KOCAELİ ÜNİVERSİTESİ FEN BİLİMLERİ ENSTİTÜSÜ

GERÇEK ZAMANLI İŞLETİM SİSTEMİ ÜZERİNDE İNSAN-MAKİNE ARAYÜZÜ TASARIMI

YÜKSEK LİSANS

Elektrik-Elektronik Müh. Yiğit AĞABEYLİ

Anabilim Dalı: Elektronik Haberleşme Mühendisliği

Danışman: Yrd. Doç. Dr. Mustafa Çakır

KOCAELİ, 2010

KOCAELİ ÜNİVERSİTESİ FEN BİLİMLERİ ENSTİTÜSÜ

GERÇEK ZAMANLI İŞLETİM SİSTEMİ ÜZERİNDE İNSAN-MAKİNE ARAYÜZÜ TASARIMI

YÜKSEK LİSANS TEZİ

Elektrik-Elektronik Müh. Yiğit AĞABEYLİ

Tezin Enstitüye Verildiği Tarih: 17 Haziran 2010

Tezin Savunulduğu Tarih: 5 Ağustos 2010

Tez Danışmanı	Üye	Üye
Yrd.Doç.Dr.Mustafa Çakır	Yrd.Doç.Dr.Dilek Tükel	Doç.Dr. Erhan Bütün
(.....)	(.....)	(.....)

KOCAELİ, 2010

ÖNSÖZ VE TEŞEKKÜR

Bu tezin hazırlanmasında bana destek olan danışman hocam Sayın Yrd. Doç. Dr. Mustafa ÇAKIR'a, bilgilerinden istifade ettiğim Kocaeli Üniversitesi Elektronik ve Haberleşme Mühendisliği öğretim üyelerine, araştırma görevlisi arkadaşlarıma ve her zaman yanımda olan aileme en derin şükran hislerimi sunarım.

İÇİNDEKİLER

ŞEKİLLER DİZİNİ

TABLOLAR DİZİNİ

KISALTMA LİSTESİ

GZ	: Gerçek Zamanlı
ICT	: Industrial Control Trainer (Endüstriyel Kontrol Eğiticisi)
FPGA	: Field Programmable Gate Array (Alanda Programlanabilir Kapı Dizisi)
DSP	: Digital Signal Processor (Sayısal Sinyal İşleyici)
CAN	: Controller Area Network (Kontrol Alan Ağı)
RTOS	: Real Time Operating System (Gerçek Zamanlı İşletim Sistemi)
GPL	: General Public Licence (Genel Açık Lisans)
LGPL	: Lesser General Public Licence(Daha Az Genel Açık Lisans)
ISR	: Interrupt Service Routin (Kesme İşleme Rutini)

GERÇEK ZAMANLI İŞLETİM SİSTEMİ ÜZERİNDE İNSAN-MAKİNE ARAYÜZÜ TASARIMI

YİĞİT AĞABEYLİ

Anahtar Kelimeler: Gerçek Zaman, İnsan Makine Arayüzü, İşletim Sistemi, Linux Çekirdeği, HMI, RT-Preempt

Özet: Bu çalışmada, üzerinde gerçek zamanlı Linux işletim sistemi bulunan bir gömülü sistem üzerinde bir HMI(insan-makine arayüzü) cihazı tasarlanması amaçlanmıştır. Uygulama, bir gömülü merkezi donanım bilgisayarı ve buna bağlı insan-makine arayüzü, köprü donanımları ve kontrol edilecek olan bir eğitim setinden oluşmaktadır. Merkezi donanım bilgisayarı ve köprü donanımlarını haberleştirmek için CAN haberleşme protokolü kullanılmıştır. Sistemin gerçek zaman performansını ölçmeye yönelik bazı testler yapılmıştır. Testler sonucunda RT-Preempt yaması uygulanmış gerçek zamanlı Linux çekirdeğinin, standart Linux çekirdeğine göre çok daha iyi bir gerçek zaman performansı gösterdiği tesbit edilmiştir.

HUMAN-MACHINE INTERFACE DESIGN ON REAL TIME OPERATING SYSTEM

YİĞİT AĞABEYLİ

Key Words: Real Time, Human Machine Interface, Operating System, Linux Kernel, HMI, RT-Preempt

Abstract: The purpose of this study is to realize an HMI(human-machine interface) device on an embedded system which runs realtime Linux on it. The application consists of an embedded central hardware computer and human-machine interface on this computer, bridge hardware and the trainer set to be controlled. CAN communication protocol is used to communicate the central hardware computer and the bridge hardware. Some tests are realized to measure the real time performance of the system. According to the test results, it is determined that real time Linux kernel with RT-Preempt patch applied, has a much better real time performance compared to standart Linux kernel.

1. GİRİŞ

Günümüzde gerçek zamanlı işletim sistemlerinin endüstriyel ve askeri teknolojik uygulama alanlarında kullanımı çok büyük önem arzetmektedir. Uygulamanın isterlerini sağlayacak özelliklere sahip bir işletim sisteminin yokluğu bazen çok ciddi ticari zararlara yol açabileceği gibi, bazen ölümle sonuçlanan kazalara veya ciddi yaralanmalara da sebep olabilmektedir. Bir sistemin gerçek zaman performansı basitçe şöyle açıklanabilir: Bir komutun gelmesiyle, ilgili komut işleyici veya görevin çalışmaya başlaması arasında geçen zamanın sıfıra yakınlığıdır. Gerçek zamanlı işletim sistemlerinde bu sürenin rastgele olmaması gerekir. Bu sürenin maksimum değeri gerçek zaman performansını belirler. Farklı uygulamalar için farklı gerçek zaman performans ihtiyaçları olabilir. Mesela; bir mühimmat atım sisteminde iki atış arasındaki maksimum sürenin 10 saniye olması istenebilirken, bir otomobil motoru kontrol sisteminde bir sinyal için tolere edilebilir maksimum süre 1 milisaniye olabilir. Yani sistemin gerçek zaman performansını belirleyen en önemli ölçüt aslında uygulamanın ihtiyaç duyduğu gerçek zaman performansıdır. İşletim sistemlerindeki gerçek zaman performansını ve gecikmeleri etkileyen en önemli konu kesme işleme mekanizmalarıdır. Kesme işleyiciler işlerini hemen bitirmeli ve diğer kesmeleri devre dışı bırakmamalıdırlar. QNX, VxWorks, Linux gibi gerçek zamanlı birçok işletim sistemi bulunmaktadır. Bunların içinde Linux oldukça dikkat çekicidir. Linux çekirdeğine eklenen gerçek zaman performansı yamalarıyla veya Linux türevi gerçek zaman işletim sistemleriyle, ticari rakipleriyle boy ölçüşecek bir seviyeye gelmiştir.

Gömülü sistemler denince akla yakın zamana kadar tek bir olay döngüsü içinde belli başlı basit görevlerin gerçekleştirildiği mikrodenetleyicili sistemler akla gelmekteydi. Gelişen mikroişlemci ve bellek teknolojisiyle beraber, gömülü sistemler üzerinde artık işletim sistemleri de bulunabilmektedir. Bu işletim sistemleri yapılmak istenen göreve göre özelleştirilebilmekte ve düşük güç tüketimine rağmen, ciddi performans artışları sağlanabilmektedir. Linux işletim sistemi, esnekliği ve

kararlılığı sayesinde bu tip gömülü sistemler üzerinde kullanmak için oldukça uygundur.

Endüstriyel sistemlerde birbiriyle organize bir şekilde çalışan farklı elektromekanik birimlerin kontrolü ve durumlarının gözlemlenmesi çok önemlidir. SCADA sistemleri operatörlerin işini kolaylaştırmanın yanında belli otomatik kontroller yaparak sistem güvenliğinin arttırılmasını sağlar. SCADA sistemleri çok farklı coğrafyalardaki sistemlerin tek merkezden gözlem ve kontrolünü sağlayan yazılımsal ve elektronik donanımlardan oluşurlar. İnsan-makine arayüzleri SCADA sistemlerinin önemli parçalarından biridir.

Bu tez çalışmasında, gömülü bir sistem üzerinde çalışan gerçek zamanlı Linux' un endüstriyel otomasyon alanında kullanımına yönelik bir uygulama gerçekleştirilmiş ve gerçek zamanlı Linux ile standart Linux karşılaştırılmıştır.

2. GERÇEK ZAMANLI İŞLETİM SİSTEMLERİ VE LINUX

Gerçek zamanlı sistemler, fonksiyonel olarak doğru çalışmanın yanında, yanıt sürelerinin(response time) de çok önemli olduğu sistemlerdir. Sistem sadece yazılımı çalıştırıp, işlemi sürdürmeyi amaçlamaz; bunun yanında bazı zaman kısıtlarına da uymak zorundadır. Yani, gerçek zamanlı sistemler zaman çizgilerine önem verir.

2.1. Katı Gerçek Zaman(Hard Real Time)

En kötü durum yanıt süreleri, sistem isterlerine uygun olan sistemlere katı gerçek zamanlı denir. Çok net bir tanımlama yapmak pek mümkün değildir; ancak şu söylenebilir: Bir sistem gerekli olan, maksimum en kötü durum yanıt süresini sağlayabiliyorsa, katı gerçek zamanlı olarak değerlendirilebilir. Bir işletim sistemi için "katı gerçek zamanlı" dır demek yerine; bu işletim sistemi bazı uygulamaların katı gerçek zaman gereklerini karşılar, bazılarınınkini ise karşılamaz demek daha doğru olur. İşletim sisteminin karşılaması gereken maksimum yanıt süresi, uygulamanın ihtiyaçlarına göre birkaç mikrosaniye olabileceği gibi, birkaç saniye de olabilir.

Katı gerçek zamanlı sistemin bir kesmeye yanıtının determinist, garanti edilmiş ve kesilebilir(preemptive) olması gerekir. Ortalama yanıt sürelerini değil, en kötü durum sürelerini temel alır. Özetle, katı gerçek zamanlı sistem, bir olaya sınırlı bir zaman içinde, çökmeden, ne olursa olursun yanıt verebilen sistemdir.

2.2. Yumuşak Gerçek Zaman(Soft Real Time)

Katı olmayan gerçek zamandır. Yani zaman isterleri "katı gerçek zaman" larınki kadar kesin olmayan sistemlerdir. Yumuşak gerçek zamanlı sistemlerin , gerçek zamanlı olarak çalışması istenir; ancak gerçek zaman isterlerinin karşılanmaması durumunda sistemin çalışmaya devam ettiği fakat performansının düştüğü görülür.

Yumuşak gerçek zamanlı sistemlerde bir yanıt süresi hedeflenir ve bu sağlanmaya çalışılır.

2.3. Gerçek Zamanlı Sistemlerin Kullanım Alanları

Çoğu bilgisayar sisteminin en azından bazı gerçek zaman kısıtları vardır. Yani, mutlaka işlemleri ile ilgili zaman çizgileri mevcuttur. Son zamanlarda ortaya çıkan ihtiyaçlarla beraber, bu kısıtlar daha bağlayıcı hale gelmiştir. Örnek olarak iki bilgisayar arasındaki VoIp[1] telefon bağlantısını düşünüldüğünde, bazı paketlerin kaybolması durumunda dahi iletişim devam eder. Ama çok fazla paket kaybı olursa, iletişim başarılı olamaz. Sistemin bu durumda başarımı düşer. Bu durum ölümcül sonuçlar doğurmaz. Sistem kendini bir süre sonra toplayabilir. Birkaç paketin kaybolması pek önemli sayılmaz. Bu, yumuşak gerçek zamanlı bir sistem örneğidir.

Katı gerçek zamanlı sistemlerde ise zaman kısıtları oldukça yüksek ve kritik öneme sahiptir. Kısıtların sağlanmaması durumunda kazalar veya can kayıpları meydana gelebilir. Bazı robotlar, endüstriyel otomasyon uygulamaları, uçuş kontrol bilgisayarları bu kapsamda değerlendirilebilirler. Aslında yanıt zamanı kısıtlarına uyulmadığında çöken sistemlere, katı gerçek zamanlı denilebilir. Linux gerçek zamanlı bir uygulamada kullanılmak istendiğinde bazı sıkıntılarla karşılaşılabilir. Linux, katı gerçek zamanlı bir işletim sistemi olarak tasarlanmamıştır. Yanıt sürelerinin "katı" dan ziyade "yumuşak " olduğu masaüstü ve sunucu uygulamaları için tasarlanmıştır. Linux tasarlanırken veri akış uygulamaları pek düşünülmemiş, genel amaçlı bir işletim sistemi olarak geliştirilmiştir.

Günümüzde Linux' un popülerliği ve gücü; pekçok geliştiriciyi Linux'u gömülü ve gerçek zamanlı sistemlerde kullanmaya yöneltmektedir. Açık kaynak kodlu olmasından dolayı ve gerek çok küçük "footprint"[2] boyutu, gerekse gerçek zaman performansı sayesinde ihtiyaçlara çok çabuk cevap verebilmektedir. Değişen ihtiyaçlara göre Linux 'un adapte edilmesinde birden çok yaklaşım mevcuttur. Şu

[1] Tcp/Ip paketleri ile ses iletme yöntemi
[2] Kod derlendiğinde elde edilen çalıştırılabilir imaj

anda iş çizelgeleyicisi(scheduler) ve sonsuz önceliklilik(preemptability)[1] ile ilgili çok sayıda iyileştirme üzerinde çalışılıyor ve deneniyor. Bu iyileştirmelerden bazıları standart çekirdeğe de dahil edilmiş durumdadır. Zamanla bilgisayar donanımlarındaki gelişmeler de gerçek zaman performansının artmasına katkıda bulunacaktır. Ancak bunların hiçbiri en kötü zaman yanıtını, 5-10 mikrosaniye seviyesinin altına düşürecek gibi görünmemektedir. Çok sayıda iş çizelgeleyicisi ve sonsuz önceliklilik yamasından sonra bile, şu anda çekirdeğin yanıt süresi milisaniyenin altına inmemiştir. Aslında bu süre yumuşak gerçek zamanlı uygulamaların çoğu için oldukça yeterlidir. [2]

Onlarca mikrosaniye seviyesinde zaman kısıtlarına uyulmasının gerektiği durumlar için de çözümler mevcuttur. Bu noktada RTLinux vey RTAI gibi çözümler devreye giriyor. Bu sistemler Linux atmosferinin dışında çalışarak mikrosaniyeler seviyesinde en kötü durum yanıt zamanları sağlayabilirler. Temel olarak, sistemin tüm kaynaklarına sahip oluyorlar ve Linux ile çakışmadan, olaylara, kesmelere derhal yanıt verebiliyorlar. RTLinux bunu şu şekilde yapıyor: Yazılımsal bir kesme kontrolcüsü oluşturuyor(aslında sistemin kesme kontrolcüsünü emüle ediyor). Etki olarak RTLinux kullanmak ; gerçek zamanlı bir olaya yanıt vermek için sisteme el koyduğundan; gerçek zamanlı işlemler için ayrı bir işlemci atamaya benzemektedir.

Ancak bunun da bir bedeli vardır. Uygulamanın katı gerçek zamanlı olması gereken kısımları, normal Linux çekirdek fonksiyonları yerine RTLinux un sağladığı fonksiyonlar kullanılarak dizayn edilmek zorundadır. Sistemde iki adet işlemci varmış gibi düşünülebilir (biri hızlı , diğeri yavaş); ancak buna karşılık iki farklı api kullanılması gerekmektedir. Kısaca RTLinux (RTAI) ile normal uygulamalarınızı, direk olarak gerçek zaman avantajlarından faydalanacak şekilde çalıştıramazsınız. Bu yaklaşımı savunanlar uygulamalarınızı gerçek zamanlı kısmı ve gerçek zamanlı olmayan kısmı şeklinde ikiye ayırmanızı tavsiye ediyorlar. Ama büyük ihtimalle çoğu programcı sadece Linux ta programlayıp, Linux çekirdeğinin el verdiği ölçüde gerçek zamanlı olmasını tercih edebilirler. Bu noktada yakın zamanda standart çekirdeğe eklenmiş olan bir eklenti kullanılabilmektedir. CONFIG_PREEMPT_RT

[1] Preemtability, “kesilebilirlik” olarak da tercüme edilebilir; ancak Türkçe literatürde “sonsuz önceliklilik” olarak kullanılmaktadır

topluluğunun geliştirdiği bu eklenti ile standart Linux katı gerçek zaman performansıyla çalıştırılabilmektedir.

Sistemin gerçek zaman performansını tespit etmek veya karakterize etmek konusunda yazılımcıların üzerinde anlaştığı ölçütler olduğunu söylemek zordur. Projenin tipine göre, kullanılacak olan donanıma göre vs. çok farklı ölçütleri göz önünde bulundurmak gerekebilmektedir.

2.4. Yaygın Olarak Kullanılan Gerçek Zamanlı İşletim Sistemleri

2.4.1. QNX Neutrino

QNX Neutrino piyasada en çok kullanılan RTOS lardan biridir. Mikrokernel mimarisine sahip ve POSIX standartlarıyla uyumludur. FAA DO-278[1] ve MIL-STD-1533[2] sertifikalarına sahiptir. Mikrokernel mimarisinde, çekirdek sadece dört temel işlevi yerine getirir: görev planlama, işlemler arası iletişim, düşük seviyeli ağ iletişimi ve kesme işleme. Aygıt sürücüleri de dahil olmak üzere geri kalan bütün iş kullanıcı işlemi olarak çalışır. Böylece çekirdek hızlı sağlam ve küçük olur. [1]

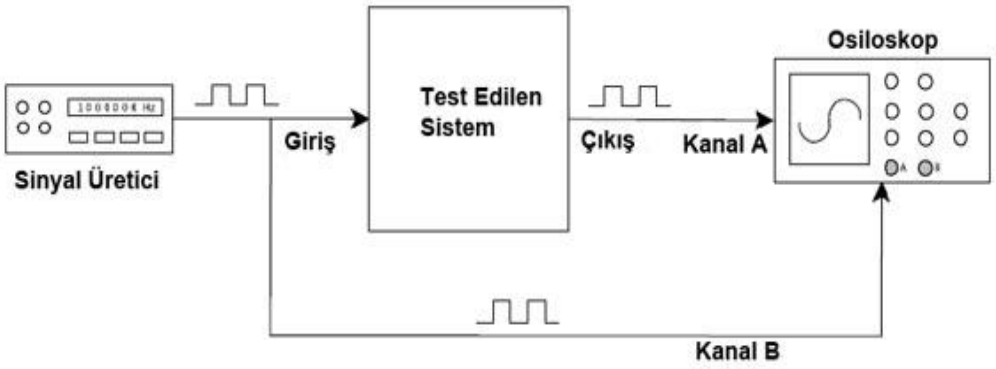
Mikrokernelin en önemli avantajlarından biri, kritik bir hata oluştuğunda(mesela dosya sisteminde) sistemin geri kalanı bundan etkilenmez. Yani mikrokernel mimarisi diğerlerine oranla çok daha sağlam bir yapı sunar. Sistemin bütün kısımları izole olduğu için , çok sık kullanılması gereken "bellek koruma", ek bir yük getirir.

