – D Matlab/Simulink Üzerinde *Embedded Target For MPC555* Ayarları

Tezde model helikopter için kontrolcü tasarlama bölümünde bahsedilen gerçek zamanlı gömülü sistemde hızlı kontrolcü prototiplendirme için bir takım ayarların önceden yapılması gerekmektedir. Embedded Target For MPC555 sistemi derleyici olarak Diab Compiler ya da Metrowerks CodeWarrior programlarını desteklemektedir. Bu tezde Metrowerks CodeWarrior kullanıldığından burada buna ilişkin ayarlar verilecektir. Bu ayarların yapıldığı *MPC555 Target Preferences Setup* ekranı Matlab **Start** menüsünden açılabilir. Burada seri bağlantının yapılacağı seri port ve haberleşme hızı, Metrowerks CodeWarrior programının kurulu olduğu yer belirtilmelidir.

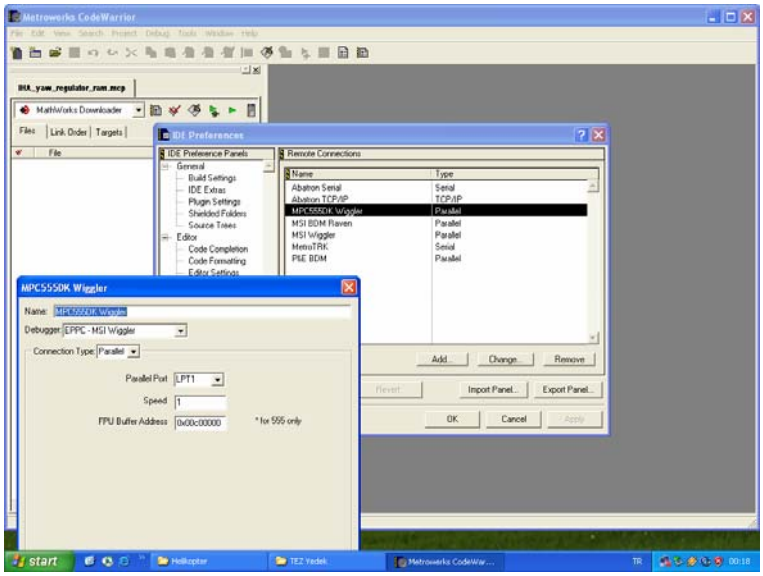


Şekil D.1 : Embedded Target For MPC555 yapılandırma ekranı.



Şekil D.2 : MPC555 Target Preferences Setup menüsü.

Simulink modelinden MPC555 tabanlı gömülü sistem için üretilen kod Metrowerks CodeWarrior ile derlendikten sonra paralel port aracılığıyla yüklenmektedir. Dolayısıyla Metrowerks CodeWarrior yazılımında da *IDE Preferences* menüsünden bu bağlantı tanımının yapılması ve gerekli yükleme arayüz yapılandırma kodunun belirtilmesi gerekmektedir. Kaldı ki MPC555 tabanlı gömülü sistem, hızlı kontrolcü prototiplendirme kapsamında kullanılmadan önce Metrowerks CodeWarrior programı ile birlikte gelen demo uygulamalarla çalışırlığı test edilmelidir. Özellikle problem oluşturan, gözden kaçan ve bilinmediği takdirde sistemin çalıştırılamamasından dolayı çok uğraştıran konulardan biri de BIOS'dan Paralel Port haberleşme modu olarak standart olan '*Bidirectional*' yerine 'EPP' seçilmesi gerektiğidir.



Şekil D.3 : Metrowerks CodeWarrior yapılandırma ekranı.

EK – E Mikrosoft Force Feedback Joystick'in USB Üzerinden Okunması İçin C++ İle Yazılmış Program Kodları

