Gerçek Zamanlı Gömülü Sistem ve Yazılım Tasarımı'nda ASELSAN Yaklaşımı

Evrim Kahraman[1] Vedat Ünal[2]

[1,2] ASELSAN A.Ş Mikrodalga ve Sistem Teknolojileri Grubu
[1]e-posta: ekahraman@aselsan.com.tr [2]e-posta: unal@aselsan.com.tr

Özetçe

Gömülü sistemler, özel amaçlarla geliştirilen, amaca özel ve genellikle kısıtlı donanımlar üzerinde, güç sınırlamalarına uyan, gerçek zaman ihtiyacını karşılayan, çevre koşulları ile uyumlu ve özel çevre birimleri ile yüksek etkileşimli olarak çalışan sistemlerdir. Bu bildiride, gerçek zaman ihtiyacı ve donanım ve çevre birimleri ile yüksek etkileşimin, gömülü sistemler üzerinde çalışan gömülü yazılımlara etkileri incelenecek ve çözüm yöntemleri sunulacaktır.

ASELSAN A.Ş. Mikrodalga ve Sistem Teknolojileri (MST) Grubu projelerinin gömülü sistemleri kapsamında belirlenen sistem ve yazılım mimarisinden, uygulanan tasarım yöntemleri ve alınan tasarım kararlarından bahsedilecek, gerçek zamanlılık ihtiyacının yazılım tasarımına etkileri irdelenecektir.

1. Giriş

Haberleşme, tıp, savunma, otomotiv ve havacılık uygulamaları gibi geniş bir yelpazede yer alan gömülü sistemler ile günümüzde sıkça karşılaşmaktayız. Bu sistemler, geliştirilme amaçları doğrultusunda oluşturulan özel bilgisayar donanımları üzerinde çalışan gömülü yazılımlar ile sorumlu oldukları görevleri yerine getirirler. Bu açıdan, genel amaçlı kullanılan kişisel bilgisayarlardan ayrılmakta ve sistem tasarımının sadece yeterli donanımları içermesiyle boyut ve maliyette kazanç sağlanmasına imkan vermektedirler [1]. Çoğunlukla kullanıcı arayüzü içermeyen bu tür sistemlerde, hata tespit ve hataya dayanıklılık önemli bir faktör olarak karşımıza çıkmaktadır; hata kaynaklarının erken tespit edilebilmesi, hata oluştuğunda ise sistemin mümkün olduğunca görevini aksatmamak üzere çalışması beklenmektedir.

Gömülü sistemlerde, çoğunlukla gerçek zamanlılık isterleri de bulunmakta ve bu tür sistemler "Gerçek Zamanlı Gömülü Sistem" olarak tanımlanmaktadır. Bu sistemlerde doğruluk için sadece görevin yapılması yetmemekte, bu görevin belirli bir zaman penceresi içinde tamamlanmış olması beklenmektedir. Gerçek zamanlılık, çok yüksek hız performansı değil, zaman performansının kararlılığını gerektirir; yani temel kriter planlanabilirlik ve öngörülebilirliktir.

Gömülü sistemler, özel donanımlar ile çalıştıkları için, donanım bağımlılıkları yüksektir. Bu nedenle, gelişen teknolojilerin ya da seçilen ve kullanılan donanım çözümlerinin piyasadan çekilme durumu dikkate alınmalıdır.

Bildiri akışı içerisinde, 2.bölümde gerçek zamanlı gömülü yazılımlara ait tanımlar verilerek, 3.bölümde

ASELSAN MST grubunun - özellikle Silah Sistemleri projeleri - tecrübelerine dayanılarak, gerçek zamanlı gömülü sistem ve yazılım tasarımı aşamalarında izlenen yöntemler verilecektir.

2. Gerçek Zamanlı Gömülü Sistem/Yazılım Özellikleri

2.1. Gerçek Zamanlı Gömülü Sistem / Yazılım nedir?

Özel bir amaca yönelik işlemleri yapan, kaynakları diğer bilgisayar sistemlerine göre kısıtlı olabilen sistemler "gömülü" olarak tanımlanabilir.