QNX; ARM, MIPS, PowerPC, SH4 ve PC mimarilerini destekler. Bu işletim sistemine eklenen güncel özelliklerden biri "Adaptive Partitioning" dir. Bu özellik işlemci kısıtlarının, işlemlere göre ayarlanabilmesini sağlar. Mesela A işleminin işlemci kullanımını CPU zamanının %30 una, B işleminin işlemci kullanımını CPU zamanının %10 una sınırlandırmak mümkündür..

"Dedicated Systems" tarafından yapılan bir çalışmaya göre karşılaştırıldığı sistemlere (VxWorks AE 1.1, Windows CE) göre QNX 6.1 ciddi bir üstünlük

[1] İletişim, Navigasyon, Gözetleme ve Hava Trafik Yönetim Sistemleri Yazılım Doğruluk Güvencesi Kuralları.
[2] Seri veri yolunun mekanik, elektriksel ve fonksiyonel karakteristiklerini belirleyen standarttır.

sağlamıştır. En kötü yüklenme senaryosunda, yüzlerce işlem bir arada çalışırken ve sisteme 1 Mhz frekansta " ping flood[1] " uygulanmasına rağmen, herhangi bir çökme veya cevap verme süresinde uzama gözlenmemiştir(Şekil 2.1).



Şekil 2.1: Gerçek zamanlı sistemi test etmek için kullanılan bir düzenek[1]

QNX i diğer RTOS lardan ayıran en önemli özellikleri arasında; diğer RTOS larda olmayan "Adaptive Partitioning" , 100 ms gibi kısa bir sürede cihazın aktif hale getirilmesi ve oldukça gelişmiş ve kullanışlı bir geliştirme ortamına sahip olması sayılabilir. Şu anda deneme için 1 aylık sürümü ücretsiz olarak sunulmaktadır. Daha da iyisi AR-GE ve akademik uygulamalar için tamamıyla ücretsizdir. Otomotiv sanayinde büyük oranda QNX kullanılmaktadır. Ancak tüm bu artı özelliklerine rağmen askeri ve uzay uygulamaları için istenen DO-178B[2] sertifikasyonuna sahip olmaması, QNX in bu alanlarda yaygınlaşmasını engellemektedir. [3]

2.4.2. VxWorks

Yakın zamanda Microsoft tarafından satın alınmış olan Windriver VxWorks, gömülü sistemler pazarında en çok kullanılan RTOS dur. VxWorks' un kullanım alanlarına örnek olarak Ulusal Havacılık ve Uzay Dairesi, NASA' nın keşif robotları ve birçok askeri uygulama verilebilir. Bu işletim sistemi güvenilirlik, güvenlik-kritik

[1] Cihaza sürekli olarak yankı talebi yollayarak sistemin yüklenmesini sağlayan bir yazılım
[2] Havacılık Sistemleri ve Ekipmanlarında Yazılım Etkenleri Standardı

uygulamalara uygunluk konusunda uluslararası standartlara uyumluluğunu belgeleyen birçok sertifikaya sahiptir.

VxWorks un esnekliğiyle ilgili ilginç bir örnek NASA nın Mars görevlerinden birinde yaşanmıştır. Robot, Mars yüzeyine indiğinde bir yazılım hatası oluştu. Sistem yazılımı, bir zamanlayıcının robotun bilgisayarını devamlı olarak sıfırlamasından dolayı kararsız hale geldi. Uzaktan ayıklama sistemi kullanan geliştiriciler robottan 70 milyon kilometre uzakta olmalarına rağmen sorunu tespit edip düzelttiler. Problemin kaynağı bir "priority inversion" dı ve VxWorks un "priority inversion" protokolü değiştirilerek kolayca çözülmüş oldu.

3.85 us lik en kötü yanıt süresi ile VxWorks 6.2 RTOS en düşük yanıt süreli RTOS lar içerisindedir. Düşük gecikme süresi sistemin oldukça deterministik olduğunu gösteriyor. [1]

VxWorks' ün bazı sürümleri EAL 7[1] sertifika seviyesini destekleyebilmektedir. Bu sebeple askeri teknoloji alanında faaliyet gösteren şirketler tarafından yıllardır kullanılmaktadır. Ancak VxWorks' ün gelişen teknoloji ihtiyaçlarına artık cevap veremediği ve önümüzdeki 10 yıl içinde kullanım yaygınlığının oldukça azalacağı ve piyasadan silineceği düşünülüyor. [3]

2.4.3. Green Hills INTEGRITY

INTEGRITY RTOS, telifsiz lisans özelliğine sahip bir RTOS. Özellikle güvenilirlik, erişilebilirlik ve hata tolerans ihtiyacı duyan gömülü sistemlerde kullanılmak üzere tasarlanmıştır. velOSity mikrokerneli üzerine inşa edilmiştir ve özellikle MMU(Bellek Yönetim Birimi) ya sahip 32 veya 64 bitlik modern sistem dizaynlarıyla uyumludur. INTEGRITY, hata ve kötü niyetli teşebbüs kaynaklı yanlış işlemlerden, kendisini ve kullanıcı işlemlerini koruyup izole etmek için, donanımsal bellek korumayı kullanır. Desteklediği platformlar arasında ARM, XScale, Blackfin, Freescale ColdFire, MIPS, PowerPC, ve x86 mimarileri bulunur . Güvenlik kritik uygulamalar için sertifikalanmış INTEGRITY-178B sürümü mevcuttur. [4]

[1] Doğruluk Seviyesi Değerlendirmesi 7. Seviye(Uluslararası Standart)

INTEGRITY oldukça yeni bir RTOS olmasına rağmen savunma teknolojileri alanında kendini kanıtlamış durumdadır. VxWorks' te bulunmayan birçok ek özelliğe sahip olmasının yanında, sektörde aranan birçok güvenlik sertifikasını da almış bulunmaktadır. INTEGRITY sistem özelliklerine uygun ve arzu edilen performans ve güvenlik seviyelerine göre farklı işletim sistemleri sunmaktadır. Yazılan kodun farklı sistemlere taşınması diğer RTOS lara nisbeten kolay olabilmektedir.[3]

2.4.4. Linux

Linux gerçek zamanlı bir işletim sistemi olarak dizayn edilmemiş olmasına rağmen, son zamanlarda açık kaynak kodu topluluğu bu konuda çok sayıda çalışma yapmaktadırlar. Bu çalışmalar sonucunda da çok sayıda farklı yaklaşımlarla oluşturulmuş gerçek zamanlı Linux alternatifleri ortaya çıkmıştır. Gerçek zamanlı Linux un bedelsiz açık kaynak kodlu sürümleri (ADEOS, RTAI..)dışında, ticari sürümleri(LynuxWorks, FSMLabs, MontaVista..) de mevcuttur. Linux geniş kullanıcı desteği sayesinde çok sayıda donanım sürücüsüne sahip olmasına rağmen, ticari fikri mülkiyet hakkı bulundurmamasından dolayı, pazardaki potansiyelini tam olarak yakalayamamıştır. Linux' un ilerleyen zaman içinde ne şekilde gelişeceğinin kestirilememesi, tercih edilebilirliğini azaltmaktadır. Tüm bu olumsuzluklara rağmen Linux' un ve gerçek zamanlı Linux' un kullanımı günden güne hızla artmaktadır. Bir sonraki bölümde Linux daha detaylı bir şekilde ele alınacaktır.

Bu bölümde bahsedilen işletim sistemleri ve bunlara ek olarak windows ve uC/OS-II işletim sistemlerine ait bazı ölçümler Tablo 'de verilmiştir.

Tablo 2.1: Ölçülen En Kötü Zaman Değerleri. A: Kesme Gecikmesi, B: Gecikme Sapması[1]

	Win XP	Win CE	Neutrino	µC/OS-II	Linux	RTAI	VxWorks
A	848µs	99µs	35,2µs	3,2µs	98µs	11,4µs	13,4µs
B	700µs	88,8µs	32µs	2,32µs	77,6µs	7.01µs	10,4µs

2.5. Linux ve Gerçek Zamanlı Linux

Linux ortalama performans ve çıktı verecek şekilde dizayn edilmiştir. Gerçek zamanlı işletim sistemleri deterministik olmalıdırlar. Yani standart Linux gerçek zamanlı bir işletim sistemi değildir; ancak Linux'un bir takım gerçek zaman özellikleri de mevcuttur.

2.5.1. Linux' un gerçek zaman özellikleri

Linux, gerçek zamanlı olmasa da temel çekirdek kodunda, gerçek zamanlılığı sağlamaya yönelik bazı özellikler mevcuttur. Bu özellikler şöyle özetlenebilir:

Görev Zamanlama: Linux görev zamanlayıcısı gerçek zaman POSIX[1] uyumludur. Gerçek zamanlı bir sistemin temel özelliği olan, sabit öncelikli SCHED_FIFO[2] protokolünü destekler. Bunun yanında POSIX in gerektirdiği SCHED_RR[3] ve SCHED_OTHER[4] 'ı da sağlar. Öncelikler [0…99] aralığındadır. Ek olarak, Linux 2.6 görev zamanlayıcısı çok sayıda işlemi, problemsiz bir şekilde yönetebilir. Bu yeni "O(1)" karmaşıklığındaki zamanlayıcı Ingo Molnar tarafından geliştirilmiştir.

[1] IEEE tarafından tespit edilmiş olan, Unix türevi işletim sistemlerinde kullanılacak olan yazılım arayüzlerini belirleyen standart
[2] "İlk Giren İlk Çıkar" zamanlayıcı algoritması
[3] "Round Robin" zamanlayıcı algoritması
[4] Çoğu işlem tarafından varsayılan olarak kullanılan çok amaçlı zamanlayıcı algoritması

Sanal Bellek: Sanal bellek kullanan bir sistemde gerçek zamanlı uygulamalar çalıştırılamaz. RAM tükendiğinde karşılaşılan rastgele ve uzun gecikme süreleri, gerçek zamanlı bir uygulama için kabul edilemez. Linux, mlock() ve mlockall() fonksiyonlarıyla belirtilen bellek adresleri veya tüm bellek için sayfalamayı iptal eder. Bu sayede kilitlenen bellek, işlem bitip çıkana kadar RAM üzerinde kalır. mlock() ve mlockall() fonksiyonları da POSIX gerçek zaman uzantısında mevcuttur.

Bellek Paylaşımı: Linux işlemleri birbirleriyle ve sürücülerle POSIX.1b 'ye uygun olarak, mmap() fonksiyonuyla bellek paylaşımı yapabilirler. Linux, POSIX gerçek zaman ekletisine uygun olarak açık paylaşılan bellek nesneleri sağlar.

Gerçek Zamanlı Sinyaller: Yüksek çözünürlüklü zamanlayıcı adımlarını, işlemler arası hızlı mesajlaşmaları, asenkron I/O 'nun tamamlandığını ve doğrudan sinyal gönderimi gibi asenkron olayların oluşumunu işlemlere haber verdikleri için,sinyaller gerçek zamanda çok önemli bir yer tutar. Linux, POSIX gerçek zaman sinyalleri ile tamamıyla uyumludur.

POSIX asenkron I/O: I/O cihazlara ulaşmanın Unix tarafından tanımlanan standart yolu, sonraki dosya erişiminin, yalnızca önceki telep tamamlandığında gerçekleşebildiği, read(), write() bloklayan fonksiyonlarıdır. AIO[1] mekanizması uygulama işlemlerinin ve uygulama tarafından başlatılan I/O taleplerinin üstüste olabilmesini sağlar. Bir işlem, tek bir dosyaya veya çok sayıda dosyaya bir veya daha fazla I/O başlatabilir. Hatta, bağlam değişiminden kaynaklanan gecikmeyi azaltmak için, tek bir sistem çağrısı, bir veya çok sayıda dosya üzerinde, bir dizi I/O işlemi başlatabilir. Linux AIO yeni çekirdeklerde bu özelliği sağlamaktadır.

POSIX iş parçacıkları: POSIX iş parçacıklarının şu anki uygulaması(POSIX 1033.1c) Linux İş Parçacıkları olarak bilir. Linux iş parçacıkları glibc' ye entegre olmuştur ve bir parçası olarak dağıtılmaktadır. [5]

[1] Asynchronous Input Output(Asenkron Giriş Çıkış)

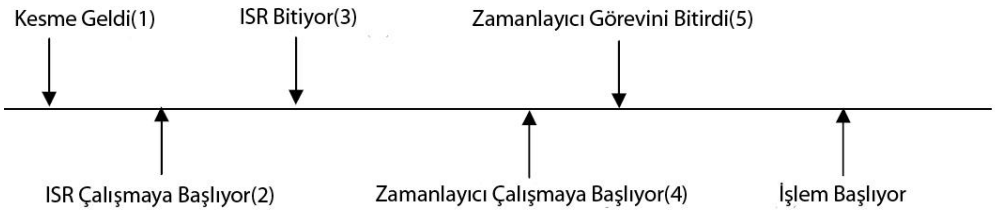
2.5.2. Linux' un gerçek zamanlı olmamasının sebepleri

2.5.2.1. Sonsuz öncelikliliği desteklemeyen çekirdek

Standart Linux çekirdeği sonsuz öncelikliliği(preemptability) desteklemez. Yani işlemci çekirdek kodunu çalıştırırken diğer hiçbir işlem ya da olay çekirdeğin çalışmasını yarıda kesemez. Bu sebeple standart Linux bir RTOS değildir. Burada çekirdek kodundan kasıt kesme işleyiciler[1], "bottom halve" ler, "tasklet"[2] ler ve sistem çağrılarıdır.

2.5.2.2. Deterministik olmayan yanıt süresi

Yanıt süresi bir kesmenin oluşmasıyla onu bekleyen işlemin çalışmaya başlaması arasında geçen süre olarak tanımlanabilir. Bu sürenin deterministik olması gerekir ve düşük değerler tercih edilir. Yanıt süresini belirleyen çalışma noktaları Şekil 2.2 de gösterilmiştir. Linux çekirdeğinde yanıt süresini belirleyen dört ana etken vardır. Bunlar :



Şekil 2.2: Linux Çekirdeğinin Kesme Yanıtının Çalışma Noktaları

Kesme Gecikmesi: Bir kesmenin oluşmasıyla, ilgili kesme işleyicisinin çalışması arasında geçen süredir. Sistem kaynaklarına özel erişimi garantilemek için, Linux çekirdeği kesmeleri devre dışı bırakabilir; bu da kesme gecikmesinde artışa ve deterministikliğin kaybolmasına sebep olur. Şekil 2.2 de 1 ve 2 arasında geçen süre.

[1] Gelen donanımsal kesmeye karşılık olarak çalıştırılan fonksiyon

[2] "Bottom Half" ler ve Tasklet' ler, çekirdek içindeki kesme işleme mekanizmasının parçalarıdır.

Kesme Süresi: Kesme işleyicide geçen süredir. Daha çok donanım sürücüleriyle alakalıdır. Şekil 2.2 de 2 ve 3 arasında geçen süre.

Zamanlayıcı Gecikmesi: Zamanlayıcı gecikmesi, kesmenin oluşmasıyla, zamanlama rutininin çalışması arasında geçen süredir. Bunun iki sebebi vardır:

- Linux' un kesilemez(non-preemptible) kısımlarının özel erişimi garantilemesi

- Zamanlayıcının sabit zaman araklıklarıyla çağrılması (zamanlayıcı tikleri, i386 mimarisinde 10 ms)

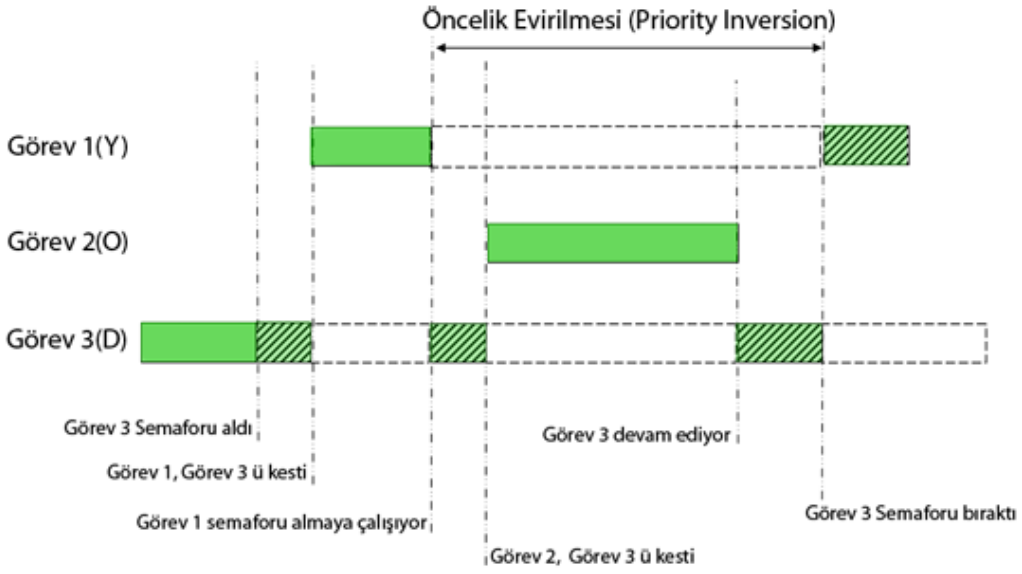
Kısaca, uyandırılmış olan yüksek öncelikli bir görevin devam etmesinden kaynaklanan gecikmedir. Şekil 2.2 de 3 ve 4 arasında geçen süre.