JoystickSource.cpp

```
#include "JoystickSource.h"
#include <math.h>


//-----------------------------------------------------
// InitJoystick()
// Initializes Joystick...
//-------------------------------------------------------------
__declspec(dllexport) HRESULT InitJoystick()
{
        HRESULT hr;

        //Gets a pointer to IDirectInput Interface and Creates a DInput Object...
        if( FAILED( hr = DirectInput8Create( GetModuleHandle(NULL),
DIRECTINPUT_VERSION,
                        IID_IDirectInput8, (VOID**)&g_pDI, NULL ) ) ){
      return hr;
      }

   // Looks for Joystick attached to computer and supports Force-Feedback...

   if( FAILED( hr = g_pDI->EnumDevices( DI8DEVTYPE_JOYSTICK,
                        EnumJoystickCallback,
                        NULL, DIEDFL_FORCEFEEDBACK |
DIEDFL_ATTACHEDONLY ) ) ){
         return hr;

      }


   //Makes sure There is a Joystick with specified properties just before...
   if(g_pJoystick==NULL)
        return E_FAIL;


   // Sets data format to a predefined format...(c_dfDIJoystick2)
   if( FAILED( hr = g_pJoystick->SetDataFormat( &c_dfDIJoystick2 ) ) )
                return S_FALSE;


  // Sets the cooperative level to let DInput know how this device should
  // interact with the system and with other applications.
```

```
        if( FAILED( hr = g_pJoystick->SetCooperativeLevel(GetForegroundWindow(),
                                        DISCL_EXCLUSIVE | DISCL_BACKGROUND ) ) )
{
                return hr;

        }


// Breaks Auto-Center Properties..

DIPROPDWORD dipdw;
dipdw.diph.dwSize       = sizeof(DIPROPDWORD);
dipdw.diph.dwHeaderSize = sizeof(DIPROPHEADER);
dipdw.diph.dwObj        = 0;
dipdw.diph.dwHow        = DIPH_DEVICE;
dipdw.dwData            = FALSE;

if( FAILED( hr = g_pJoystick->SetProperty( DIPROP_AUTOCENTER, &dipdw.diph ) ) )
        {
    return hr;
        }

// Enumerates Joystick's Objects...Sets Range properties for found axes...

 if ( FAILED( hr = g_pJoystick->EnumObjects( EnumAxesCallback,
                                        (VOID*)&g_NumForceFeedbackAxis, DIDFT_ALL ) ) ){
         return hr;
         }


// Initializes Force Feedback Effect...

DWORD          rgdwAxes[2]     = { DIJOFS_X,DIJOFS_Y};
 LONG          rglDirection[2] = { 0,0};
 DICONSTANTFORCE cf             = { 0 };

 DIEFFECT eff;
 ZeroMemory( &eff, sizeof(eff) );

 eff.dwSize                  = sizeof(DIEFFECT);
 eff.dwFlags                 = DIEFF_CARTESIAN | DIEFF_OBJECTOFFSETS;
 eff.dwDuration              = INFINITE;
 eff.dwSamplePeriod          = 0;
 eff.dwGain                  = DI_FFNOMINALMAX;
 eff.dwTriggerButton         = DIEB_NOTRIGGER;
 eff.dwTriggerRepeatInterval = 0;
 eff.cAxes                   = g_NumForceFeedbackAxis;
 eff.rgdwAxes                = rgdwAxes;
 eff.rglDirection            = rglDirection;
```

```
    eff.lpEnvelope              = 0;
    eff.cbTypeSpecificParams    = sizeof(DICONSTANTFORCE);
    eff.lpvTypeSpecificParams   = &cf;
    eff.dwStartDelay            = 0;

//  Acquires Joystick...

          hr = g_pJoystick->Acquire();

          if( FAILED( hr))
              return hr;


          //Creates Effect...
    if( FAILED( hr = g_pJoystick->CreateEffect( GUID_ConstantForce,&eff, &g_pEffect,
NULL ) ) )
      {
       return hr;
      }


   //Makes sure The Effect was created...
    if( g_pEffect==NULL)
        return E_FAIL;


          return S_OK;
}

//-----------------------------------------------------------------------------
//EnumJoystickCallback Function...
// Called when found a Joystick...And Creates an interface with it...So we can use it...
//-----------------------------------------------------------------------------
BOOL CALLBACK EnumJoystickCallback( const DIDEVICEINSTANCE* pdidInstance,
                                    VOID* pContext )
{
    HRESULT hr;

    // Obtains an interface to Joystick...
    hr = g_pDI->CreateDevice( pdidInstance->guidInstance, &g_pJoystick, NULL );

    // Recursive calls till it will proceed...
    if( FAILED(hr) )
        return DIENUM_CONTINUE;