Kritik görevler üstlenen sistemler için görevin yerine getirilme zamanı da önemli bir nokta olabilmektedir. Yerine getirilmesi gereken işlemin doğru olabilmesi için sadece mantıksal olarak doğru olmasının yeterli olmadığı, doğru zamanda yerine getirilmiş olmasının da gerektiği sistemler "gerçek zamanlı" sistemler olarak tanımlanabilir [2]. Gerçek zamanlı sistemler iki ana alt grupta düşünülebilir;

Katı Gerçek Zamanlı: Doğru zamanda gerçekleştirilmeyen bir davranış felaketle sonuçlanan olaylara götürebilir, ölüm veya kazalara neden olabilir. Silah sistemleri, uçuş kontrolü, tren kontrolü sistemleri katı gerçek zamanlı sistemlere örnek olarak verilebilir.

Gevşek Gerçek Zamanlı: Doğru zamanda gerçekleştirilmeyen davranış, istenmemesine rağmen, oluşması durumunda herhangi bir mal veya can kaybına yol açmaz. Sistem, servis kalitesi düşük olarak, görevini yerine getirmeye devam eder. Görüntü ve ses işleme, haberleşme uygulamaları gevşek gerçek zamanlı sistemlere örnek olarak verilebilir.

Gömülü sistemlerde yer alacak yazılımlar "gömülü yazılım" olarak adlandırılırken, sistemin gerçek zamanlılık gereksinimi var ise, geliştirilecek yazılımının ek özellikler taşıması gerekir ve bu tür yazılımlar "gerçek zamanlı gömülü yazılım" olarak adlandırılır.

Gömülü yazılımlar doğrudan donanım üzerinde çalışabilecekleri gibi, bir işletim sistemi üzerinde de çalışabilirler. Bu işletim sistemleri, gerçek zamanlılık,

küçük alan kaplama (small footprint), gelişmiş donanım erişim yetenekleri gibi gömülü yazılımların temel ihtiyaçlarını karşılayabilirler. Çok küçük boyutlu yazılımlarda işletim sistemi kullanmamak tercih edilebilmekle beraber, yazılımdan beklenenler arttıkça işletim sistemi tarafından sunulan yeteneklerin kullanımı tasarım sürecini kolaylaştırmaktadır.

2.2. Gerçek zamanlı işletim sistemleri

Bir işletim sisteminin gerçek zamanlı işletim sistemi olarak değerlendirilebilmesi için taşıması gereken ana özellikler şu şekilde sıralanabilir;

- Önceden kestirilebilir (deterministic) davranış sağlamalı
- Eş zamanlı birden fazla görevi icra edilebilmeli (multitasking)
- Görevler istenen sıra ile icra edilmeli (scheduling)
- Görevler önceliklendirilebilmeli (priority)
- Periyodik görevler için ayarlanabilen yüksek çözünürlüklü sistem saati olmalı (clock tick)
- Zamana bağlı işlevler için zamanlayıcı (timer) içermeli
- Giriş/Çıkış erişimi desteği olmalı (Input/Output)

Günümüzde bu özellikleri sağlayan işletim sistemleri yaygınlaşmıştır; vxWorks, Integrity, LinuxRT bunlar arasında sayılabilir.

Gömülü sistemlerde kullanılan özel donanımlarda işletim sistemlerinin çalıştırılabilmesi için işletim sistemi ile uyumlu arayüzleri sağlayan donanım destek yazılım paketi (BSP) geliştirilmektedir. Donanım destek yazılımlarının geliştirilmesi ilave bir yük getirdiği için, işletim sistemi kullanılan uygulamalarda çoğunlukla bu paketin hazır sunulduğu hazır ticari ürünler (COTS) kullanılmaktadır.

İşletim sistemleri "Görev" (task) ve "Öncelik" belirlemelerine ilişkin altyapı içermekte ve çeşitli görev sıralaması mekanizmalarını sağlamaktadır. Bu tür mekanizmalar çalışma zamanının başında belirlenerek yazılımın çalıştığı süre boyunca uygulanabildiği gibi, yazılım çalışırken de değiştirilebilmektedir.

3. Gerçek Zamanlı Gömülü Sistem/Yazılım Tasarımı

Gerçek zamanlı gömülü sistem/yazılım tasarımı kendine özgü bir yaklaşım gerektirir. Hem sistem, hem de yazılım geliştirme aşamalarında donanım bilgisi önemli ölçüde gereklidir. Çoğu zaman yazılım ve donanım geliştirme çalışmaları iç içe yürür; problemin kaynağının anlaşılması da en az çözümü kadar zor bir problem olarak karşımıza çıkar.