Zamanlayıcı Süresi: Zamanlayıcı rutininde harcanan süredir. 2.4 çekirdekte O(n) karmaşıklığından dolayı, Linux da bu süre çalışan görevlerin sayısına bağlıdır. 2.6 çekirdekde ise O(1) karmaşıklığı dolayısıyla daha tahmin edilebilirdir. Şekil 2.2 de 4 ve 5 arasında geçen süre.

Zamanlayıcı çalışmasını bitirdiğinde, önceliğe ve kalan zaman dilimine göre, kullanıcının beklediğinden başka bir görev seçebilir. Görevin seçilmiş olduğunu varsayarak, bağlam değişimi yapmak zorundadır; bu da yanıt süresini arttırır.

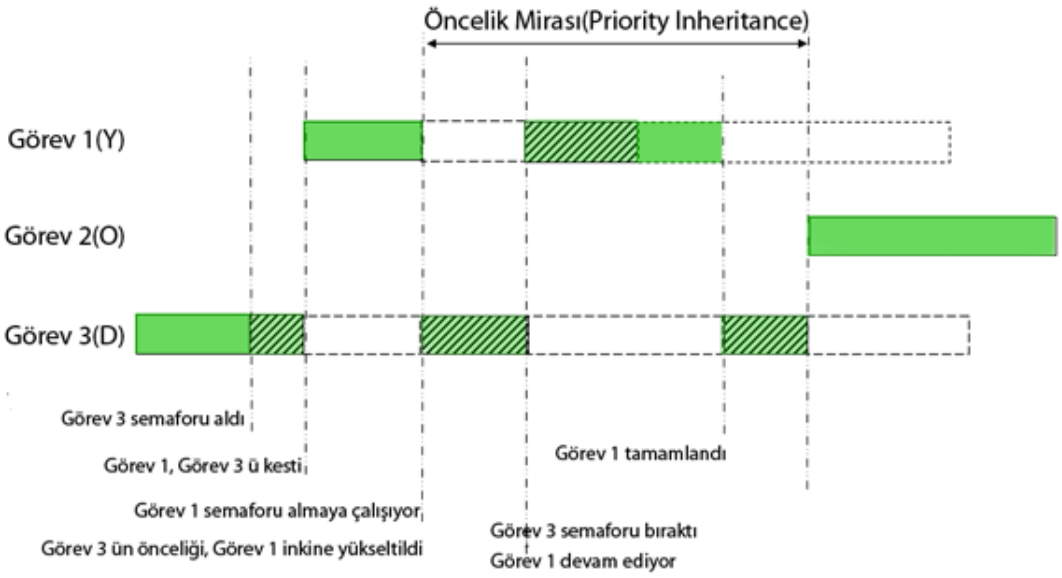
2.5.2.3. Öncelik mirası

"Öncelik Mirası Protokolü", öncelik evirilmesinin oluşmasını önler(Şekil 2.3). Öncelik mirası protokolünde, eğer yüksek öncelikli bir görevin kritik bir bölgeyi kullanımı bloklanmışsa, kritik bölgeyi kilitlemiş olan düşük öncelikli görevin önceliği yükseltilir. Düşük öncelikli görev, bloklanan görevin önceliğini miras alır. Bu sayede orta öncelikli bir görevin, kritik bölgeyi kilitleyen düşük öncelikli bir görevi kesmesi durumunda ortaya çıkabilecek kontrolsüz öncelik evirilmesi önlenmiş olur(Şekil 2.4). Şunu da unutmamak gerekir ki; öncelik mirası orta öncelikli görevlerin gecikmesine sebep olur.



Şekil 2.3: Öncelik Evirilmesi [1][7]

Gerçek zamanlı işletim sistemlerinin deterministik olabilmeleri için öncelik evirilmesini önlemeleri gerekir. Linux un öncelik evirilmesini önleyecek bir mekanizması veya öncelik mirası protokolü yoktur.

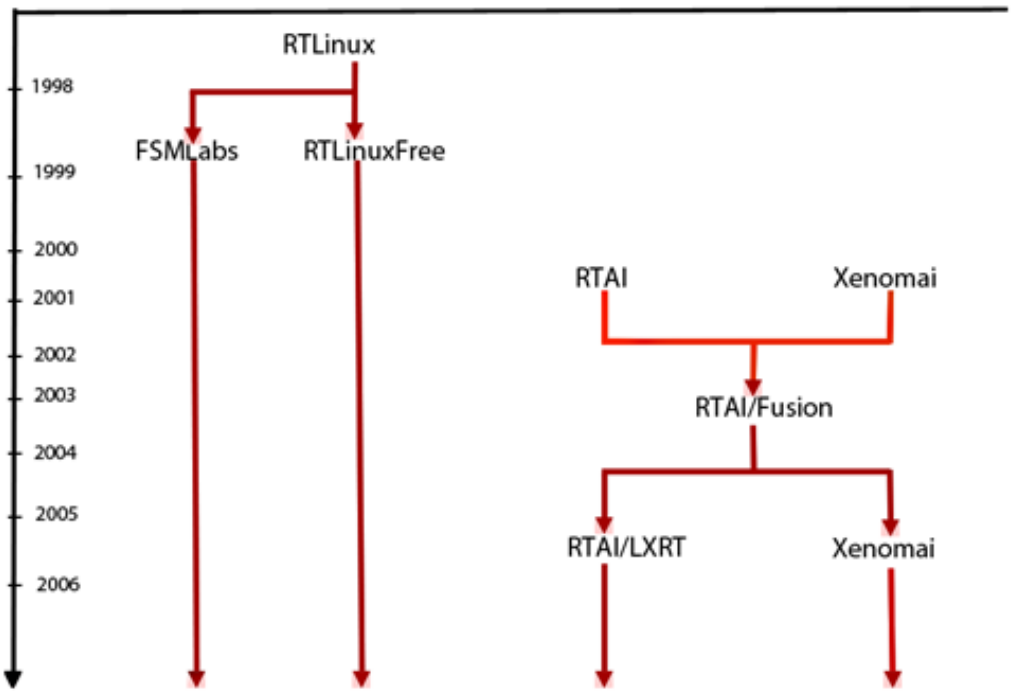


Şekil 2.4: Öncelik Mirası [7]

[1] Semafor: Birden fazla işlem tarafından ulaşılması gereken korumalı bir değişken veya soyut veri tipi

2.5.3. Gerçek zamanlı Linux çözümleri

Önceki bölümde Linux ‘un gerçek zamanlı özelliklerinden bahsedildi. Bu bölümde ise bazı “Gerçek Zamanlı Linux Dağıtımları” açıklanacak. Bu sistemler birbirlerinden oldukça yoğun bir şekilde etkilenmişlerdir(Şekil 2.5). Açıklanacak olan sistemlerden, “CONFIG_PREEMPT_RT” yaması dışındakiler, gerçek zaman performansı sağlayabilmek için özelleşmiş bir mikrokernel mimarisi kullanmaktalar. Bu mikrokernel bir işletim sisteminin; kesme işleme, görev zamanlama, zamanlayıcı kontrolü, görevler arası haberleşme gibi çekirdek fonksiyonlarından sorumludur. Bu mikrokernel üzerinde birden çok *etki alanı*[1] mevcuttur. Görev zamanlama esnasında farklı görev niteliklerinin birbirinden ayrılabilmesi için, etki alanlarının herbirinin sabit bir önceliği vardır. Genelde standart Linux çekirdeğine çok düşük bir öncelik verilirken, gerçek zamanlı görevler yüksek önceliklerle çalışırlar. Bu sayede, gerçek zamanlı bir görev çalışmaya başladığında, Linux çekirdeği kesilebilmektedir; ve bu durumda standart Linux çekirdeği, etkin olmayan bir işlem gibi çalışır.

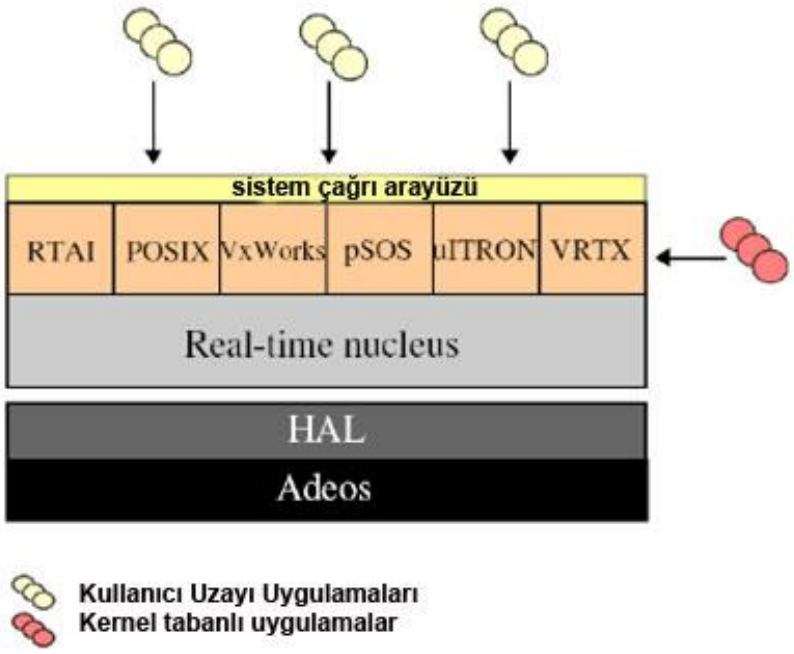


Şekil 2.5: Gerçek Zamanlı Linux’un Tarihçesi

[1] Domain

2.5.3.1. Xenomai

Xenomai, Linux çekirdeğiyle beraber çalışan, kullanıcı uzayı uygulamalarına arayüz esnekliği sağlayarak katı gerçek zaman desteği vermek için GNU/Linux ortamına entegre edilmiş bir gerçek zaman geliştirme platformudur. Xenomai, genel RTOS fonksiyonlarını kullanan bir çekirdek üzerinde, herhangi bir gerçek zaman arayüzünün oluşturulabileceği soyut bir RTOS "core" unu, temel alır. Bu çekirdek üzerinde bir tek jenerik "core" a ait servisler kullanılarak istenildiği kadar "skin" üretilebilir. "Skin" ler, uygulamaların kendilerine has özel gerçek zaman arayüzleridir(Şekil 2.6). Xenomai' nin arm, powerpc, blackfin, nios II, x86 gibi birçok platform için desteği bulunmaktadır.



Şekil 2.6: Xenomai Mimarisi[11]

Çift çekirdekli bir sistem olarak çalışan Xenomai, dış kesmelere hızlı ve deterministik yanıt süreleri sağlayabilmek ve ayrıca standart Linux çekirdeği ile çok iyi entegre olmuş gerçek zaman servisleri sunabilmek için özel bir çekirdek desteğine ihtiyaç duyar. Bu destek, Xenomai kodunu çekirdekle beraber derlemeden önce, standart Linux çekirdek üzerinde uygulanması gereken bir yama şeklinde, Adeos projesi içinde gerçek zaman desteği sunan bir kod katmanı ile sağlanmıştır. Adeos

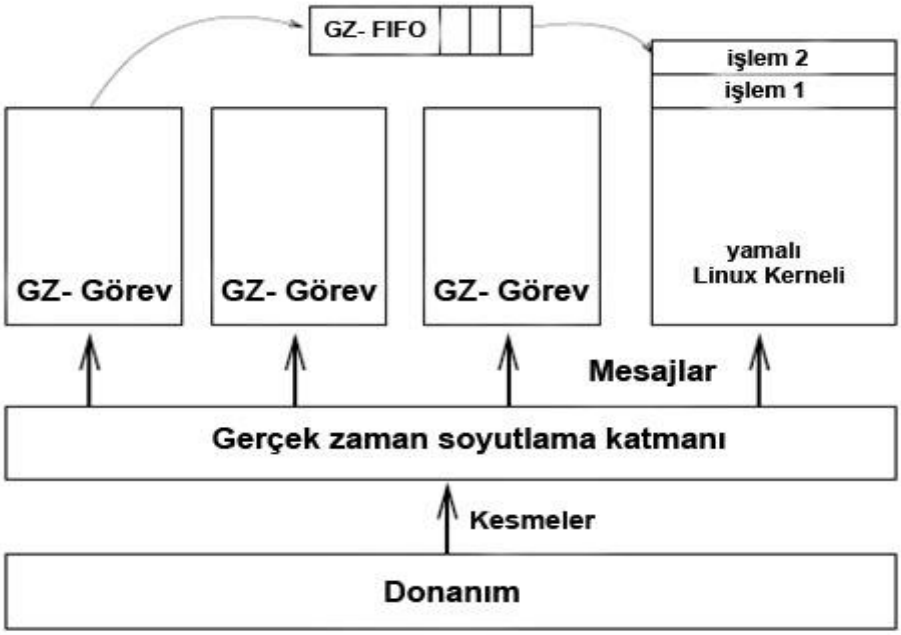
özünde, kesmeleri de içeren önceliği yüksek olayları, kendi içine yönlendiren I-pipe olarak isimlendirilen bir kesme hattı(interrupt pipeline) oluşturur. Xenomai literatüründe, aynı donanım üzerinde Linux çekirdeğine gerçek zaman özellikleri olan ikinci bir çekirdeği de barındırabilme özelliği katan, I-pipe ve Adeos aynı kod parçasını ifade eder.

Xenomai, uygulamalar için Xenomai çekirdeği üzerine kurulmuş birden çok API sunabilir. Bir Xenamai API si, kendisini VxWorks benzeri bir RTOS arayüzüne benzetebilir veya özel bir amaç için yeni bir programlama arayüzü sağlayabilir. Temelde bütün bu API ler ortak çekirdek temelli olamalarına rağmen Xenomai' nin farklı görünmesini sağlar. Bu API uygulamalarına "skin" denir.

2.5.3.2. RTAI

RTAI, RTLinux projesinden ayrılarak devam eden bir projedir. X86, PowerPC ve ARM platformları için destek sunar. RTAI nin, 2002 yılında Xenomai projesiyle birleşmesi sebebiyle, ortak yönleri oldukça fazladır. Xenomai' nin asıl API si ile RTAI API sindeki fonksiyonların sadece isimleri biraz farklıdır. Ancak RTAI, Xenomai gibi farklı API ler sunmaz. RTAI nin, mevcut gerçek zaman uygulamalarını çalıştırmaya uyumlu olma gibi bir amacı yoktur.

RTAI, ilk zamanlarda RTHAL(Real Time Hardware Abstraction Layer) yaklaşımını benimsemişken; RTLinux ile yaşanan lisans problemleri sebebiyle, 2003 yılında ADEOS nanokernel yaklaşımı projeye adapte edildi(Şekil 2.7). RTAI nin kullanıcı kütüphaneleri, LGPL lisansıyla dağıtıldıkları için; RTAI ticari uygulamalarda da lisans bedeli ödenmeksizin kullanılabilmektedir.



Şekil 2.7: RTAI Mimarisi[8]

RTAI, Xenomai ‘ ye benzemekle birlikte; Xenomai de bulunmayan “Showroom” ve “RTAI-Lab Toolchain” gibi araçlar RTAI nin üstünlükleri arasında sayılabilir. Showroom, RTAI’ nin tüm özelliklerinin test edilebileceği bir test yazılımıdır. “Showroom” çok çeşitli ölçümler yapmaya olanak sağlar. “RTAI-Lab Toolchain” ise blok şemaları, RTAI çalışabilir dosyalarına çevirebilen bir araçtır. Bu araç Matlab ve Scilab ile entegre olarak da çalışabilmektedir.

Tablo 2.2 ve **Tablo 2.3**‘de RTAI ve standart Linux’a ait kesme gecikme süreleri ve yüksüz bağlam değişim süreleri gösterilmektedir. 99.999% eşik, ölçümlerin yüzde 99.999 u için geçerli olan gecikme süresini ifade eder.

Tablo 2.2: Kesme Gecikmeleri[14]

İşletim Sis.	**Maksimum (µs) (Yüksüz)**	**Maksimum (µs) (Yüklü)**	**99.999% Eşik(µs) (Yüksüz)**	**99.999% Eşik(µs) (Yüklü)**
2.4 Linux	8.5	113.5	8.5	16.5
2.6 Linux	31.0	49.5	16.0	39.5
2.4 Linux&RTAI	15.5	16.5	7.0	15.5
2.4 Linux&LXRT	10.0	20.0	7.0	16.5

Tablo 2.3: Yüksüz Bağlam Değişim Süreleri[14]

İşletim Sistemi (yüksüz)	**Kesme Gecikmesi (µs)**	**Sonsuz Öncelik gecikmesi (µs)**	**Bağlam Değişim Süresi (µs)**
2.4 Linux	3.5 -- 4.0	12 -- 14	8.0 -- 10.5
2.6 Linux	4.0 -- 4.5	12 -- 14	7.5 -- 10
2.4 Linux RTAI	3.5 -- 4.0	8 -- 10	4.0 -- 6.5
2.4 Linux LXRT	3.5 -- 4.0	10 -- 12	6.0 -- 8.5

2.5.3.3. RTLinux

1998 yılında geliştirilmeye başlanan RTLinux, ilk gerçek zamanlı Linux eklentilerindendir. 1999 yılında FSMLabs bünyesinde ticari bir ürün haline getirilen ve RTLinux/Pro olarak isimlendirilen sürümünün yanında; bir topluluk tarafından desteklenen RTLinux/Free sürümü de bulunuyordu. Zamanla topluluk desteği oldukça zayıflayan RTLinux/Free nin geliştirilmesi neredeyse durmuş iken, Windriver firması 2007 yılında FSMLabs'ı satın aldı ve açık kaynak kodu topluluğunun sahip olduğu kodun geliştirilmesini üstlendi. Şu anda RTLinux' un; akademik amaçlarla kullanılmak üzere sunulan "Open RTLinux" ve ticari sürümü

“Wind River Real-Time Core” olmak üzere iki farklı sürümü mevcuttur. RTLinux i386, Alpha, MIPS ve PowerPC mimarilerini desteklemektedir.

2.5.3.4. RT-Preempt yaması

Standart Linux çekirdeği sadece yumuşak gerçek zaman kısıtlarını karşılayabilmektedir. Kullanıcı uzayı zaman yönetimi için, temel POSIX fonksiyonları sağlar; ancak katı gerçek zaman garantisi yoktur. Ingo Molnar’ın gerçek zaman yaması ve Thomas Gleixner’ın yüksek çözünürlük destekli genel saat olayı katmanı sayesinde, çekirdek katı gerçek zaman kabiliyetleri kazanmıştır. RT-Preempt yaması tez kapsamında geliştirilen sistemde kullanılmıştır.