    // Stop searching Joystick...We found a Joystick that we want...
    return DIENUM_STOP;
}
```

```
//-----------------------------------------------------------------------------
// EnumAxesCallback Function...
// Callback function for objects (axes,buttons,etc..) on Joystick.
//Enables user interface elements for objects an scales axes...
//-----------------------------------------------------------------------------
BOOL CALLBACK EnumAxesCallback( const DIDEVICEOBJECTINSTANCE* pdidoi,
                        VOID* pContext )
{

   DWORD* pdwNumForceFeedbackAxis = (DWORD*) pContext;

   if( (pdidoi->dwFlags & DIDOI_FFACTUATOR) != 0 )
      (*pdwNumForceFeedbackAxis)++;

   DIPROPRANGE diprg;
   diprg.diph.dwSize       = sizeof(DIPROPRANGE);
   diprg.diph.dwHeaderSize = sizeof(DIPROPHEADER);
   diprg.diph.dwHow        = DIPH_BYID;
   diprg.diph.dwObj        = pdidoi->dwType;
   diprg.lMin              = -1000;
   diprg.lMax              = +1000;



         // Sets ranges of Objects...
         if( FAILED( g_pJoystick->SetProperty( DIPROP_RANGE, &diprg.diph ) ) )
                  return DIENUM_STOP;

         return DIENUM_CONTINUE;
}

//-----------------------------------------------------------------------------
// UpdateJoystick()
// Gets Joystick's State...
//-----------------------------------------------------------------------------
__declspec(dllexport) HRESULT UpdateJoystick()
{
   HRESULT     hr;

   if(g_pJoystick==NULL )
      return E_FAIL;


  // Polls the Joystick to read its'state...
   hr = g_pJoystick->Poll();

         if( FAILED(hr) )
   {

      hr = g_pJoystick->Acquire();
```

```
        while( hr == DIERR_INPUTLOST )
            hr = g_pJoystick->Acquire();

                    return S_OK;
    }

   // Gets Joystick's state...
    if( FAILED( hr = g_pJoystick->GetDeviceState( sizeof(DIJOYSTATE2), &js ) ) )
    {return hr;}

            return S_OK;

}

//-----------------------------------------------------------------------------
// SetAndApplyEffect()
// Downloads Effect to the Joystick and Initializes it...
//-----------------------------------------------------------------------------
 __declspec(dllexport) HRESULT SetAndApplyEffect(int forcex,int forcey)
{
            HRESULT hr;

    LONG rglDirection[2] = { 0,0 };
    DICONSTANTFORCE cf;

            rglDirection[0] = forcex;
            rglDirection[1] = forcey;

            cf.lMagnitude = (DWORD)sqrt( (double)forcex * (double)forcex +
                                (double)forcey * (double)forcey );


    DIEFFECT eff;
    ZeroMemory( &eff, sizeof(eff) );
    eff.dwSize              = sizeof(DIEFFECT);
    eff.dwFlags             = DIEFF_CARTESIAN | DIEFF_OBJECTOFFSETS;
    eff.cAxes               = g_NumForceFeedbackAxis;
    eff.rglDirection        = rglDirection;
    eff.lpEnvelope          = 0;
    eff.cbTypeSpecificParams  = sizeof(DICONSTANTFORCE);
    eff.lpvTypeSpecificParams = &cf;
    eff.dwStartDelay          = 0;


            if( NULL == g_pEffect )
                    return E_FAIL;

    hr=g_pEffect->SetParameters( &eff, DIEP_DIRECTION |
DIEP_TYPESPECIFICPARAMS | DIEP_START );
```

```
    if(FAILED(hr))
        return hr;

    return S_OK;

}

 //-----------------------------------------------------------------------------
// Break()
// Releases Joystick,stops effects...
//-----------------------------------------------------------------------------

__declspec(dllexport) HRESULT Break()
{
   HRESULT hr;

   if( g_pJoystick )
       hr=g_pJoystick->Unacquire();

        hr=g_pEffect->Stop();

        return S_OK;
}

//-----------------------------------------------------------------------------
// PlayEffect()
// Starts Effect...
//-----------------------------------------------------------------------------
__declspec(dllexport) HRESULT PlayEffect()
{
        HRESULT hr;

        if( FAILED( hr = g_pJoystick->SendForceFeedbackCommand( DISFFC_RESET  ) ) )
                return hr;

        if( FAILED( hr=g_pJoystick->Acquire()))
                return hr;
       if( g_pEffect )
                g_pEffect->Start( 1, 0 );

        return S_OK;
}
```