Diğer yazılımlarda hiçbir zaman düşünülmeyen konular gömülü yazılımlarda çözümlenmesi gereken problemlere dönüşebilir. Örneğin pille beslenen bir cihazın yazılımında pil ömrünü uzun tutmak önemli bir problem olarak öncelik kazanır.

3.1. Gerçek zamanlı gömülü sistem tasarımı

Gömülü sistem tasarımında en önemli ve geri dönüşü en zor kararlar sistem tasarımı aşamasında verilen donanım – yazılım görev paylaşımı kararlarıdır. Günümüzde FPGA yeteneklerindeki gelişmelerin sonucu olarak yazılım ile yapılan birçok işlem FPGA kullanılarak yapılabilmektedir.

Gerçek zaman performansı da sistem tasarımı aşamasından itibaren çözülmesi gereken bir problem olarak karşımıza çıkar. Özellikle görüntü işleme sistemleri gibi karmaşık algoritmalar içeren sistemlerde donanım – yazılım paylaşımında gerçek zaman gereksinimleri de değerlendirilmelidir.

Gömülü sistem yazılımı, kendisine verilen görevleri yerine getirmek üzere, bünyesinde işlemci ve giriş /çıkış amaçlı arayüzleri bulunduran ve "işlemci kartı" olarak adlandırılan donanımlar üzerinde koşacak şekilde tasarlanmaktadır. Sistem gereksinimlerinin karşılanması için tek bir işlemcinin kapasitesinin ve kaynaklarının yeterli olamayacağı durumlar da söz konusudur. Birkaç işlemcinin bir arada çalıştığı sistemlerde işlemciler arası yük paylaşımı da iyi analiz edilmesi gereken bir konudur. Donanım arabağlantılarının sistem tasarımı aşamasında kesinleşiyor olması nedeni ile, işlemciler arası görev paylaşım kararlarının doğru alınmış olması, daha ileriki aşamalarda gerçekleştirilecek olan yazılım geliştirme ve entegrasyon çalışmalarının sorunsuz geçmesini sağlayabilir. Hazır ticari donanımların kullanılması durumunda bazı görev paylaşımları bir zorunluluk olarak karşımıza çıkabilir. Örneğin, ihtiyaç duyulan bağlantı sayısı bir ek kartın sağlayabileceğinden fazla ise ve de üzerinde bir ek kart bağlantı yeri bulunan bir hazır ticari donanım kullanılıyorsa; bu bağlantılar iki ayrı kart üzerinde yer almak zorunda kalırlar. Bu durumun oluşturacağı sıkıntılar sistem tasarımı aşamasında yeterince incelenmezse ileride çözümü daha zor problemler haline dönüşebilirler.

Yazılım – donanım arasındaki görev paylaşımında en önemli karar nedenlerinden biri de esnekliktir. Çoğunlukla en esnek tutulması istenen, geliştirmeye açık kalması istenen görevler yazılımlar tarafından gerçekleştirilir. Bu da yazılım gereksinimlerinin donanım gereksinimlerinden daha geç olgunlaşmasına yol açar. Bu aşamadaki en kritik konu yazılım ile sağlanabilecek esnekliğin de bir sınırı olduğudur. Sınırları önceden doğru kestirilememiş esneklik kararları da ileride yazılımın yetersiz kalmasına yol açabilmektedir.

Hazır ticari ürün kullanabilmek için en önemli alt yapı standart bir veri yolu üzerinden birbirine bağlanmış donanımlar ile çalışmaktır. VME, PCI gibi uluslararası kabul görmüş, hazır ürünlerin yaygın olarak bulunabildiği veri yollarının tercih edilmesi ile hem hazır seçenekler oluşturulurken, hem de ileri aşamalar için (özellikle ürün desteği aşaması) çözüm seçenekleri hazırlanmış olacaktır.

Geliştirilen sistemin çalışacağı ortamın zorlu çevre koşulları ihtiyaçlarını karşılamak üzere sağlamlaştırılmış çözümler için de dünyada standart yapılar tanımlanmıştır. Özellikle silah sistemleri gibi her türlü koşulda çalışması beklenen sistemler için yüksek sıcaklıklara dayanıklı, toz, nem gibi problemlere karşı çözüm oluşturan seçenekler de hazır donanım olarak temin edilebilmektedir. Bu ürünlerin pazarı daha dar olmasına karşın son yıllarda ülkemizde alternatif donanımlar geliştirilmiş ve başarısı ispat edilmiştir. Bunlara örnek olarak ASELSAN'ın silah sistemleri verilebilir.