RT-Preempt yaması Linux çekirdeğini tamamen sonsuz öncelikli(preemptive) bir çekirdeğe dönüştürür. Yama çekirdek kodunda temel olarak şu değişiklikleri yapar[15]:

- “rtmutex” ler kullanarak temel çekirdek kilitlerini sonsuz önceliklendirilebilir(preemptible) hale getirir. Mesela “spinlock_t” ve “rwlock_t” tarafından korunan kritik bölümler “rtmutex” kullanılarak sonsuz önceliklilendirilebilir hale gelmiştir. “raw_spinlock_t” kullanarak sonsuz önceliklendirilemez bölümler oluşturmak hala mümkündür.

Tablo 2.4: RT_PREEMPT yamasının yaptığı değişiklikler

Standart sürüm:	*RT_PREEMPT yaması ile:*

struct rt_mutex {
spinlock_t wait_lock;
struct plist_head wait_list;
struct task_struct *owner;
};
	struct rt_mutex {
	raw_spinlock_t wait_lock;
	struct plist_head wait_list;
	struct task_struct *owner;
	};

typedef struct { raw_spinlock_t raw_lock; unsigned int break_lock; struct lockdep_map dep_map; } spinlock_t;	typedef struct { struct rt_mutex lock; unsigned int break_lock; struct lockdep_map dep_map; }spinlock_t;
typedef struct { raw_rwlock_t raw_lock; unsigned int break_lock; struct lockdep_map dep_map; } rwlock_t;	typedef struct { struct rt_mutex lock; int read_depth; unsigned int break_lock; struct lockdep_map dep_map; } rwlock_t;

❖ Tablo 2.4'te gösterildiği gibi rt_mutex lerin kullanımıyla çekirdek içi "spinlock" lar ve "semaphore" lar için öncelik mirası sağlanmıştır. İstendiği takdirde semaforlar için öncelik mirası, compat_semaphore ve compat_rw_semaphore tanımlarıyla iptal edilebilir.

❖ Kesme işleyiciler, sonsuz önceliklendirilebilir çekirdek iş parçacıkları haline gelmiştir. Rt-Preempt yaması, softirq(soft interrupt handler) leri, bir kullanıcı uzayı uygulaması gibi "task_struct" ile ifade edilen çekirdek işparçacığı bağlamında

çalıştırır. Herhangi bir kesmenin flaginin SA_NODELAY ile işaretlenerek kesme bağlamında çalıştırılması mümkün olmasına rağmen, sadece fpu_irq, irq() irq2 ve lpptest kesmeleri SA_NODELAY ile işaretlenmişlerdir. Bunlardan sadece irq()(CPU zamanlayıcı kesmesi) normal olarak kullanılır. fpu_irq, kayar noktalı işlemci kesmeleri için, lpptest ise kesme gecikme ölçümleri için kullanılır. Yazılımsal zamanlayıcılar(add_timer(),vs.) donanımsal kesme bağlamında çalıştırılmazlar, uygulama bağlamında çalıştırılırlar ve kesilebilir yapıdadırlar.

SA_NODELAY çok sık kullanılmamalıdır. İşlemci saati zamanlayıcılarında, temel çekirdek bileşenleri ve görev zamanlayıcının sıkı süre kısıtlarından dolayı, SA_NODELAY kullanımı uygundur. Ve ayrıca, SA_NODELAY kullanan kesme işleyiciler, çok dikkatli kodlanmalıdırlar; aksi takdirde ölükilitlerine ve sistem "oops" larına sebep olabilirler.

İşlemci saati zamanlayıcıları(scheduler_tick(),vs.) donanımsal kesme bağlamında çalıştırıldıkları için , uygulama bağlamında çalışan kod ile paylaşılan tüm kilitlerin "raw spinlock"(raw_spinlock_t veya raw_rwlock_t) olması gerekir , ve uygulama bağlamı kodu tarafından talep edildiği zaman "_irq" türündekiler kullanılmalıdır(spin_lock_irqsave() gibi…). Bunlara ek olarak, uygulama bağlamındaki kod, SA_NOLDELAY kesme işleyicisiyle paylaşılan "işlemci saati" değişkenlerine eriştiği zaman donanımsal kesmeler iptal edilmelidir.

❖ Rt-Preempt yamasıyla spin_lock() lar uyuyabilir hale geldiğinden, sonsuz önceliklilik(preemption) veya kesmeler iptal edildiğinde, çağırılmaları kural dışıdır. Bazen bu durum, sonsuz öncelikliliğin tekrar aktif hale getirilmesine kadar spin_lock() a ihtiyaç duyan işlemin geciktirilmesiyle çözülebilir:

- Mesela "task_struct"' ın içinde, "spinlock_t alloc_lock" kilidini almak kurala uygun olunca çalıştırılmak üzere, "put_task_struct_delayed()" fonksiyonu bir "put_task_struct()"'ı kuyruğa atar.

- Yukarıdaki put_task_struct_delayed()' e benzer şekilde daha sonra çalıştırılmak üzere, mmdrop_delayed() fonksiyonu bir mmdrop()'u kuyruğa atar.

- TIF_NEED_RESCHED_DELAYED, bir yeniden zamanlama yapar ama bunu yapmadan once işlemin kullanıcı uzayına dönmeye hazır olmasını veya bir sonraki preempt_check_resched_delayed()' ı bekler(hangisi önce olursa). Her iki durumda da amaç, o anda çalışan görev kilidi bırakmadan uyandırılan yüksek öncelikli görevin çalışamayacağı durumlarda, gereksiz kesmelerin(preemption) önlenmesidir. TIF_NEED_RESCHED_DELAYED olmadan, yüksek öncelikli görev, düşük öncelikli görevi, sadece bloklanan kilidi beklemek için, hemen kesecektir.

Bu durumun çözümü spin_unlock()' tan hemen sonra bulunan "wake_up()" 'ın "wake_up_process_sync()" olarak değiştirilmesi olacaktır. Eğer uyandırılacak olan görev, o anda çalışan görevi kesecekse, uyandırma TIF_NEED_RESCHED_DELAYED ile geciktirilir.

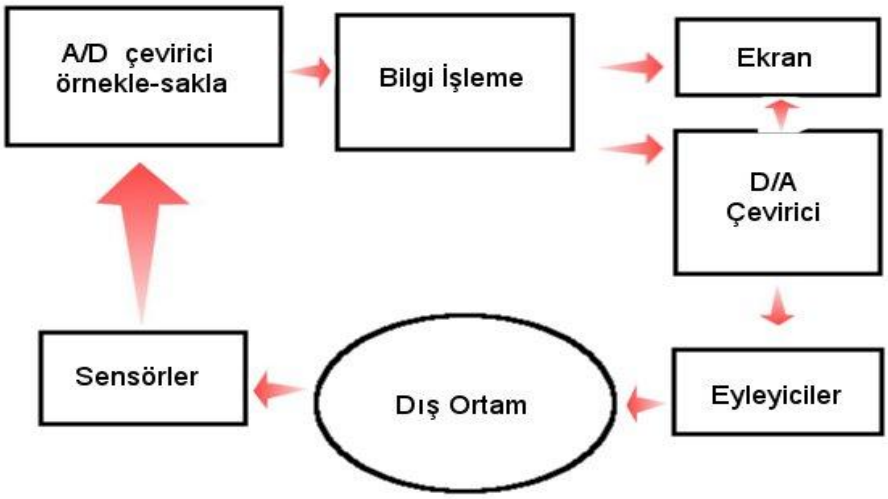
Tüm bu durumlarda çözüm, daha güvenli ve güvenilir çalışabilir duruma gelene kadar işlemi geciktirmektir.

❖ Eski Linux zamanlayıcı API si, yüksek çözünürlüklü çekirdek zamanlayıcıları ve zaman aşımı için farklı altyapılar kullanacak şekilde ayrılmıştır. Böylece kullanıcı uzayı POSIX zamanlayıcılar için yüksek çözünürlük sağlanmıştır.

3. GÖMÜLÜ SİSTEMLER

Bir cihaz içerisinde, özel bir görevi yerine getirmek üzere tasarlanmış olan bilgisayar sistemleridir. Gömülü sistemin kendine ait bir mikrodenetleyici, miktoişlemci ya da DSP(Sayısal Sinyal İşleyici) mevcuttur. Modern gömülü sistemler üzerinde ihtiyaca göre flash bellek, ROM gibi donanım birimlerinin yanında; eternet, CAN, I2C gibi iletişim arayüzler de bulunabilmektedir. Endüstride gömülü sistemler , ürün maliyetlerinin ciddi manada düşürülmesini sağlar. Basit bir cihazın çalıştırılabilmesi için, bir kişisel bilgisayar tahsis etmek yerine, çok düşük maliyetli bir gömülü sistem kullanılabilmektedir.

Kişisel bilgisayarlar, genel amaçlı kullanıma uygun üretilir. Bir kişisel bilgisayar üzerinde onlarca farklı uygulama çalıştırılabilir, yeni uygulamalar yüklenebilir veya silinebilir; ancak gömülü sistemlerde durum farklıdır. Gömülü sistemlerin kullanıcıya sağladığı esneklik oldukça kısıtlıdır. Gömülü bilgisayar kendisinden beklenen görevi gerçekleştirmek üzere yazılımsal ve donanımsal olarak özelleştirilir. Genelde gömülü sistemler üzerindeki yazılımlar çok sık değiştirilmez veya donanımlarına müdahale edilmez. Uzun süre aynı konfigürasyonda çalışacak şekilde dizayn edilirler. Gömülü sistemler üzerinde genelde depolama birimi olarak ROM kullanılır. Sistem üzerinde çalışacak yazılım, sistemin ROM una kaydedilebilir. Bu yazılım, “firmware” olarak adlandırılır. Gömülü sistem donanımları genellikle bir donanım döngüsü içerisinde kullanılırlar(Şekil 3.1).



Şekil 3.1: Gömülü Sistem Donanım Döngüsü

Gömülü sistemlerin genel özellikleri şu şekilde özetlenebilir:

*Genel amaçlı mikrodenetleyici gibi bir merkezi işlem birimi barındırırlar.

*Özel bir uygulama veya amaç için tasarlanırlar.

*Basit bir ara yüzü vardır veya hiç ara yüzü yoktur.

*Genellikle kısıtlı kaynaklara sahiptirler; küçük bellek parmak izi vardır ve sabit disk yoktur.

*Güç harcama limitleri vardır; özellikle pil kullanarak enerjilendiriliyorsa büyük avantaj sağlar.

*Çoğu zaman genel-amaçlı hesaplama platformu olarak kullanılırlar.

*Kullanılması amaçlanan yazılım ve donanım entegre olarak gelir.

*Genelde uygulama yazılımı donanıma gömülüdür, kullanıcı seçemez.

Bununla beraber günümüzde, donanımsal kapasiteleri oldukça yüksek gömülü sistemler de mevcuttur. Bunlara örnek olarak ARM ve POWERPC tabanlı gömülü sistemler verilebilir. MMU(Bellek Yönetim Birimi) ne sahip olan bu sistemler üzerinde işletim sistemleri rahatlıkla çalıştırılabilmektedir. Bu işlemciler, RISC mimarisinin avantajlarından da faydalanmaktadırlar. Bu tür platformlar üzerinde yüksek seviyeli programlama yapmak mümkündür. Hatta bazı özel araçlar kullanarak, kişisel bir bilgisayarda çalışan kodun, bu sistemler üzerinde çalışmasını sağlamak mümkündür. Gömülü sistemlerin donanımsal kapasiteleri arttıkça, üzerlerinde çalışan yazılımların büyüklüğü ve karmaşıklığı artmaya devam edecektir.

3.1. Tarihçesi

Yukarıdaki tanımı göz önünde bulundurursak, düşündüğümüz anlamdaki ilk gömülü sistemler ancak Intel'in ilk mikroişlemciyi ürettiği 1971 yılından sonra görülmeye başlanmıştır. 4004 işlemcisi, Busicom isimli bir Japon şirket tarafından üretilen hesap makineleri için üretilmişti. 1969 yılında Busicom, Intel' den, her biri yeni bir hesap makinesi modeli için, bir takım özel üretim kart tasarlamasını istedi. Intel bu talep üzerine 4004'ü üretti. Her bir hesap makinesi için ayrı bir tasarım yapmak yerine, Intel tüm hesap makinesi modelleri için kullanılabilecek genel amaçlı bir entegre üretmeyi planladı. Bu genel amaçlı işlemci, bir harici bellek çipinde tutulan bir grup komutu okuyup, işlemek amacıyla tasarlanmıştı. Intel'e göre hesap makinelerinin ayırt edici özellikleri yazılımsal olarak ayarlanacaktı.

Sonraki dönemde bu entegrelerin kullanımı hızla yaygınlaştı. İlk gömülü sistemler insansız uzay araçları, bilgisayar kontrollü trafik lambaları, uçuş kontrol sistemleri gibi donanımlarda kullanıldı. 1980 lerde gömülü sistemler mikrobilgisayar çağını sessizce başlattılar ve mikroişlemcileri kişisel ve profesyonel hayatın her yerine soktular. Etrafımızda gördüğümüz her türlü elektronik aletin içerisine girdiler.

Gömülü sistemlerin sayılarının artması kaçınılmaz görünmektedir. Daha şimdiden çok büyük market potansiyeli olan gömülü sistemler mevcuttur: Merkezi bilgisayar tarafından kontrol edilen aydınlatma düğmeleri ve termostatlar, çocuk veya kısa boylu insanlar olması durumunda açılmayan akıllı hava yastığı sistemleri, PDA lar,

dijital kameralar, navigasyon sistemleri… Şu açıkça görülmektedir ki; gömülü sistemlere olan ihtiyaç artarak devam edecektir.

3.2. Gömülü Sistem Donanımları

3.2.1. Mikrodenetleyiciler ve mikroişlemciler

Mikrodenetleyici; işlemci çekirdeği, belleği, osilatör ve programlanabilir giriş/çıkış birimlerini tek bir entegre üzerinde bir araya getiren işlem birimidir. Mikroişlemci ise sadece işlemci çekirdeğinden oluşur. Mikroişlemci ve mikrodenetleyici arasındaki temel fark budur. Genelde mikroişlemcilerin işlem kapasitesi ve harcadıkları güç mikrodenetleyicilere göre daha yüksektir. 4 bitlik ve birkaç Khz de çalışan mikrodenetleyiciler olduğu gibi, 32 bitlik ve yüzlerce Khz hızlara ulaşabilen mikrodenetleyiciler de mevcuttur.

Mikrodenetleyici ve mikroişlemciler, üç temel fonksiyonu yerine getirerek çalışır. Bunlar “al, çöz ve çalıştır” fonksiyonlarıdır. Bu işlem bir döngü halinde devam eder. Program sayacı(Program Counter) çalıştırılacak bir sonraki komutun adresinin tutulduğu kayıt belleğidir(registy). Mikrodenetleyici ilk önce program sayacındaki adreste tutulan komutu alır(fetch). Alınan komutun mikrodenetleyici tarafından uygulanabilmesi için çözülür(decode) . Ardından, çözülen komut çalıştırılır; son olarak ise program sayacının değeri arttırılır. Bu döngüye “al, çöz, çalıştır” (fetch-decode-execute) döngüsü denir(Şekil 3.2).

Şekil 3.2: "Al-çöz-çalıştır" Döngüsü(Fetch-Decode-Execute)

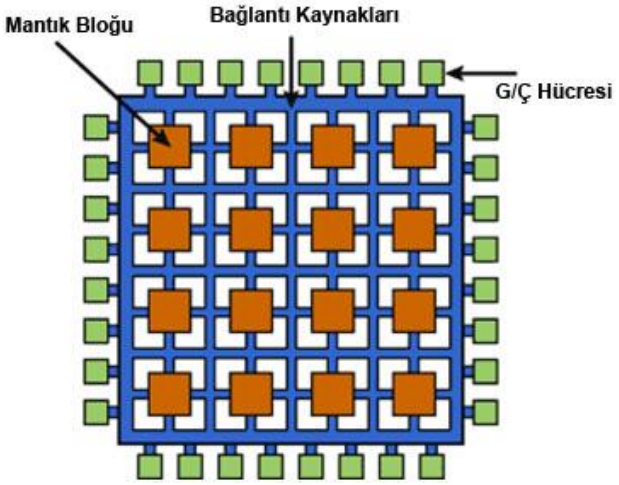
Mikrodenetleyiciyi belleğe bağlayan çok sayıda hat vardır. Mikrodenetleyicinin bellekten bir şey okuması gerektiğinde, ya da belleğe bir şey yazması gerektiğinde, adres veri hattı(address bus), yollardan birinin adresini bildirir. Adres veri hattı tek yönlüdür; yani mikrodenetleyici istediği adrese ulaşabilir, ancak bellek ulaşamaz. Veri yolu(data bus) is çift yönlüdür, mikrodenetleyici ve bellek bu hat üzerinden veri gönderebilir.

Bu hatlar dışında kontrol veri hattı(control bus) denilen ayrı bir hat da mevcuttur. Bu hat belleğin okunuyor mu, yoksa belleğe yazılıyor mu olduğuna ve mikdenetleyiciden belleğe(ya da tam tersi) diğer hatlardaki verinin hazır olduğuna dair sinyal gönderir. Kontrol veri hattındaki kablolardan biri Okuma/Yazma(R/W) yoludur. Bu yolun değeri bellek okunurken "logic 1"e, belleğe yazılırken ise "logic 0" a çekilir. Bir de benzer şekilde "adres veri hattı hazır" (adres veri hattı aktif) yolu vardır. Bu yol, adres veri hattında geçerli bir adres varsa mikrodenetleyici tarafından "1" e çekilir. Bu sinyal belleğe adresin hazır olduğunu haber verir. "Veri yolu hazır"(data bus enable) sinyali de mikrodenetleyici ve bellek arasında benzer bir görev yapar.