FFJoystick.cpp

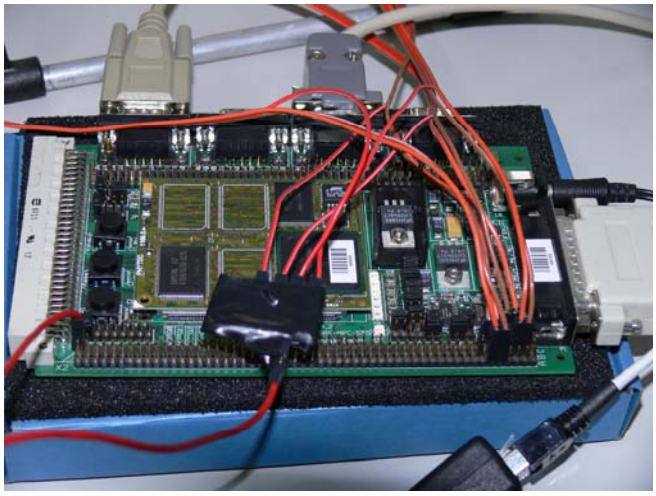
```
#define S_FUNCTION_NAME  FFJoystick
#define S_FUNCTION_LEVEL 2

#include "simstruc.h"
#include "mex.h"
#include <dinput.h>
#include <math.h>
#include <windows.h>


extern DIJOYSTATE2 __declspec(dllimport) js;

extern "C"  HRESULT __declspec(dllimport) InitJoystick();
extern "C"  HRESULT __declspec(dllimport) UpdateJoystick();
extern "C"  HRESULT __declspec(dllimport) SetAndApplyEffect(int,int);
extern "C"  HRESULT __declspec(dllimport) Break();
extern "C"  HRESULT __declspec(dllimport) PlayEffect();

bool first_time=1;

/* Function: mdlInitializeSizes
================================================
 */

static void mdlInitializeSizes(SimStruct *S)
{

   ssSetNumSFcnParams(S, 0);
   if (ssGetNumSFcnParams(S) != ssGetSFcnParamsCount(S)) {
      return;
   }


   if (!ssSetNumInputPorts(S, 2)) return;
   ssSetInputPortWidth(S, 0, DYNAMICALLY_SIZED);
   ssSetInputPortWidth(S, 1, DYNAMICALLY_SIZED);
   ssSetInputPortDirectFeedThrough(S, 0, 1);
   ssSetInputPortDirectFeedThrough(S, 1, 1);


   if (!ssSetNumOutputPorts(S,6)) return;
   ssSetOutputPortWidth(S, 0, DYNAMICALLY_SIZED);
         ssSetOutputPortWidth(S, 1, DYNAMICALLY_SIZED);
         ssSetOutputPortWidth(S, 2, DYNAMICALLY_SIZED);
         ssSetOutputPortWidth(S, 3, DYNAMICALLY_SIZED);
         ssSetOutputPortWidth(S, 4, DYNAMICALLY_SIZED);
         ssSetOutputPortWidth(S, 5, DYNAMICALLY_SIZED);
```

```
   ssSetNumSampleTimes(S, 1);


   ssSetOptions(S,
                SS_OPTION_WORKS_WITH_CODE_REUSE |
                SS_OPTION_EXCEPTION_FREE_CODE |
                SS_OPTION_USE_TLC_WITH_ACCELERATOR );
 }


/* Function: mdlInitializeSampleTimes
=======================================
 */

static void mdlInitializeSampleTimes(SimStruct *S)
{

   ssSetSampleTime(S, 0, CONTINUOUS_SAMPLE_TIME);
   ssSetOffsetTime(S, 0, 0.0);


}

#define MDL_INITIALIZE_CONDITIONS
#if defined(MDL_INITIALIZE_CONDITIONS)

static void mdlInitializeConditions(SimStruct *S)
{
      HRESULT hr;
         if(first_time){
         if( FAILED(hr=InitJoystick()) ){
                  if(hr==E_FAIL)
                  {
                           mexErrMsgTxt("Acces error...Check Joystick's Connections...");
                           ssSetErrorStatus(S,"Acces error...Check Joystick's Connections...");
                  }
                  else
                  {
         mexErrMsgTxt("ERROR:Joystick couldn't initialize...");
         ssSetErrorStatus(S,"ERROR:Joystick couldn't initialize...");
         }
         return;
      }
                  first_time=0;
         }


}
#endif /* MDL_INITIALIZE_CONDITIONS */
```