3.2. Gömülü Yazılım Tasarımı

Sistem tasarımı aşamasında gömülü yazılımdan beklentiler tanımlanmaya başlanmasına karşın, yazılım geliştirme sürecindeki analiz ve tasarım aşamalarında alınacak birçok önemli karar vardır. Özellikle yazılım tasarımında alınan kararlar, yazılımın görevini doğru şekilde yerine getirmesi üzerinde çok etkilidir.

3.2.1. Gömülü Yazılım Ön Tasarımı

Gömülü yazılımlar genellikle küçük boyutlu olarak düşünüldüğü için basit geliştirme teknikleri ile geliştirilebileceği görüşü hakimdir. Ancak, uçuş kontrol sistemi veya silah sistemi gibi daha karmaşık görevler üstlenen ve genellikle büyük boyutlu birkaç yazılım içeren gömülü sistemler düşünüldüğünde bu görüşün başarıdan uzak olduğu açıktır.

Günümüz yazılım teknolojisinde öne çıkmış olan UML [3] gibi tasarım yaklaşımlarının gömülü sistem/yazılımlarda da uygulanması mümkündür. Özellikle UML kullanan araçlar hızla yaygınlaşmaktadır. Bu araçlarla yazılım modeli tanımlanmakla beraber otomatik kod üretiminin de yapılabilmesi popülerliğini artırmaktadır. Gerçek zamanlı gömülü yazılımlarda UML kullanımı anlaşılırlığı artırırken gerçek zaman performansını bir miktar düşürebilmektedir. Katı gerçek zaman gerektiren kod bölümlerinin daha çok tasarımcı/programcı katkısı ile geliştirmesine de olanak veren bu araçlar tüm kritik sistemlerde kullanılabilmektedir. ASELSAN'da geliştirilen silah sistemleri, füze ikaz ve kendini koruma sistemleri bu duruma örnek teşkil etmektedir.

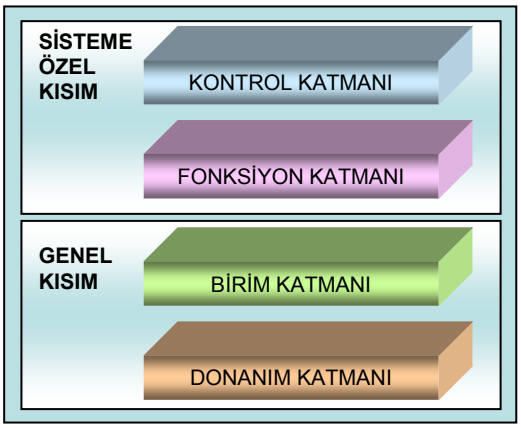
C dili, özellikle donanım erişiminde sağladığı yeteneklerle, gömülü sistemler için ideal geliştirme dili olarak düşünülmekle beraber, günümüzde C++, Java, Ada gibi nesne yönelimli diller giderek önem kazanmışlardır. Nesne yönelimli tasarım ile sağlanan modülerlik, UML araçlarıyla desteklenerek model tabanlı yazılım geliştirme yaklaşımı ile yazılım tasarım kalitesi artırılabilmektedir. Bu yöntemlerle, anlaşılabilirlik, test edilebilirlik, tekrar kullanılabilirlik, değişikliklere karşı uyumlanabilirlik artırılmaktadır. Gereksinim ve senaryolar, UML "Kullanım Durum Diyagram"'ları (Use-Case Diagram) ile ifade edilerek paydaşlar arası görüş alış-verişi kolaylaştırılmaktadır. [4]

Yazılım geliştirme altyapısına ilişkin gereksinimler de bu aşamada değerlendirilmelidir. Gömülü yazılımlarda kaynak kısıtları nedeni ile yazılım geliştirme ve derleme hedef platformda yapılamamakta, yazılım geliştirme altyapısının bulunabileceği yüksek kabiliyetli ve geniş kaynaklara sahip bir başka bilgisayar üzerinde gerçekleştirilmektedir. Böylece oluşturulan çalıştırılabilir yazılım, kaynakları kısıtlı hedef platforma yüklenebilmektedir. Yazılımın hedef platforma yüklenebilmesini sağlayacak altyapının önceden değerlendirilmiş olması, gerekli bağlantıların (ethernet/seri kanal gibi) sistem tasarımında yer almış olması gerekmektedir. Hedef platform ile bağlantıyı sağlayan bu altyapılar, yazılım geliştirme aşamasında, yazılımın test edilmesi ve hataların çözümlenmesi amacıyla da kullanılacak arayüzler olmaktadır.