Mikrodenetleyiciler sahip oldukları mimariye göre sabit uzunluklu ya da değişken uzunluklu komutlar kullanabilirler. RISC mimarisini kullanan işlemciler genelde sabit uzunluklu komutlar kullanırken, CISC mimarisini kullanan işlemciler değişken uzunluklu komutlar kullanırlar. Her iki mimarinin de kendine göre bazı avantajları mevcuttur. RISC mimarisinde donanım karmaşıklığı CISC e göre daha azdır. CISC ise komut setlerinin çeşitliliği sebebiyle programlama da bazı kolaylıklar sağlar. İki mimariyi kullanan işlemcilerin karşılaştırılmasında çok sayıda farklı ölçüt mevcuttur. Bu ölçütler detaylı olarak anlatılmayacak. Ancak şunu unutmamak gerekir ki uygulamamızın isterlerine göre donanım seçilirken, bu ölçütlerin göz önünde bulundurulması çok önemlidir.

3.2.2. FPGA

FPGA (Alanda Programlanabilir Kapı Dizileri), genellikle bir elektronik devrenin donanımsal karmaşıklığını azaltmak için kullanılır. FPGA yazılımı, FPGA içerisindeki mantık kapılarının birbirine uygun şekilde bağlanmasına dayanır(Şekil 3.3). FPGA içerisinde herhangi bir bellek ya da mikroişlemci bulunmaz. Sahip olduğu teknolojiye göre ve üzerine yüklenen yazılıma göre programlanabilen mantık bloklarına sahiptir. FPGA ler çok hızlı çalışma kabiliyetine sahiptirler. Yüklenen kodun tamamını paralel ve tek döngüde çalıştırılır.

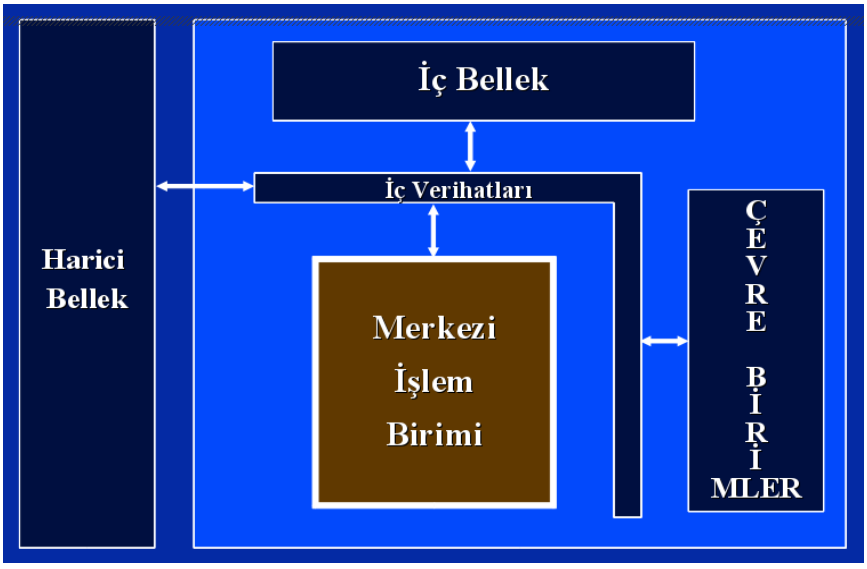


Şekil 3.3: FPGA Genel Yapısı

FPGA defalarca programlanabilir. Bu durum donanım geliştiriciler için oldukça faydalıdır. Bir entegre üretilmeden önce FPGA üzerinde tasarlanıp test edilebilir. Daha sonra nihai ürün meydana getirilir. FPGA ler paralel işleme ve yüksek hesaplama performansı gerektiren uygulamalar için çok uygundur. FFT ve konvolusyon hesaplamaları, şifreleme uygulamalarında düşük saat frekansında çalışan FPGA ler bile çok önemli performans artışları sağlarlar.

3.2.3. DSP

DSP(Dijital sinyal işleyici) aslında bir çeşit mikroişlemcidir. Özel mimarisi sayesinde, hızlı dijital sinyal işleme kabiliyeti kazandırılmıştır. Lineer fazlı FIR filtreleri, adaptiv filtreler gibi bazı uygulamalarda analog sinyaller yerine dijital sinyallerin işlenmesi önemli performans avantajları sağlar. Gelişen dijital sinyal işleme teknikleri sayesinde bu uygulamalar, analog sinyal işleme teknikleriyle ulaşılması mümkün olmayan performanslarda çalıştırılırlar. DSP ler, normal mikroişlemcilere göre çok daha az güç tüketirler ve düşük maliyetlidirler.



Şekil 3.4: DSP Blok Şeması

Bu işlemciler toplama ve çarpma işlemlerini gerçekleştirmek üzere optimize edilmiştir. Bu işlemler tek döngüde donanımsal olarak yapılır. DSP lerin kayan nokta ve sabit nokta hesaplamaları yapan türleri mevcuttur. Kayan nokta hesaplamaları

yapabilen işlemciler daha yüksek doğruluk, yüksek sinyal-gürültü oranı ve kullanım kolaylığı sağlamalarına rağmen; sabit nokta DSP lere göre daha fazla güç tüketirler, daha yavaş olabilirler ve daha pahalıdırlar. Bir sinyalin işlenmesi esnasında geçen süre, örnekleme süresinden küçükse DSP gerçek zamanlı olarak çalışabilir. DSP lerin bu şekilde gerçek zamanlı olarak çalışması gerekir. Sinyal işleme algoritması, örnekleme süresi gibi parametrelerin doğru şekilde ayarlanması önem arz eder.

3.3. Kullanım Alanları

Gömülü sistemlerin kullanım alanları oldukça geniştir. Basit bir kol saatinde kullanılabildikleri gibi, dev santralleri kontrol eden sistemlerin içerisinde de bulunabilirler. Ve benzer şekilde, bir gömülü sistem tek bir mikroişlemciden oluşabileceği gibi; paralel çalışan onlarca işlemci, çevre birimlerinden oluşabilir. Bir sistemin gerçek zaman kısıtı olan parçaları genellikle gömülü sistemler üzerinde oluşturulur. Gömülü sistemlerin kullanım alanları aşağıda birer örnekle özetlenmiştir.

3.3.1. Askeri ve uzay uygulamaları

TÜBİTAK' ın geliştirdiği elde taşınabilir mayın sisteminde metal detektörü ve GPR algılayıcıları bulunmakta olup örüntü tanıma teknikleri ile gömülü cisim teşhisi gerçekleştirilmektedir. Metalik ve nonmetalik mayınları tespit eden ve daha sonra bilgisayarlı örüntü tanıma teknikleri ile cinslerini belirleyebilen bir üründür. Bu sistem ile metal ve plastik içeren tüm mayınlar tespit edilebilmektedir. Gömülü cisimlerin cinslerinin belirlenmesinden sonra tespit sonuçları kullanıcıya görsel ve işitsel olarak aktarılmaktadır. Bu cihaz gerek askeri gerekse sivil amaçlı mayın temizleme işlemlerinde kullanılabilecek niteliktedir. Bununla birlikte toprak altında gömülü bulunan boru, kablo, asfalt kitlesi ve tarihi kalıntıların araştırılmasında bu sistemden faydalanmak mümkündür. Elde taşınabilir mayın sistemi üzerinde gömülü Linux çalışan ve powerpc işlemciye sahip bir gömülü sistem örneğidir. Sistem veri işleme aşamasında FPGA lerden de faydalanmaktadır.

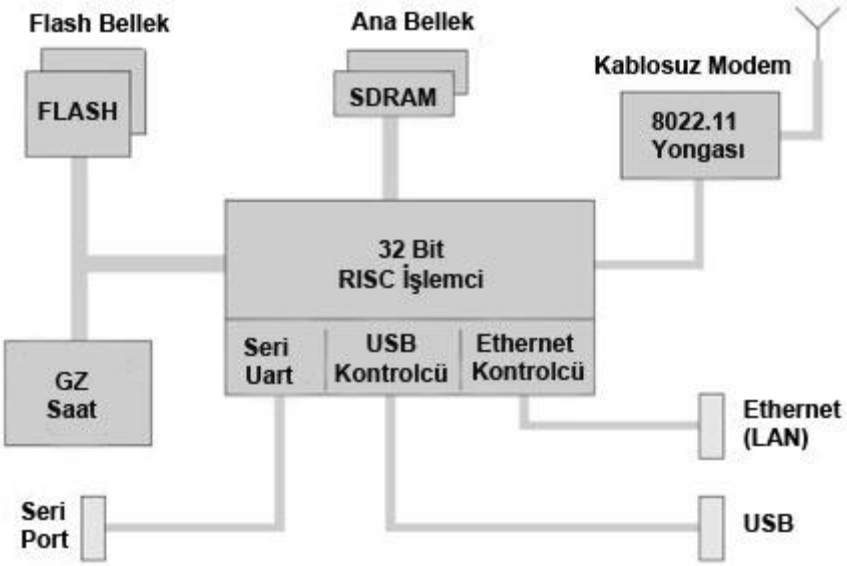
3.3.2. Tıbbi elektronik teknolojisi

Kalp pilleri tıp alanında oldukça sık kullanılan gömülü cihazlardandır(Şekil 3.5). Milyonlarca kalp hastasının yaşamı bu küçük cihaza bağlı durumdadır. Bu tip cihazlar katı gerçek zamanlı sistemlerdir. Son yıllarda harcanan enerjinin optimizasyonu ve cihazın ağırlığı gibi kritik noktalarda çok ciddi iyileştirmeler yapılmıştır.

Şekil 3.5: Kalp Pili(St Jude Medical)

Kalp pili üzerinde elektriksel uyarıların üretilmesini sağlayan bir jeneratör ve üretilen uyarının kalbe iletilmesini sağlayan elektrot mevcuttur. Kalp pilinin çalışma frekansı hastanın metabolik yapısına uygun olarak dışarıdan ayarlanabilmektedir. Kalp pillerinin ömrü 10 yıla kadar uzayabilmektedir. Cihaz üzerindeki bir sistem sayesinde, pilin ne kadar ömrü kaldığı dışarıdan tespit edilebilmektedir.

3.3.3. İletişim uygulamaları



Şekil 3.6: Bir Kablosuz Erişim Noktasının Donanım Mimarisi[9]

Şekil 3.6 de günümüzde kullanılan kablosuz erişim noktalarında rastlanabilecek, yüksek seviyeli donanım mimarisi gösterilmiştir. Sistemde 32 bitlik bir RISC işlemci kullanılmış. Flash bellek, kalıcı program ve veri depolama birimi olarak tasarlanmış. Ana bellek olarak, uygulama tipine göre birkaç megabayttan yüzlerce megabayta kadar arttırılabilecek olan, senkron dinamik rastgele erişimli bellek(SDRAM) kullanılmış. Genelde pille beslenen gerçek zaman saati modülü(real-time clock module) gün ve saati tutmak için kullanılır. Bu örnekte, RS-232 üzerinden seri konsolun yanında ethernet ve usb arabirimleri de bulunmaktadır. 802.11 entegresi kablosuz modem fonksiyonunu sağlar.

3.3.4. Elektronik uygulamaları ve tüketici cihazları

Şekil 3.7: TomTom Navigasyon Sistemi

Şekil 3.7 ‘de gösterilen TomTom Go 910, üzerinde gömülü Linux işletim sitemi bulunan bir navigasyon sistemi. Sistem içerisinde 12 kanallı bir GPS ünitesi bulunmakta. Ayrıca “kılavuzlu uydu teknolojisi “ sayesinde uydu sinyallerinin olmadığı ya da çok zayıf kaldığı tüneller ve yüksek bina araları gibi yerlerde pozisyon tahmini yapabilmektedir. Cihaz üzerinde 400 Mhz ARM9 işlemci mevcuttur. Flash bellek üzerinden 64 mb lık RAM kullanarak açılır. 480X272 lık 4inch bir dokunmatik TFT LCD dokunmatik ekrana ve gömülü bir antene sahiptir. 20 GB lık sabitdisk yanında, içerisinde haritalar bulunan SD kartları yükleyebilmek için SD kart girişi, mini-usb girişi, entegre hoparlörler ve şarj giriş slotları da cihaz üzerinde bulunmaktadır.

3.3.5. Endüstriyel otomasyon ve süreç denetimi uygulamaları

Endüstriyel uygulamalarda ihtiyaçlar uygulama tipine göre değişkenlik gösterse de, tipik olarak şu maddeler önem arz eder:

- Bulunabilirlik ve güvenilirlik
- Güvenlik

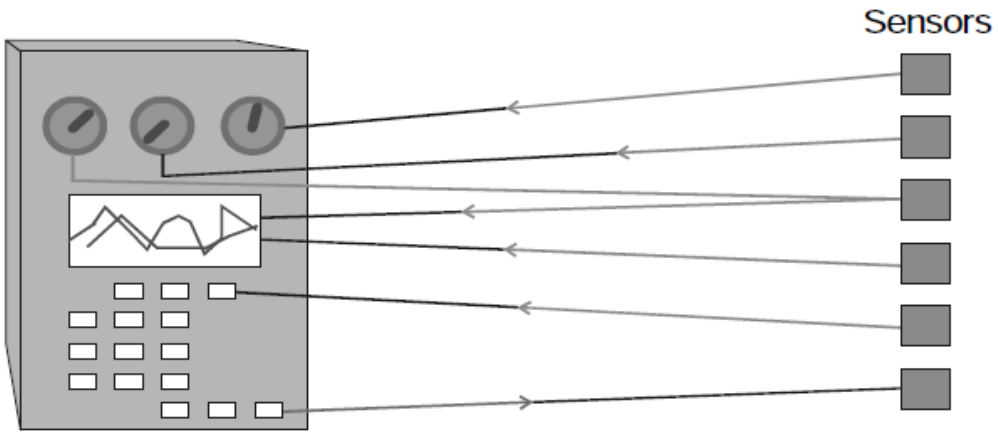
- Devamlılık
- Emniyet
- Gerçek zaman, deterministik yanıt
- Güç tüketimi
- Kullanım ömrü

Endüstri de SCADA sistemlerinde, motor kontrol uygulamalarında, algılayıcı ağlarında, RFID, endüstriyel haberleşme ve daha birçok alanda gömülü sistemler kullanılmaktadır.

4. SCADA SİSTEMLERİ

4.1. Temel Özellikler

SCADA(Supervisor Control And Data Acquisition), kontrol sistemleri kullanılmaya başlandığından beri bilinen bir kavramdır. İlk SCADA sistemleri gösterge, ışık ve grafik çizicilerden oluşan panellerdi. Ana kontrol görevi, giriç cihazlarını el ile ayarlayan operatöre aitti. Bunlara benzer paneller aslında santrallerde, fabrikalarda ve enerji üretim tesislerinde kullanılmakta. Şekil 4.1 de algılayıcıdan-panele tip bir SCADA sistemi gösterilmiştir. Bu basit SCADA sisteminde herhangi bir CPU, RAM veya yazılım bulunmamaktadır. Algılayıcılar direkt olarak göstergelere, ışıklara ve anahtarlara bağlı durumdadır. Bu tip bir sistem maliyet olarak avantajlı olsa da; yüzlerce algılayıcı kullanılan bir sistemde yüzlerce ayrı kablo kullanılması gerekir. Ve ayrıca böyle bir sistemin tekrar konfigüre edilmesi ve kontrolü de oldukça güçtür. Herhangi bir otomatik kontrol de olmadığı için bir operatörün 24 saat gözlem yapması gerekir.



Şekil 4.1: Algılayıcıdan Panele tip SCADA[10]

Modern SCADA sistemlerinde birbirinden uzakta bulunan sistemleri ve teçhizatları birbirlerine bağlama ihtiyacı bulunur. Bu mesafe birkaç metreden binlerce kilometreye kadar değişebilir. Komutlar, programlar ve gözlem bilgisinin alınması için telemetri sistemleri kullanılır.

Aslında SCADA, telemetri ve veri toplamanın birleşiminden oluşur. SCADA; bilginin toplanması ve merkeze bildirilmesi, gerekli analiz ve kontrolün gerçekleştirilmesi, son olarak da verinin operatör ekranlarında gösterilmesi işlerini yapar. Daha sonra gerekli kontrol komutları işleme geri aktarılır.

4.2. SCADA Donanımları

SCADA sistemi, en az bir merkez istasyon ve bu istasyona bağlı uzak istasyon birimlerinden(RTU[1]) oluşur. RTU lar sahadan veri toplar ve bu veriyi iletişim sistemi üzerinden merkez istasyona gönderir. Merkez istasyon aldığı veriyi insan-makine arayüzü üzerinde gösterir ve operatörün uzak kontrol işlerini yapmasına olanak verir.

Doğru ve zamanında elde edilen veri, tesis işlemlerinde iyileştirme sağlar. Diğer faydaları da; etkili, güvenilir ve daha önemlisi güvenlikli işlem sağlar. Bu durum önceki otomasyonsuz sistemlere göre daha düşük maliyetlerin oluşmasına yardımcı olur.

Gelişmiş bir SCADA sisteminde 5 temel seviye bulunur:

- Saha ölçümleri ve kontrol ekipmanları
- Yönetim terminalleri ve RTU' lar
- İletişim sistemi
- Merkez istasyon

[1] Remote Terminal Unit(Uzak İstasyon Birimi)

- Ticari veri işlem bölümü bilgisayar sistemi

RTU, sahadaki analog ve dijital algılayıcılar için bir ara yüz sağlar.

İletişim sistemi uzaktaki sistemlerle merkez istasyon arasında bir köprü oluşturur. Bu iletişim sistemi; kablo, fiber-optik, radyo, telefon hattı, mikrodalga veya uydu üzerinden bile sağlanabilir. Etkili ve optimum veri transferi için, özel protokoller ve hata tespit mantıkları kullanılır.

Merkez istasyon çok sayıdaki RTU dan verileri alır ve genellikle bilginin gösterilmesi ve uzaktaki teçhizatın kontrolü için bir operatör ara yüzü sağlar. Büyük telemetri sistemlerinde, alt merkezler uzak istasyondan veriyi alır ve ana merkez istasyona aktarır.

4.3. SCADA Yazılımları

Ticari ve açık kaynak kodlu SCADA yazılımları mevcuttur. Şirketler ürettikleri donanımlarla haberleşme sağlamak için ticari yazılımlar üretirler. Bu sistemler "anahtar teslim" çözümler olarak satılırlar. Bu sitemlerdeki temel problem sistem sağlayıcıya olan bağımlılıktır. Açık kaynaklı sistemlerin sağladığı çoklu çalışabilirlik özelliği bu sistemleri popüler hale getirdi. Bu sistemler sayesinde farklı üreticilere ait donanımlar aynı sistem içerisinde çalışabilir hale geldi.