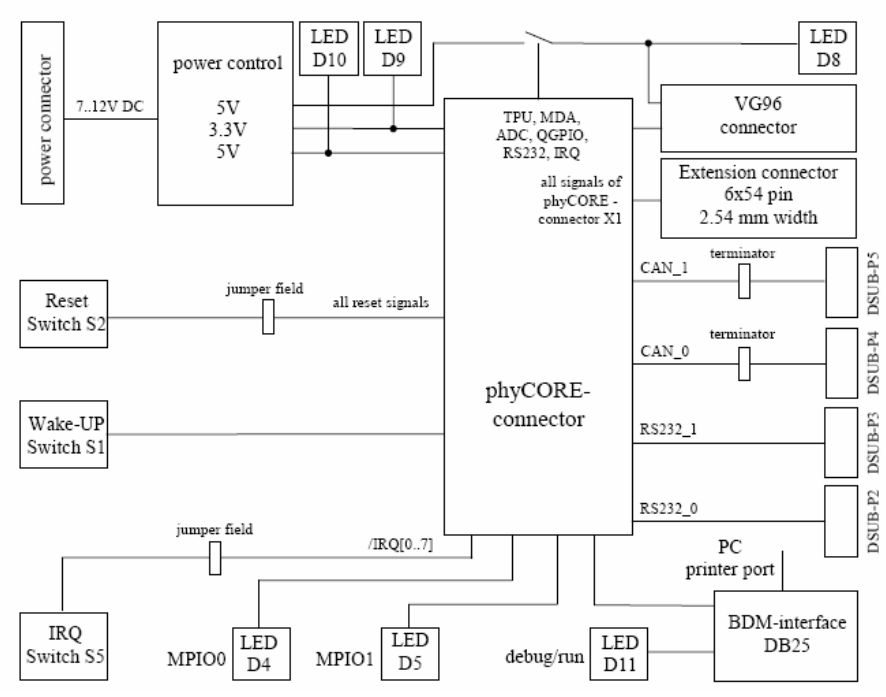
```
static void mdlStart(SimStruct *S)
{
        HRESULT hr;
    if( FAILED(hr=PlayEffect()) ){
    mexErrMsgTxt("ERROR:Effect Couldn't Start...");
    ssSetErrorStatus(S,"ERROR:Effect Couldn't Start...");
    return;
    }
}
/* Function: mdlOutputs
=============================================================
 * Abstract: Joystick's Axes and Aspects
 */

static void mdlOutputs(SimStruct *S, int_T tid)
{
   HRESULT hr;

   int_T           i;
   real_T          *y    = ssGetOutputPortRealSignal(S,0);// Output for Joystick's Rotation
   real_T          *y1   = ssGetOutputPortRealSignal(S,1);// Outpur for Joystick's Buttons
   real_T          *y2   = ssGetOutputPortRealSignal(S,2);// Output for Joystick's axis
   real_T          *y3   = ssGetOutputPortRealSignal(S,3);// Output for Joystick's axis
   real_T          *y4   = ssGetOutputPortRealSignal(S,4);// Output for Joystick's Throttle
   real_T          *y5   = ssGetOutputPortRealSignal(S,5);// Output for Joystick's POV (Point-
of-view)
   InputRealPtrsType uPtrs1 = ssGetInputPortRealSignalPtrs(S,0);// Force-Feedback Input for
First axis
   InputRealPtrsType uPtrs2 = ssGetInputPortRealSignalPtrs(S,1);// Force-Feedback Input for
Second axis

        if(FAILED(UpdateJoystick())){
                mexErrMsgTxt("ERROR:Couldn't get Joystick's State...");
                ssSetErrorStatus(S,"ERROR:Couldn't get Joystick's State...");
        return;
        }

        *y2=js.lX;
        *y=js.lRz;
        *y3=js.lY;
        *y4=js.rglSlider[0];
        if(js.rgdwPOV[0] > 27000) *y5=-1;
        else *y5=js.rgdwPOV[0];

   for(i = 0; i < 128; i++ )
                {
                        if ( js.rgbButtons[i] & 0x80 )
                                *y1=i;
                }
```

```
        if(*uPtrs1[0] < 10000 && *uPtrs1[0] > -10000 && *uPtrs2[0] < 10000 &&
*uPtrs2[0] > -10000)
        {
                if(FAILED(SetAndApplyEffect(*uPtrs1[0],*uPtrs2[0]))){
                        mexErrMsgTxt("ERROR:Couldn't download force...");
                  ssSetErrorStatus(S,"ERROR:Couldn't download force...");
        return;
            }
         }

}

/* Function: mdlTerminate
=====================================================
 */
static void mdlTerminate(SimStruct *S)
{

         if(FAILED(Break())){
                mexErrMsgTxt("ERROR:Couldn't unacquire Joystick...");
                ssSetErrorStatus(S,"ERROR:Couldn't unacquire Joystick...");
        return;
         }
}



#ifdef  MATLAB_MEX_FILE
#include "simulink.c"
#else
#include "cg_sfun.h"
#endif
```