3.2.2. Gömülü Yazılım Mimarisi ve Yazılım Tasarımı

Karmaşık görevler üstlenen gömülü sistemlerde, yazılımın beklenilen görevleri yerine getirmesinin yanı sıra anlaşılır bir mimari yapı içerisinde esneklik, gelişmeye açıklık, test edilebilirlik, uyumluluk gibi kalite kriterlerini en iyi şekilde karşılayabilecek olması da önemli olmaktadır. Katmanlı mimari [5][6], görev dağılımını sağlamaya ve katmanlar arası soyutlamaya olanak verir nitelikte olması nedeniyle bu kriterleri sağlamaya yardımcı olabilmekte, dolayısıyla da gömülü yazılımlar için uygun bir mimari olarak değerlendirilmektedir.

ASELSAN'da geliştirilen silah sistemleri için katmanlı mimari benimsenmiş ve katmanlar Şekil-1'de gösterilen şekilde belirlenmiştir.

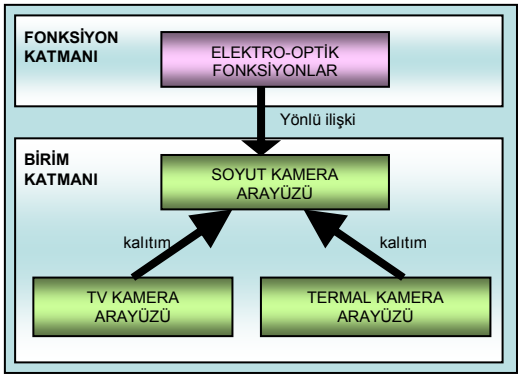


Şekil 1. Yazılım katmanları

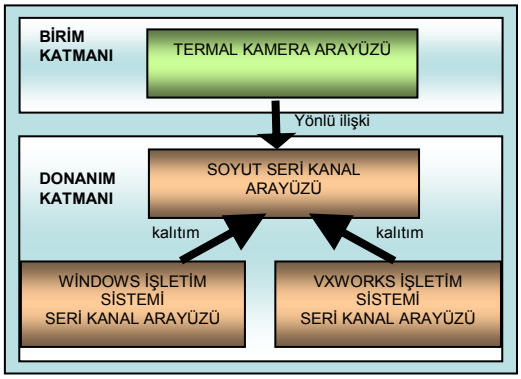
Katmanların görev dağılımı ise şu şekilde yapılmıştır;
Donanım Katmanı: Donanıma erişim için tanımlanacak bileşenler (seri kanal, soket haberleşmesi, paralel giriş/çıkış kartı erişimi vb.),
Birim Katmanı: Donanım katmanı üzerinde çalışan birimlerin arayüz bileşenleri (TV Kamera, Sistem Kumanda Birimi Arayüzü vb.),
Fonksiyon Katmanı: Fonksiyonel görevleri sağlayan kısım (Elektro-optik Fonksiyonlar, Balistik Hesap vb.)
Kontrol Katmanı: Ana yönetim işlevlerinin ve senaryoların sağlandığı kısım

Belirlenen mimari doğrultusunda, yazılım tasarımının UML ile gerçekleştirilmesi sağlanabilmektedir. Mimaride yer alan öğeler ve ilişkileri "Nesne Model Diyagram"'ları (Object Model Diagram) ile, öğeler arası etkileşimler "Ardıl Etkileşim Diyagram"'ları (Sequence Diagram) ile, öğelerin iç işleyişi ise "İşleyiş Diyagram"'ları (State Diagram) ile modellenebilmektedir. [4]

Katmanlar arası soyutlamalar sayesinde değişen/eklenen birim ya da donanımlara uyumluluk hızlıca sağlanabilmektedir. ASELSAN Silah Sistemleri projeleri kapsamında katmanlı mimaride uygulanan birim soyutlama ve donanım soyutlama örnekleri sırası ile Şekil-2 ve Şekil-3'te görülebilmektedir.