SCADA sistemler pazarında Citect ve WonderWare oldukça sık kullanılan iki açık kaynaklı yazılım paketidir.

SCADA yazılımları temel olarak şu fonksiyonlara sahiptirler:

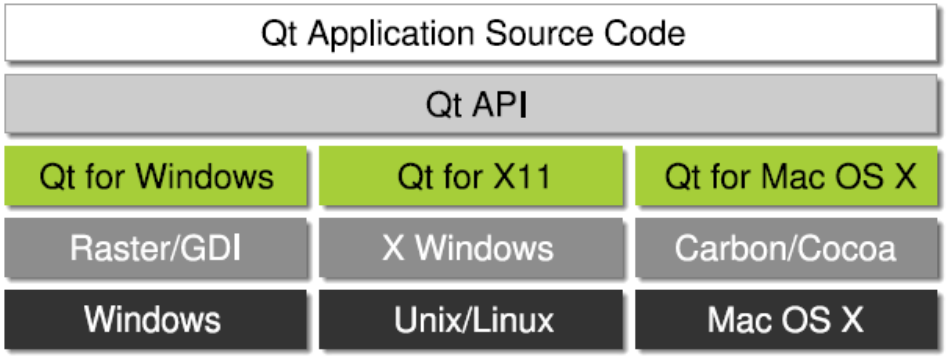
- İnsan-makine arayüzü
- Grafik ekran
- Alarmlar

- RTU ve PLC arayüzü
- Genişletilebilirlik
- Veriye ulaşım
- Veri tabanı
- Ağ haberleşmesi
- Hata toleransı
- İstemci/Sunucu dağıtık işlem kabiliyeti

5. QT C++ KÜTÜPHANESİ

Qt, görsel arayüzün, kod üzerinde değişiklik yapmaksızın farklı platformlar üzerinde çalışırılabilmesini amaçlayan bir görsel kütüphane olarak Trolltech firması tarafından geliştirilmeye başlandı. Ancak zamanla eklenen yeni kütüphanelerle büyüyen proje; basit bir görsel kütüphane olmanın ötesine geçerek oldukça geniş bir yazılım geliştirme çerçevesi haline geldi. 2009 yılında Trolltech firmasının NOKIA tarafından satın alınmasıyla, Qt mobil dünyada bir standart haline gelmeye başladı. Google, Adobe, Skype, Synopsys ve MyVr, Samsung, Nokia gibi şirketler ürünlerinde Qt' yi yoğun olarak kullanmaktalar. Qt' nin sahip olduğu kolay anlaşılır API, farklı platformlarda çalışabilme, ürün desteği gibi avantajları sebebiyle, Türkiye'de TÜBİTAK, HAVELSAN, ASELSAN gibi birçok kurum da projelerinde Qt'yi kullanmaktadır.

Qt şu anda Windows95 ile Windows7 arasındaki versiyonlar, Mac OS® X, Linux ve gömülü Linux, AIX, BSD/OS, FreeBSD, HP-UX, IRIX, NetBSD, OpenBSD, Solaris, Tru64 UNIX sistemlerini desteklemektedir(Şekil 5.1). Qt ile geliştirilen arayüz, bu platformların hepsinde aynı görünüme sahip olacaktır. Bu sayede uygulamaları farklı platformlara geçirilmesi için harcanan iş gücünden ciddi manada bir tasarruf edilmiş olur. Qt, ilk zamanlarda ticari bir ürün olarak sunulmuştu. Sadece ticari lisansa sahip kullanıcılar Qt yi kullanarak ticari uygulamalar geliştirebiliyorlardı. Qt nin sunduğu diğer bir lisanslama tipi olan GPL ile, ticari ürünler geliştirilemiyordu. Qt, 2009 yılında LGPL lisansı ile de kullanılabilir hale geldi. Böylece, herhangi bir ücret ödemeden Qt ile ticari uygulamalar da geliştirmek mümkün hale geldi.

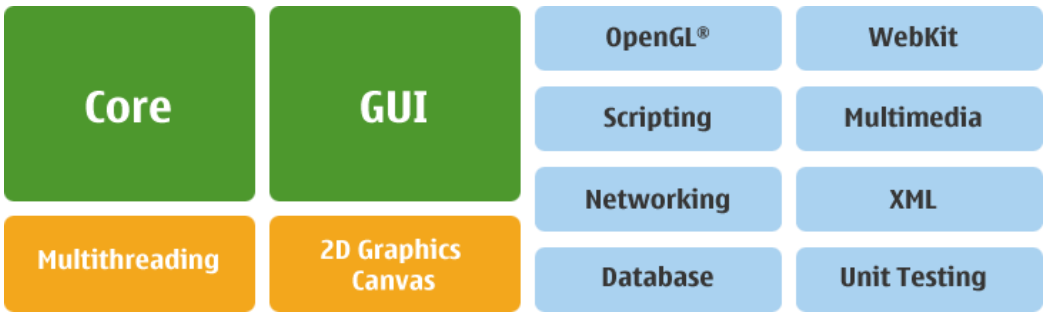


Şekil 5.1: Desteklenen Platformlar Üzerinde Qt Mimarisi[12]

Qt sahip olduğu görsel kütüphanelerin yanında veritabanı, ağ, çoklu ortam, xml, web gibi çeşitli kütüphaneler de sahiptir. Özellikle veritabanı uygulamaları geliştirenler için büyük kolaylık sağlamaktadır. Qt ile ilgili örnekleri derlemek ve nasıl işlediğini anlamak için C++ bilgisinin olması gerekmektedir. Qt içerisinde birçok araç vardır. Qt Designer, arayüz tasarlama işlemlerini gerçekleştirmektedir. Qt Designer, daha sonradan yenilerini de ekleyebileceğimiz çok sayıda parçacık (widget) sunar. Bu parçacıklar, sürükle bırak metoduyla arayüz üzerine arzu edildiği şekilde dizilebilir. Tasarım istenilen hale geldiğinde, arayüz "ui" uzantılı bir form dosyası olarak kaydedilir. Qt bu form dosyasını kullanarak, derleyicinin anlayabileceği bir c++ kodu üretir. Diğer bir araç olan "Qt linguist" ile uygulamalar kolayca çok dilli hale getirilebilir. Yazılımların uluslararası başarısı için bu araç oldukça önemli bir hale gelmiştir. Qt 'nin dil konusunda sağladığı esneklik tercih edilmesinin önemli sebeplerindendir. Açık kaynak lisansına sahip uygulamaların en büyük eksiği dokümantasyon yetersizliğidir. Qt ise bu konuda mükemmel denebilecek kadar iyidir. "Qt Assistant" yardım aracı her türlü kütüphanenin kullanımı hakkında bilgi verir ve kullanımı örneklerle zenginleştirir.[13]

Diğer bir araç olan "qmake" ise Qt kütüphanesinin derlenmesini sağlar. Qt uygulamasının yazıldığı c++ dosyasını, "ui" dosyasını ve kaynakları "pro" uzantılı bir dosya aracılığıyla derler. "qmake" aracı kullanıcıları birçok şeyi tekrar tekrar yazıp dosyayı derlemeye çalışmaktan kurtarır ve uzun bir makefile oluşturur. Derleyici de bu "makefile" ı kullanarak uygulamayı derler.

Qt Windows ortamında Visual Studio ile çalışılırken “Qt Visual Studio Integration” aracını sunar. Bu araç “Qt designer” ı Visual Studio’nun içine gömerek; “Qt designer”ın bir eksiği denebilecek, “direk olarak kod yazamama” sıkıntısını ortadan kaldırır. Ayrıca “Qt Visual Studio Integration” komut satırından “qmake” çalıştırılarak uygulamayı derleme zahmetine de son verilir. Qt içerisinde binlerce sınıf barındırır. Bu hazır sınıflar sayesinde çok az kod yazarak, oldukça verimli ve gelişmiş uygulamalar yazılabilir. Bu sınıfların başlıcaları QtCore, QtGui, QtNetwork, QtOpenGL, QtSql dir. Bu temel sınıfların altında yüzlerce farklı sınıf vardır(Şekil 5.2).

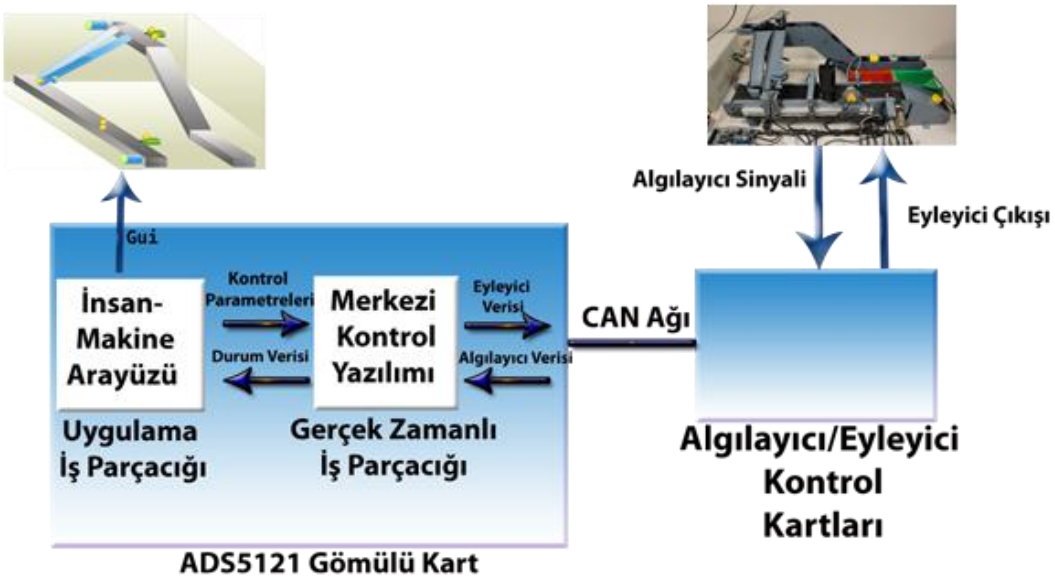


Şekil 5.2: Qt’nin Barındırdığı Temel Kütüphaneler

Qt nin bilinmesi gereken önemli mekanizmalarından biri sinyal-yuva (signals- slots) mekanizmasıdır. Signal-slot mekanizması bir çeşit “callback” mekanizması olmasına rağmen, kendine özel birçok avantajı vardır. Bu eşsiz mekanizma her şeyin başında kullanım kolaylığı sağlar. Oluşturulan nesneler arasında iletişimi sağlarken bunu tip eşleşme sorunları çözerek yapar. Bunun için signal-slot mekanizması “tip güvenli” (type- safe) dir. Bir nesne bir sinyal verdiğinde, bu sinyalin kime gideceğini bilmesi gerekmez. Sinyali alacak olan nesnenin de sinyalin kimden geldiğini bilmesi gerekmez. Bu durum, encapsulation (sarma) yapısının oluşturulmasını sağlar. Bir sinyali, birden çok nesne alabilir; ya da birden çok sinyali bir nesne alabilir.

Qt, yakın zamanda kendi IDE(integrated Development Enviroment) sini sunmaya başlamıştır. Yazılımcılara, görsel olarak kod yazma kolaylığı sağlayan bu IDE(Qt Creator), Visual Studio ve Eclipse kullanımına alternatif olabilecek niteliktedir.

6. UYGULAMA TASARIMI



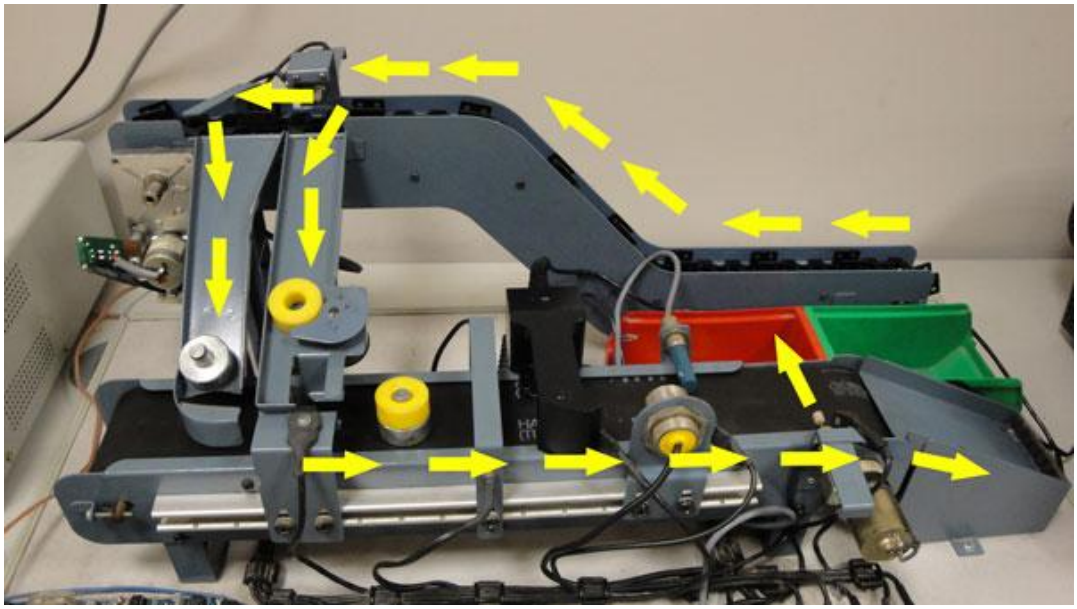
Şekil 6.1: Uygulama Tasarımı

Uygulama; üzerinde gerçek zamanlı Linux çalışan gömülü kart kullanılarak, endüstriyel kontrol eğitim setinin, kontrol edilmesi ve bunun bir arayüz ile kullanıcıya bildirilmesinden oluşuyor(Şekil 6.1). Burada merkezi kontrol yazılımının, sisteminde işletme hataları oluşmaması için gerçek zamanlı olarak çalışması gerekiyor. Yapılan işlemlerin hassasiyet derecelerini gözönünde bulundurduğumuzda, gömülü karta bir algılayıcı mesajı gelmesi ile kartın gerekli mesajı üretip hat üzerinde yayınlaması arasında geçen süre 500 ms'ye kadar tolere edilebilir. Eğer bu süre daha fazla uzarsa sistem hatalı olarak çalışmaya başlayacaktır.

6.1. Sistem Donanımı

6.1.1. Konveyör hattı

Konveyör hattı donanımı bir Endüstriyel Kontrol Eğitim Seti(ICT) ‘nden oluşuyor. Bu eğitim seti endüstride kullanılan büyük sistemlerdeki sınıflandırma, dizme ve inceleme gibi temel fonksiyonlara sahiptir. Paketleme, malzeme işleme ve kontrolü gibi temel işlemlerin yapılabilmesini sağlar. Parçaların, ilkinden ikincisine kızak kanallar ve lineer-döner bobinler yardımıyla geçmesine imkan veren iki adet konveyör hattı barındırır(Şekil 6.2). Konveyör hattı üzerine monte edilmiş olan endüstriyel algılayıcılar hat üzerindeki parçaların konumunun ve cinsinin belirlenmesini sağlar. Eğitim seti kişisel bilgisayar, PLC veya mikrodenetlecilerle kontrol edilebilir.

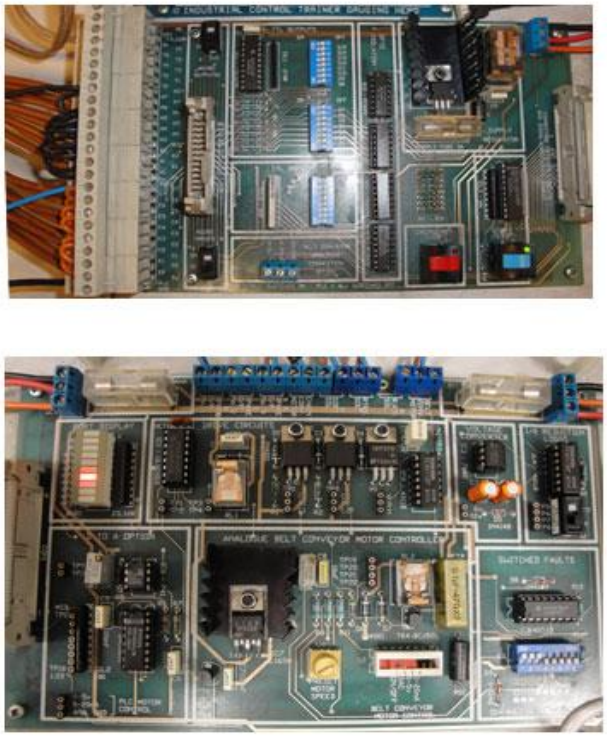


Şekil 6.2: ICT Üzerinde Nesnelerin Hareketi

Normal kullanımda, halkalar ve pimler, iki adet kızıl ötesi algılayıcının konuşlandırıldığı sınıflandırma bölgesine zincir konveyör tarafından çıkarılırlar. Burada algılayıcılardan gelen veriler kullanılarak, halkalar sınıflandırma bobini tarafından ilk kızak kanalına itilir. Bu kanal halkaları dizme bölümüne götürür. Üçüncü bir kızıl ötesi algılayıcı dizme bölümünü gözlemler. Pimler ise hat üzerinde ilerlemeye devam ettikten sonra, ikinci kızağa buradan da ikinci konveyör hattı

üzerine düşerler. İkinci konveyör hattı üzerinde yoluna devam eden pim, dizme bölümünde bekleyen halkanın altından geçerken, birbirlerine takılarak birleşirler. Birleşen halka/pim hat üzerinde ilerlemeye devam eder. Bant konveyör hattının ortasında bulunan dört algılayıcı, konveyör üzerindeki nesnenin pim, halka ya da pim/halka birleşimi olup olmadığının tespit edilebilmesini sağlar. Bu algılayıcılar endüktif, kapasitif, infrared hüzme ve yansıma tip algılayıcılardır. Kontrol yazılımı, bu algılayıcılardan gelen verilere göre nesnelerin tipini tanımlar. Beşinci kızıl ötesi algılayıcı bant konveyörün sonunda, lineer bobinin yanında konumlandırılmıştır. Bu algılayıcı, pim ve halkanın düzgün olarak birleşip birleştirilmediğinin anlaşılabilmesini sağlar. Eğer birleşme düzgün olmamışsa, lineer bobin nesneyi bantın dışına iter.