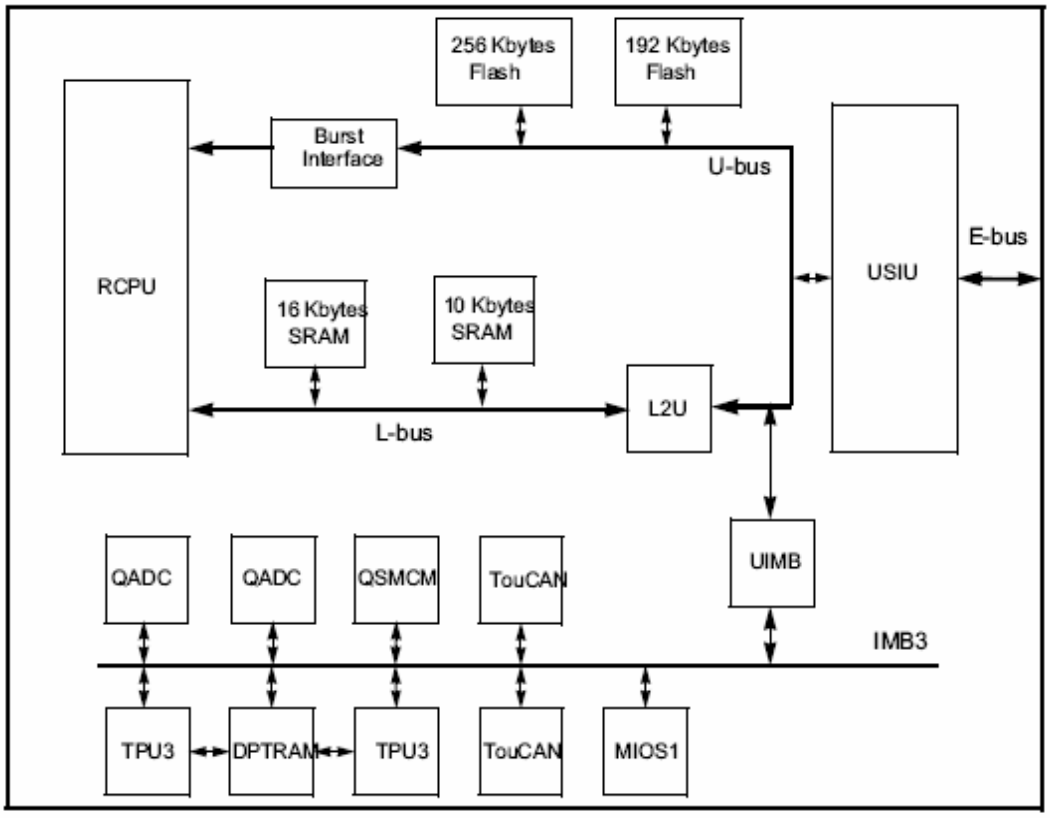
EK – F Phytec MPC555 Gömülü Sistemi



Şekil F.1 : Phytec MPC555 sistemi.



Şekil F.2 : Phytec MPC555 sistemi blok diyagramı.



Şekil F.3 : Motorola MPC555 mikrokontrolör blok diyagramı.

ÖZGEÇMİŞ

Murat DEMİRCİ 1981 yılında Ordu Ünye'de doğdu. İlk, orta ve lise öğrenimini Ordu'da tamamladı. 1998 yılında Uludağ Üniversitesi Mühendislik - Mimarlık Fakültesi Elektronik Mühendisliği bölümünü kazandı ve 2002 yılında buradan mezun oldu. 2004 yılında İstanbul Teknik Üniversitesi Fen Bilimleri Enstitüsü Mekatronik Mühendisliği programında yüksek lisans eğitimine başladı.