Şekil 2. Fonksiyon Katmanı - Birim Katmanı arası soyutlama



Şekil 3. Birim Katmanı – Donanım Katmanı arası soyutlama

Gömülü yazılım tasarımında, hata tespit ve hataya dayanıklılık, bir diğer önemli faktör olarak karşımıza çıkmaktadır. Kritik görevler için gerekli veri kaynaklarının yedeklenmesi sistem tasarımı kapsamında ele alınırken, hatanın tespiti ve bir kaynakta hata oluştuğunda yedek kaynağa geçişin – varsa gerçek zamanlılık isterlerine uygun olarak – yerine getirilmesi yazılım tasarımında ele alınmalıdır. Bu kapsamda, yazılım tasarımında, yazılımın çalışmaya başlarken yapabileceği testlere ve sürekli olarak kullandığı donanım kaynaklarını test etmeye yönelik kısımlar yer almalıdır. Örneğin, ASELSAN Silah Sistemi projelerinde, açılış sırasında işlemci kaynaklarının (bellek v.s) testi yapılarak olası problemler erken aşamada tespit edilebilirken, yazılımın çalıştığı süre boyunca da çevre birimlerle olan bağlantısı kontrol edilmekte ve hata oluşturabilecek durumlar belirlenebilmektedir.

3.2.3. Gerçek Zaman Gereksinimlerinin Yazılım Tasarımına Etkileri

Gömülü gerçek zamanlı yazılımların tasarımında, gerçek zamanlılık isterlerini sağlayacak şekilde "Görev"' olacak kısımlarının tespiti, bu görevlerin "öncelik"lendirilmesi, "Önceliklendirme" ile "Çalışma Süresi" ilişkisinin kurulmasına yönelik belirlemelerin yapılması gerekmektedir. [7]

Gerçek zamanlılık isterlerini sağlayabilmek için ASELSAN MST grubu Silah Sistemlerinin yaklaşımı şu şekilde olmuştur;

- Asenkron olarak veri gelebilen dış arayüzlere yönelik tüm bileşenler "Görev" olarak belirlenir.
- Kendi içinde işleyişe sahip olan nesneler "Görev" olarak belirlenir.
- Sistem görevleri arasında daha önemli olanlar yüksek öncelikli olarak atanır. Örneğin bir silah sistemi için ateşleme ve stabilizasyona ilişkin kısımlar yüksek öncelikli olarak belirlenirken, navigasyon yeteneğine ilişkin kısımların önceliği düşük olarak atanabilir.
- Önceliklendirme – çalışma süresi ilişkisi için eşit önceliklilerin eşit zamana sahip olacağı mekanizma seçilir.
- Periyodik işler için periyotta kabul edilebilir sapmalar bilinmeli ve tasarım ona göre şekillendirilmelidir.
- Hata tespit için eklenen işlevler sistemin yeteneklerini bozmayacak şekilde düşük olarak önceliklendirilir.

UML standardında gerçek zamanlılığa ilişkin tanımlar içerilmektedir. (UML-RT). UML-RT tabanlı bir yazılım geliştirme aracı kullanıldığında, "Görev"'lerin ve "Öncelik"'lerin belirlenmesi model üzerinde yapılabilmekte, otomatik kod üretme yeteneği ile belirlenen özelliklerde gerçek zamanlı çalışacak yazılımın üretilmesi sağlanabilmektedir. [8]

3.2.4. Gerçek Zaman Gereksinimlerinin Kodlamaya Etkileri

Gömülü yazılımlarda, gerçek zamanlılık ve donanım kaynak kısıtları doğrultusunda yazılımın dinamik ve statik davranışı şekillendirilmektedir. Örneğin, çalışma zamanında oluşturulacak nesnelerin yaratılışının dinamik ya da statik olmasında bu iki nokta belirleyici olmaktadır; gerçek zamanlılık isterleri yüksek olan görevlere ilişkin nesneler yazılımın başlangıcında bir kez yaratılarak, sürekli görevleri yerine getirmeye çalışırken, bellek kısıtı nedeni ile, gerçek zamanlılık isteri yüksek olmayan ve sürekli çalışması gerekmeyebilecek nesnelerin görevin icra edilmesi gerektiğinde oluşturulması, tamamlandığında yok edilmesi, böylelikle bellek kullanımının optimum olarak sağlanması benimsenebilir.