Arayüz kartı üzerinde , açma/kapama anahtarı, acil duruş anahtarı, optik izolasyon, güç bağlantıları ve farklı tipte denetleyicilerin bağlanabileceği devreler bulunur. Çıkış sürücüleri ve veri toplama elemanları iki ayrı kart üzerinde toplanmışlardır.(Şekil 6.3)

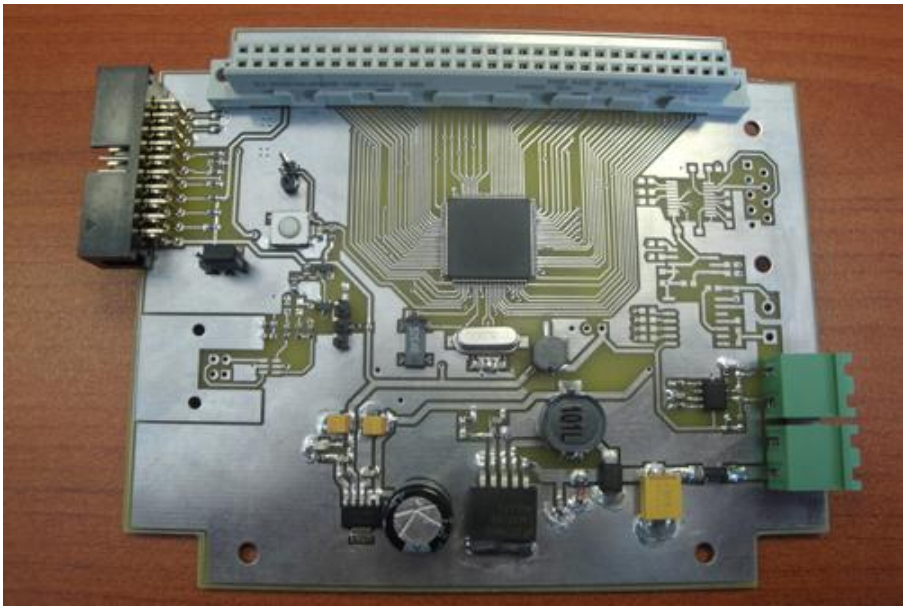


Şekil 6.3: Arayüz Kartı

6.1.2. Algılayıcı ve eyleyici kontrol kartları

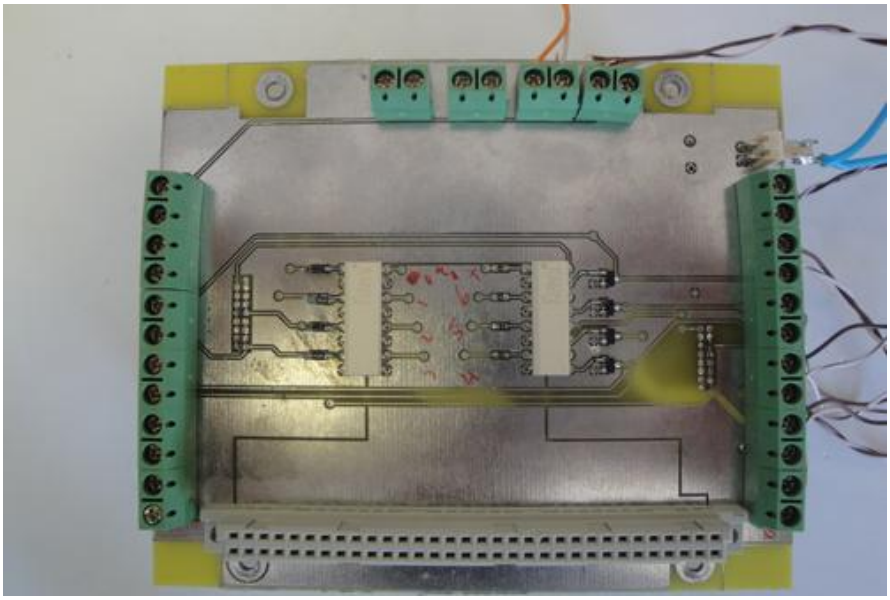
Algılayıcı ve Eyleyici kartlarının her biri, modüler iki ayrı karttan oluşmaktadır. Mikrodenetleyicinin üzerinde bulunduğu "cpu kartı" algılayıcı ve eyleyici kontrol kartlarının ikisinde de bulunur.

"Cpu kartı" genel amaçlı bir karttır; mikrodenetleyicinin farklı çevre birimlerle haberleşmesini kolaylaştıran modüler bir yapısı vardır(Şekil 6.4). Kart üzerindeki sokete farklı amaçlar için tasarlanmış devreler bağlanabilir. CAN haberleşme arayüzü bu kart üzerinde bulunur.



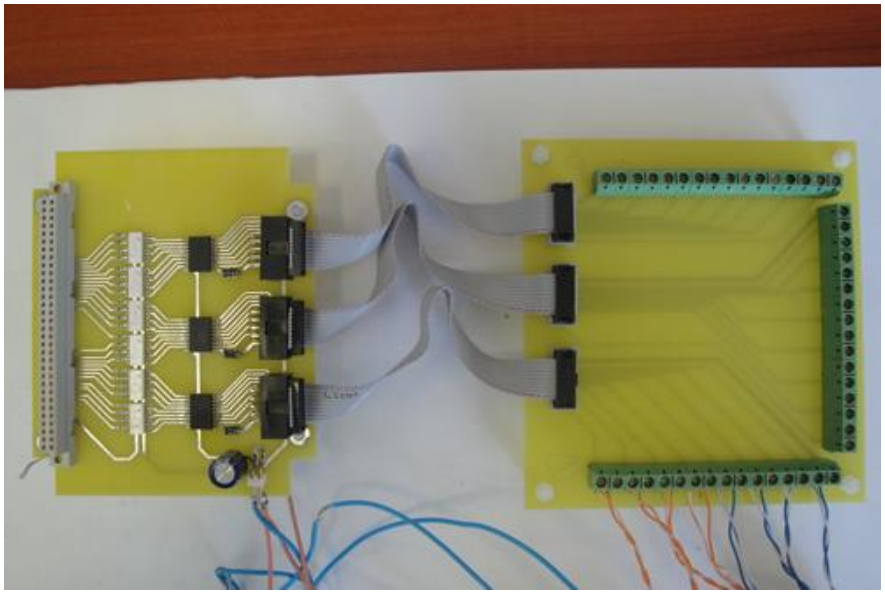
Şekil 6.4: CPU Kartı

Algılayıcı kontrol kartı; cpu kartı ve algılayıcı haberleşme arayüz kartından oluşur(Şekil 6.5). Algılayıcı haberleşme arayüz kartı, ADC ler ile gelen sinyalleri işler ve bunların CAN mesajı haline getirilmesini sağlar.



Şekil 6.5: Algılayıcı Haberleşme Arayüz Kartı

Eyleyici kontrol kartı; cpu kartı ve eyleyici kontrol arayüz kartından oluşur. Eyleyici kontrol arayüz kartı üzerinde, gelen CAN mesajlarına göre motorları ve bobinleri sürmeyi sağlayan elektronik arayüz bulunur(Şekil 6.6).



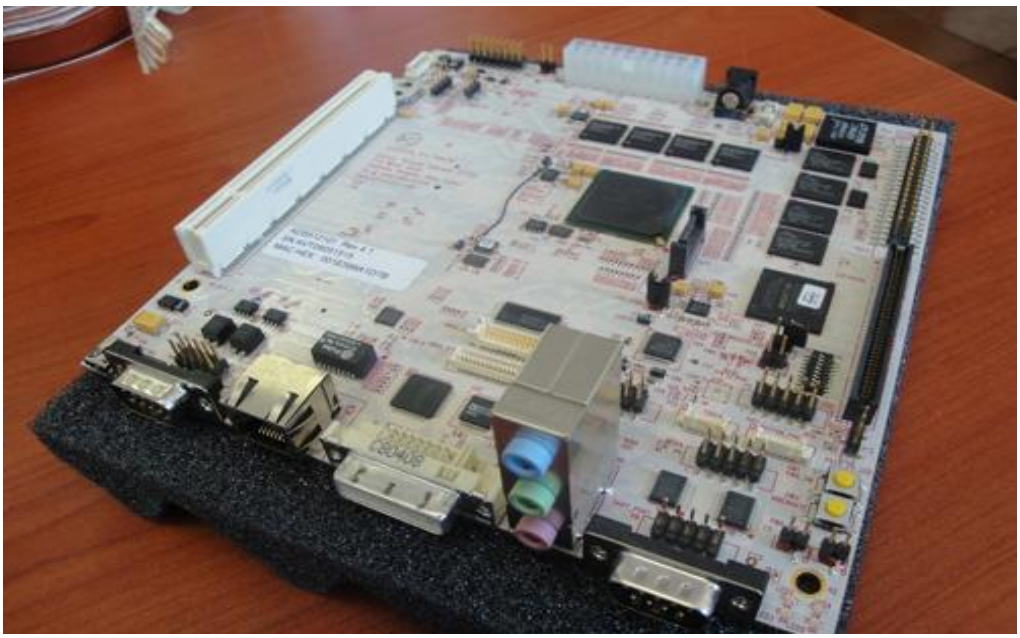
Şekil 6.6: Eyleyici Kontrol Arayüz Kartı

6.1.3. Merkezi kontrol kartı

Merkezi kontrol kartı, Siliconktx firması tarafından üretilmiş olan ADS512101 tipinde bir karttır. Kartın genel özellikleri:

- Freescale Mpc5121 işlemci tabanlı, mini-ITX standardında
- Kart üzerinde 512 Mb DDR SDRAM,
- NOR FLASH,
- NAND FLASH,
- (2) 4 kablolu RS232 portu,
- 2 CAN portu,
- USB 2.0,
- 10/100 Ethernet,
- Ses giriş/çıkış/mik,
- SATA ve PATA sürücü desteği,
- PCI,
- Micro-SD,
- 24bpp grafik,
- standard ATX güç kaynağı veya 5 Volt prize takılan adaptör ile çalışabilir.

MPC5121 işlemci üzerinde powerpc çekirdeği yanında aynı zamanda grafik işlem çekirdeği de bulunmaktadır. Bu sayede grafik işleme performansını önemli ölçüde arttırılmıştır. Kart üzerinde CAN, I2C, Ethernet gibi çok çeşitli haberleşme arayüzleri mevcuttur. Endüstri ve otomotiv sektöründe kullanıma uygun bir karttır(Şekil 6.7).



Şekil 6.7: Merkezi Kontrol Kartı(ADS512101)

6.2. Sistem Yazılımları

6.2.1. Algılayıcı ve eyleyici kontrol kartları yazılımları

Algılayıcı kontrol kartı temel olarak, ICT'den gelen algılayıcı verilerini CAN ağ mesajlarına çevirir. Eyleyici kontrol kartı ise gelen CAN ağ mesajlarına göre ICT üzerindeki konveyör motorlarını ve bobinleri sürer.

6.2.2. Merkezi kontrol kartı yazılımları

6.2.2.1. İşletim sistemi

Merkezi kontrol kartı üzerinde, RT-Preempt yaması kullanılarak hazırlanmış 2.6.24 sürüm çekirdeğe sahip Freescale Gömülü Linux işletim sistemi çalışmaktadır. Bu

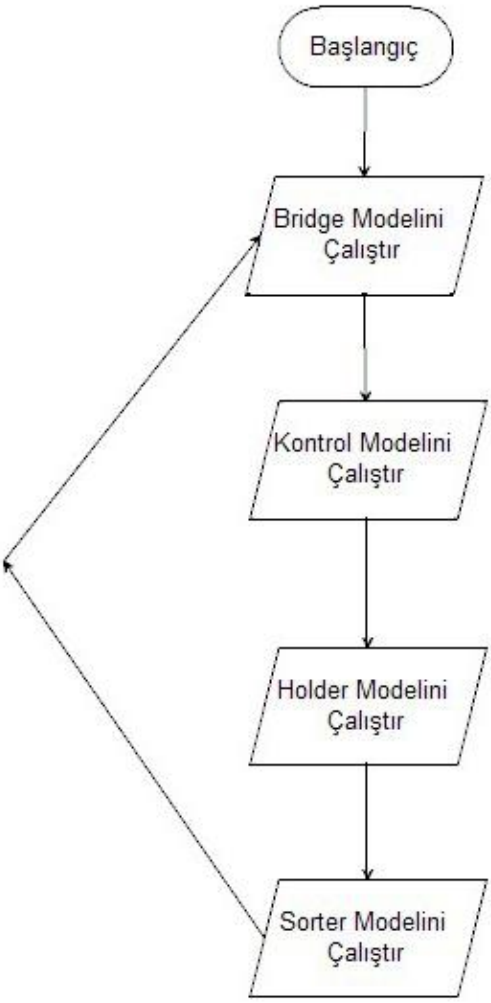
işletim sistemi üzerindeki dosya sistemi ve çekirdek, Freescale firmasının LTIB isimli aracı kullanılarak hazırlanmıştır. LTIB aracı, kart üzerinde kullanılacak olan C kütüphanelerini ve çok çeşitli uygulamaları çapraz derlemeyi oldukça kolaylaştıran bir araçtır. Çekirdeğe uygulanan yama ile karta özgü donanımların çekirdek yapılandırma ekranında ayarlanması mümkün hale gelir. LTIB aracı çıktı olarak; Linux dosya sistemi, bu dosya sisteminin istenilen formatta sıkıştırılmış halini(tar.gz, jffs2 …) , derlenmiş çekirdek imajını (uImage) ve powerpc mimarisinde ihtiyaç duyulan dtb(device tree blob) dosyalarını üretmektedir.

ADS512101 üzerinde bulunan uBoot bootloader, karta güç verildiğinde ilk olarak çalışır. uBoot, bir komut satırı üzerinden, LTIB'in üretmiş olduğu dosyaların kartın flash belleğine atılmasına olanak sağlar. uBoot argümanlarını değiştirerek işletim sisteminin kart dışında, ağ üzerinde bir dosya sitemi(NFS) kullanması da sağlanabilir.

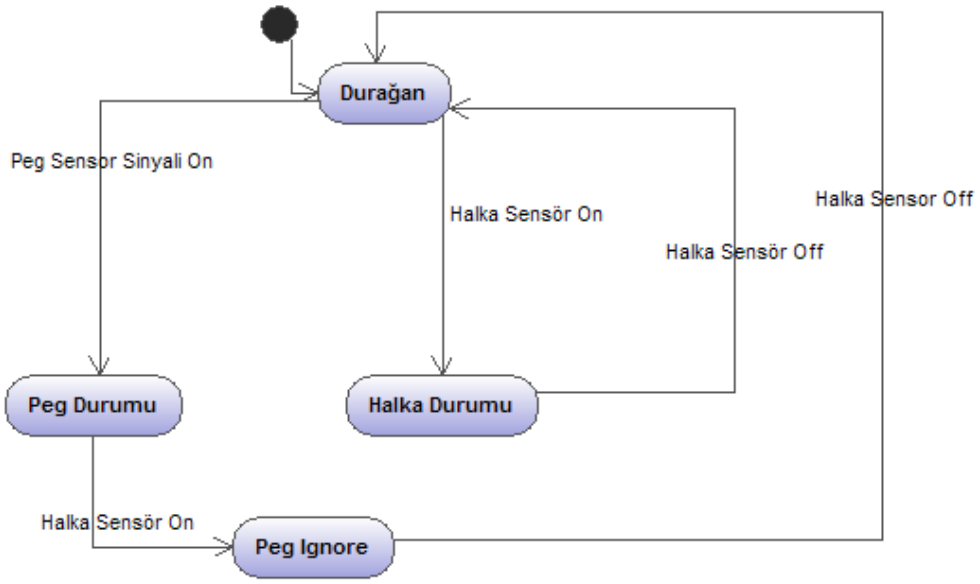
6.2.2.2. Merkezi kontrol yazılımı

Merkezi kontrol yazılımı, algılayıcıların bağlı olduğu gömülü kartlardan gelen algılayıcı mesajlarını işleyerek, eyleyicilerin bağlı olduğu gömülü kartlara mesaj gönderen bir yazılımdır.

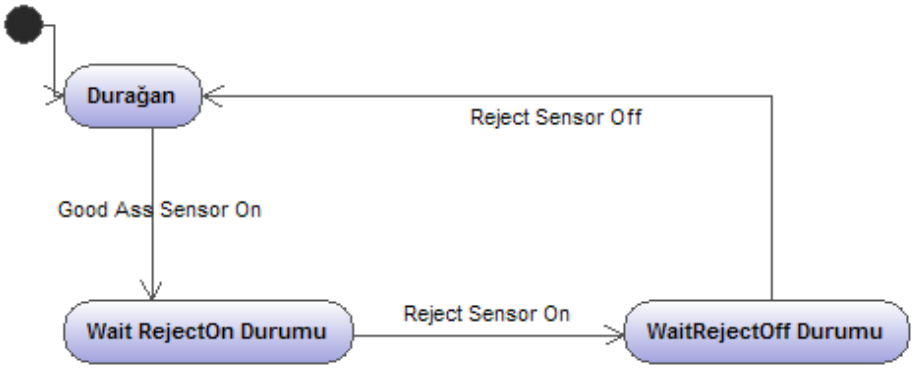
Merkezi kontrol yazılımı, dört ayrı modeli bir sonsuz döngü içerisinde çalıştırır(Şekil 6.8). Bu modeller; gelen mesajları ayrıştıran "bridge"modeli; halka ve pimlerin ayrılmasını sağlayan "sorter" modeli(Şekil 6.9); dizme bölümündeki mandalı kontrol eden "holder" modeli ve son olarak pim ve halka birleşmesinin doğruluğunu kontrol eden "kontrol" modelidir(Şekil 6.10). "Sorter" ve "kontrol" modelleri durum makinesi mantığıyla çalıştırılır.