Kullanılan programlama dilinin sunduğu yüksek seviye kütüphanelerin içerisindeki bazı hazır fonksiyonların kullanımı gerçek zamanlılık performansını olumsuz yönde etkileyebilmektedir. Örneğin, C++ dilinde STL kullanımı ve üs alma işlemi için hazır fonksiyon kullanımının performans açısından olumsuz etkileri gözlenmiştir.

3.2.5. Gerçek Zaman Gereksinimlerinin Teste Etkileri

Gömülü sistemlerde, hata oluşması durumunda, etkileşimin yüksek olması nedeni ile, hatanın donanım mı yazılım mı kaynaklı olduğu bulmak çok kolay olmamaktadır. Özellikle gerçek zamanlılık da söz konusu ise, durum daha karmaşık bir hal alabilmektedir. Böyle bir hata oluştuğunda, yazılım tasarımcısı/kodlayıcısı tarafından eklenmiş olan hata ayıklama amaçlı kod parçaları ya da geliştirme ortamının hata ayıklama (debug) altyapısı, yazılımın çalışmasını etkileyerek, hatanın ortadan kalkmasına bile sebep olabilmekte, dolayısıyla tespit edilmesini engelleyebilmektedir.

Hatanın tespitinin böylesine zor olduğu gerçek zamanlı gömülü sistemlerde, yazılımın sisteme entegrasyonu sırasında ve öncesinde yapılan testler oldukça önem kazanmaktadır. "Beyaz kutu" ya da "Birim testi" olarak bilinen, yazılımı oluşturan alt birimlerin ayrı ayrı testleri ve "Kara kutu" olarak bilinen yazılımın tamamının testleri, olası yazılım hatalarının mümkün olduğunca erken safhalarda tespitini sağlayabilir. Beyaz kutu testi kapsamında, yazılım tasarımcısı/kodlayıcısı tarafından koda yapılan müdahalelerle yazılımı oluşturan tüm alt parçaların testi tamamlanmalı ve bu parçalarda değişiklikler oldukça testlerin tekrarlanması (gerileme testleri) sağlanmalıdır. Beyaz kutu testleri tamamlanmış birimlerin yazılımı oluşturacak şekilde entegrasyonu sonrası, kara kutu testi kapsamında, bağımsız bir şekilde test senaryoları belirlenerek, belirlenen girdiler doğrultusunda yazılımın çıktılarının incelenebileceği düzenekler kurulmalıdır.

4. Sonuç

ASELSAN MST grubu kapsamındaki projelerin ihtiyaçları gereği, kritik görevler icra eden sistem ve altsistem tasarımları yapılmakta olup, bu sistemler gerçek zamanlı ve gömülü sistem özelliklerini içermektedir. Uzun yıllardır bu teknolojiler doğrultusunda çalışmalar ortaya koyan firmamız, bu alanda teknolojik gelişmeleri takip etmekte ve sistem çözümlerine yansıtmaktadır. Özellikle, yazılım tasarımı kapsamında, esneklik, gelişmeye açıklık, uyumluluk, test edilebilirlik gibi kalite faktörleri değerlendirilerek, mimari ve tasarım çalışmaları yönlendirilmektedir. Bu bildiride, gerçek zamanlı gömülü sistem ve yazılım tasarımı kapsamında elde edilen tecrübeler paylaşılmaya çalışılmış, benzer çalışmalara ışık tutulması hedeflenmiştir. Elde edilen tecrübeler aşağıdaki şekilde özetlenebilir;

- Gerçek zamanlı gömülü sistemlerin, sistem tasarımı aşamasında, gerçek zaman performansı gözönünde bulundurularak, donanım – yazılım görev paylaşımı ve işlemciler arası yük dağılımı analiz edilerek doğru kararların alınması sağlanmış; bu sayede yazılım geliştirme ve entegrasyon çalışmaları sorunsuz halledilebilmiştir.

- Standart bir veri yolu üzerinden birbirine bağlanmış donanımlar ile çalışılarak hem hazır ürünler kullanılabilmiş hem de ileri aşamalar için (özellikle ürün desteği aşaması) çözüm seçenekleri oluşturulmuştur.

- Gömülü yazılımlarda geliştirme faaliyetleri ayrı bir platformda gerçekleştirildiğinden, yazılımın hedef platforma yüklenebilmesini sağlayacak altyapı sistem tasarımı aşamasında belirlenerek, gerekli bağlantıların sistem tasarımında yer alması sağlanmıştır. Bu sayede yazılım geliştirme ve ürün desteği aşamalarında sorunsuz bir altyapı hazır olabilmiştir.

- Gerçek zamanlı gömülü yazılımların, gerçek zamanlılık ve gelişmiş donanım erişim yetenekleri gibi temel ihtiyaçları karşılayabilen işletim sistemleri üzerinde geliştirilmesi yazılım tasarım sürecini kolaylaştırmıştır.

- Katmanlı mimari, görev dağılımını kolaylaştırması ve katmanlar arası soyutlamaya olanak sunması nedeniyle gömülü yazılımların anlaşılırlık ve esneklik kalite faktörlerini karşılayabilmek açısından tercih edilmektedir.

- UML tabanlı araçlar kullanılarak, nesne yönelimli tasarımla sağlanan modülerlik desteklenmiştir. Model tabanlı yazılım geliştirme yaklaşımı ile özellikle anlaşılabilirlik, test edilebilirlik, tekrar kullanılabilirlik ve değişikliklere karşı uyumlanabilirlik yönlerinden tasarım kalitesi artırılmıştır.

- Kullanılan UML tabanlı araçlarla yazılım modeli tanımlanmakla beraber otomatik kod üretiminin de yapılabilmesi ile yazılım tasarımcısına büyük kolaylıklar sağlanmıştır. Modelden başlayarak koda kadar aynı yazılım geliştirme aracının kullanımının, yazılım geliştirme sürecinin her aşamasında tasarımın güncel olarak denetlenebilmesi nedeni ile yazılım kalitesini artırıcı etkisi vardır.

- Gerçek zamanlı gömülü yazılımların "Görev"' ve "Önceliklendirme" tasarımında, asenkron olarak veri gelebilen arayüzlere yönelik bileşenler ve kendi içinde işleyişe sahip bileşenler "Görev" olarak belirlenmiş, önemi yüksek olan sistem yetenekleri ile bağlantılı görevlerin öncelikleri daha yüksek olarak verilmiştir. Bu sayede gerçek zamanlılık isterlerinin maksimum seviyede karşılanması sağlanmıştır.

- Gerçek zamanlılık ve donanım kaynak kısıtları kodlama aşamasında da dikkate alınmıştır. Örneğin, programlama dilinin sunduğu yüksek seviye kütüphanelerin içerisindeki bazı hazır fonksiyonlar gerçek zamanlılık performansını olumsuz yönde etkilemesi nedeniyle kullanılmamıştır.

- Hata tespitinin oldukça zor olduğu gerçek zamanlı gömülü sistemlerde, yazılımın sisteme entegrasyonu öncesinde yapılan "Beyaz kutu" ve "Kara kutu" testleri sayesinde entegrasyon sırasında çıkabilecek problemlerin mümkün olduğunca erken tespit edilmesi sağlanmış ve doğruluğu yüksek yazılımlar elde edilmiştir.

5. Kaynakça

[1] "Embedded System" Wikipedia, *http://www.wikipedia.org/wiki/Embedded_system*

[2] "Real Time System" Wikipedia, *http://www.wikipedia.org/wiki/Real-time_systems*

[3] *http://www.uml.org/*

[4] Martin Fowler, Kendall Scott, "UML Distilled-Applying The Standard Object Modeling Language", Addison-Wesley, 1997

[5] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, M. Stal, “Pattern-Oriented Software Architecture, Volume 1: A System Of Patterns” , John Wiley and Sons, 2000

[6] Jeff Garland, “Representing Software Architectures for Large Scale Systems”, CrystalClear Software Inc, 2001

[7] Bruce Powel Douglass, “Doing Hard Time – Developing Real-Time Systems With UML, Objects, Frameworks, and Patterns”, Addison-Wesley, 1999

[8] Bruce Powel Douglass, “Real-Time UML – Developing Efficient Objects For Embedded Systems”, Addison-Wesley, 1998

6. Kısaltmalar

BSP	: Board Support Package
COTS	: Commercial Of The Shelf
FPGA	: Field-Programmable Gate Array
PCI	: Peripheral Component Interconnect
STL	: Standard Template Library
UML	: Unified Modeling Language
UML-RT	: Unified Modeling Language Real Time
VME	: Versa Module Eurocard