Şekil 6.8: Kontrol Yazılımı Akış Diagramı



Şekil 6.9: Sorter Model Durum Diagramı



Şekil 6.10: Kontrol Model Durum Diagramı

6.3. Sistemin Gerçek Zaman Performansının Ölçülmesi

Sistemin gerçek zaman performansını ölçerken, ilk testte algılayıcı ve eyleyici kartlarından kaynaklanan gecikmeler ölçüm dışı bırakıldı. İlk önce sistemin gerçek zaman performansının ölçülmesinde sadece Merkezi Kontrol Kartı üzerindeki gecikmelerin gözlemlenmesi uygun görülmüştür. Sistemin oluşturduğu bir CAN mesajına karşılık, bu mesajın işlenerek gerekli işin yapılması esnasında geçen süre gözlemlenerek sistemin en kötü zaman performansı tesbit edilmeye çalışılmıştır. Hat üzerine basılan bir CAN mesajına karşılık olarak Merkezi Kontrol Kartının hat üzerine bastığı mesaj arasında geçen süre sistemin en kötü zaman performansının ölçülmesinde kullanılabilir. Bu süre ölçülürken gerçek zaman yaması uygulanmış ve uygulanmamış çekirdekler ayrı olarak değerlendirilmiştir. İşletim sistemi üzerinde farklı uygulamaların baskı oluşturduğu ve oluşturmadığı durumlar her iki çekirdek için ayrı olarak gerçekleştirilmiştir.

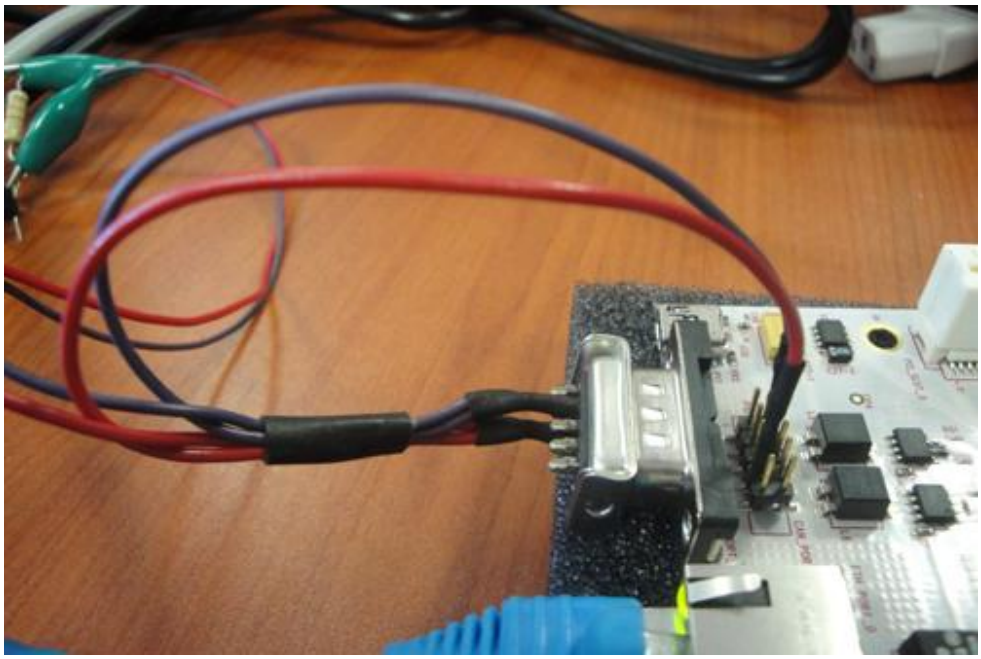
Diğer bir test düzeneğiyle ise algılayıcı ve eyleyici kartlarından kaynaklanan gecikmeler ölçülmüştür. Bu kartlardan kaynaklanan gecikmeler, merkezi kontrol kartından bağımsızdır.Ayrı olarak değerlendirilmesi gerekir.

Son test düzeneğinde, algılayıcı/eyleyici kartları ve merkezi kontrol kartlarından kaynaklanan gecikmeler hepberaber ölçülmüştür. Böylece sistemin nihai performansının ölçülebilmesi için daha net sonuçların elde edilmesi amaçlanmıştır. Bu test, algılayıcı kartının girişlerinden biri üzerinden verilen kontağın, eyleyici kontrol kartının çıkışlarından biri üzerinden gözlemlenmesi şeklinde gerçekleştirilmiştir. Bu test sürecinde de gerçek zaman yaması uygulanmış ve uygulanmamış çekirdekler ayrı olarak değerlendirilmiştir. İşletim sistemi üzerinde farklı uygulamaların baskı oluşturduğu ve oluşturmadığı durumlar da her iki çekirdek için ayrı olarak gerçekleştirilmiştir.

6.3.1. Test düzeneği

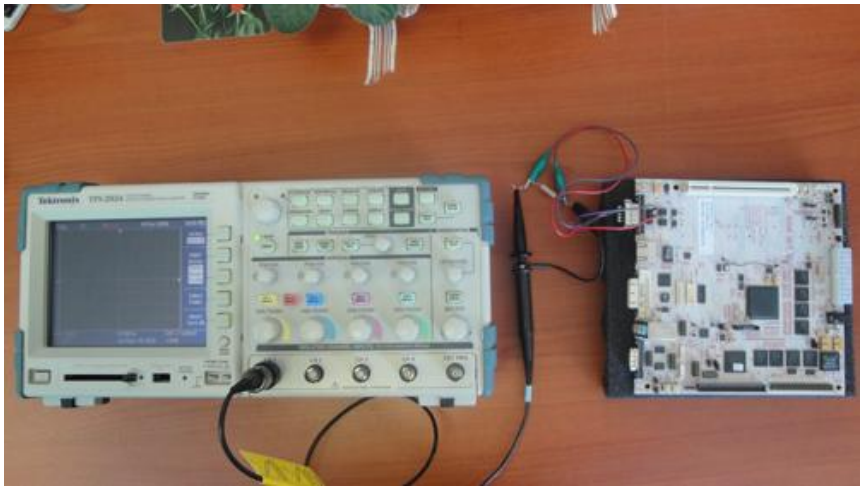
6.3.1.1. Donanımsal hazırlık

ADS512101 kartı üzerinde iki ayrı CAN ağ ara yüzü bulunmaktadır. Bu iki CAN arayüzü birbirine bağlanarak bir CAN ağı oluşturulabilmektedir. Merkezi Kontrol Yazılımı “can1” arayüzünü kullanmaktadır. Merkezi Kontrol Kartı yazılımına “can0” arayüzü üzerinden mesaj gönderip, cevap verme süresine bakarak sistem üzerindeki gecikmeler, hat gözlemlenerek ölçülebilir.



Şekil 6.11: CAN Arayüzleri Bağlantısı

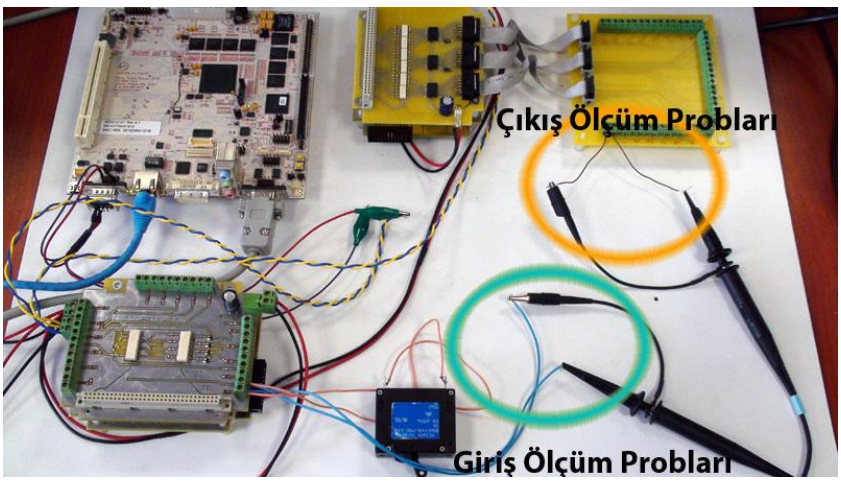
İlk test düzeneğinde, kart üzerindeki “can0” ve “can1” ara yüzlerinin 2 ve 7 nci pinleri(CAN_LOW pini ve CAN_HIGH pini) düz olarak birbirine bağlanmıştır(Şekil 6.11). Yine 2 ve 7 nci pinlerden ayrı kablo çıkışları alınarak bitirme direncinin ve osiloskop bağlantılarının bu çıkışlar üzerinden yapılabilmesi sağlanmıştır(Şekil 6.12). Dolayısıyla osiloskop ile CAN hattı gözlemlenebilir durumdadır.



Şekil 6.12: Test Düzeneği-1

İkinci test düzeneğinde ise algılayıcı ve eyleyici kontrol kartlarından kaynaklanan gecikmelerin tespit edilmesi amaçlanmıştır. Bu amaçla, merkezi kontrol kartı sistemden çıkartılarak, algılayıcı kontrol kartına bir giriş sinyali verilmesiyle, eyleyici kontrol kartının çıkış gerilimi vermesi arasında geçen süre gözlemlenmiştir.

Üçüncü test düzeneğinde, giriş/çıkış referanslı ölçümler için algılayıcı ve eyleyici kartları da ilk test düzeneğine dahil edildi(Şekil 6.13). Ve osiloskopun probları algılayıcı kartının girişine ve eyleyici kartının çıkışına bağlandı. Algılayıcı kontrol kartı üzerinden manüel olarak kontak verebilmek için girişe ayrıca bir anahtar bağlandı.



Şekil 6.13: Test Düzeneği-3

6.3.1.2. Yazılımsal hazırlık

Normalde Merkezi Kontrol Kartının konsoluna seri port üzerinden ulaşılmaktadır. Bu konsol üzerinden Merkezi Kontrol uygulaması başlatılmıştır. Karta "can0" arayüzü üzerinden mesaj gönderebilmek için Ethernet ağına bağlı herhangi bir bilgisayardan "telnet 10.18.2.105" komutu çalıştırılarak farklı bir konsol açılabilir. Bu sayede, bu konsol üzerinden karta "canwrite[1]" uygulamasını kullanarak Algılayıcı Kontrol Kartı' nın göndereceği mesajlardan biri manüel gönderilebilir. Örnek bir prosedür Tablo 6.1'teki gibidir:

Tablo 6.1: Test CAN mesajı gönderme prosedürü

```
[root@freescale apps]# ./canwrite can0 00000020#01

Bu komut sonrasında seri konsol üzerinde şu çıktıları gözlemleriz:

Reject sensor data received: 1
 reject sensor:
gonderilen 16 byte
10000011  [1] 01
```

Tablo 6.1' deki prosedür uygulandığında, hat üzerine ardı ardına iki CAN mesajı basıldığı görülür. Bunlardan ilki "id" si "*00000020" ve* "data" sı "01" olan "reject sensor" e ait algılayıcı sinyal verisidir İkincisi ise Merkezi Kontrol Yazılımının gelen mesaja karşılık ürettiği "reject actuator" bobinini harekete geçirecek olan CAN mesajıdır. Ölçümlerimizde temel aldığımız süre bu iki mesaj arasında geçen süredir.

Merkezi Kontrol Bilgisayarının ağır sistem yükü altında nasıl çalıştığı ve Merkezi Kontrol Yazılımının bu durumdan nasıl etkilendiğinin tespit edilmesi de oldukça önemlidir. Gömülü Linux sistemi üzerinde ağır sistem yükü sağlamak için hazır bir araç kullanıldı. Amos Waterland tarafından geliştirilmiş olan "Stress" isimli bu araç

[1] CAN hattına mesaj göndermeyi sağlayan basit bir uygulama

sistem üzerinde isteğe göre farklı sayılarda işlemci ağırlıklı, giriş/çıkış ağırlıklı veya bellek kullanımı ağırlıklı işlemler oluşturabilmektedir. Yapılan testlerde 75 işlemci ağırlıklı, 75 giriş/çıkış ağırlıklı ve 128 mb kullanan 3 adet bellek ağırlıklı işlem, Merkezi kontrol Yazılımı ile beraber çalıştırılmıştır.

Master Kontrol Yazılımının RT-Preempt yaması uygulanmış çekirdek ile gerçek zamanlı olarak çalışabilmesi için kod üzerinde de bazı değişikliklerin yapılması gerekmektedir. Yazılım gerçek zamanlı bir uygulama olarak çalıştırılmak isteniyorsa, "main" fonksiyonu içinde Tablo 6.2'daki eklemenin yapılması gerekmektedir:

Tablo 6.2: Uygulamanın gerçek zamanlı çalışması için eklenen kod

```
#include <sched.h>
#include <sys/mman.h>
/*PREEMPT_RT, çekirdek taskletleri ve kesme işleyiciler için  varsayılan önceliği "50"olarak ayarladığı için, bu uygulamada "49" kullanılıyor.*/
#define MY_PRIORITY (49)
/*Hatasız şekilde erişimi garanti edilmiş maksimum yığın büyüklüğü*/
#define MAX_SAFE_STACK (8*1024)
/* Saniyedeki nsaniye sayısı */
#define NSEC_PER_SEC    (1000000000)
   struct sched_param param;
/*Uygulamanın gerçek zamanlı olarak çalışacağı beyan ediliyor*/
   param.sched_priority = MY_PRIORITY;
        if(sched_setscheduler(0, SCHED_FIFO, &param) == -1) {
            perror("zamanlayici ayarlanamadi");
            exit(-1);
        }
```

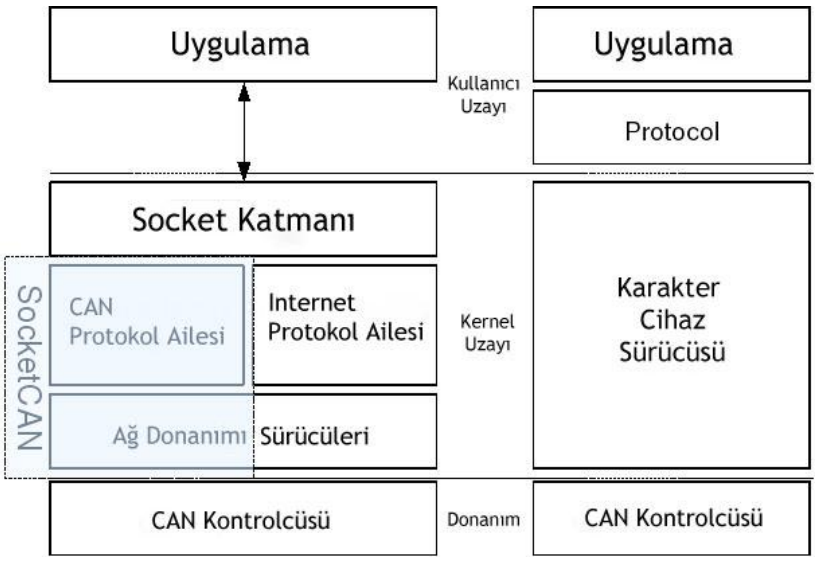
6.3.2. Gerçek zaman testi ölçümleri

Sistem üzerindeki gecikmeler iki temel kısma ayrılabilir. Bunlardan ilki Merkezi Kontrol Kartı üzerindeki gecikmeler, ikincisi ise algılayıcı ve eyleyici kontrol kartları üzerindeki gecikmelerdir. ICT' nin kendi içindeki gecikmeler çok küçük olduğu için bu çalışmada ihmal edilmiştir.

Yapılan ölçümlerde ilk önce sadece Merkezi Kontrol Kartı kaynaklı gecikmelerin gözlemlenebilmesi için, karta gönderilen ve kartın ürettiği iki CAN mesajı arasındaki süre baz alındı. Bu süreyi belirleyen faktörlerin neler olduğunu bilmek elde edilen sonuçların anlamlı olabilmesi için oldukça önemlidir.

Merkezi kontrol kartı üzerindeki esas gecikme kaynağı gerçek zamanlı olarak çalışan merkezi kontrol uygulamasının sebep olduğu gecikmedir. Şekil 6.8 de gösterildiği gibi merkezi kontrol uygulaması, modelleri belli bir periodla çalıştırır. Bu işlemin gerçekleştirilmesi bir gecikmeye sebep olur.

Merkezi kontrol yazılımı CAN mesajlarını SocketCAN api üzerinden gönderir. SocketCAN, Linux'un "Socket" yapısını kullanmaktadır. Bu API, Linux sistemlerde CAN haberleşmesinde yaygın olarak kullanılmaktadır. Çok esnek bir arayüz sağlamakla beraber bazı gecikmelere sebep olabilir(Şekil 6.14).



Şekil 6.14: SocketCAN ve Geleneksel İletişim Mimarisi

Şekil 2.2 ‘ de görüldüğü gibi kesmenin gelmesi ile ISR’ nin tamamlanması arasında geçen zaman da sistemde oluşacak gecikme süresine etki eder.

İkinci test düzeneğiyle yapılan ölçümlerde sadece algılayıcı ve eyleyici kartlarından kaynaklanan gecikmeler gözlemlendi. Merkezi Kontrol Kartı sistemden çıkartılarak algılayıcı kontrol kartından bir sinyal verildi ve eyleyici kontrol kartından bu sinyale karşılık olarak oluşan çıkış sinyali gözlemlendi.

Son test düzeneğinde ise sistemin nihai gerçek zaman performansını ölçmek için; algılayıcı kontrol kartından verilen sinyal girişine karşılık olarak bir CAN mesajı üretilmesi ve üretilen CAN mesajının Merkezi Kontrol Kartı tarafından işlenerek, eyleyici kontrol kartına yeni bir CAN mesajı göndermesi ve son olarak eyleyici kontrol kartının bir sinyal çıkışı üretmesi esnasında geçen süre gözlemlenmiştir. Bu süreyi gözlemlemek için algılayıcı ve eyleyici kontrol kartlarının giriş ve çıkışlarını izlemek yeterlidir. Yapılan bu son ölçümün ilk iki testte ölçülen değerlerin toplamına yakın süreler vermesi beklenmektedir.

Her durumda 50 adet ölçüm yapılmıştır. Yapılan ölçümlerden ikisi grafiksel olarak gösterilmiştir.

6.3.2.1. CAN mesajı referanslı ölçümler

Gelen bir mesajın Merkezi Kontrol Kartı tarafından işlenip, tekrar hatta bir mesaj basılması arasında geçen süre ölçülmüştür. Bu ölçüm gerçek zamanlı olarak çalışan Merkezi Kontrol Yazılımının en kötü zaman performansının değerlendirilebilmesi sağlar. Şekil 6.15 ‘de düzenek gösterilmiştir.

Şekil 6.15: Giren ve Çıkan Mesaj Sürelerinin Ölçümü

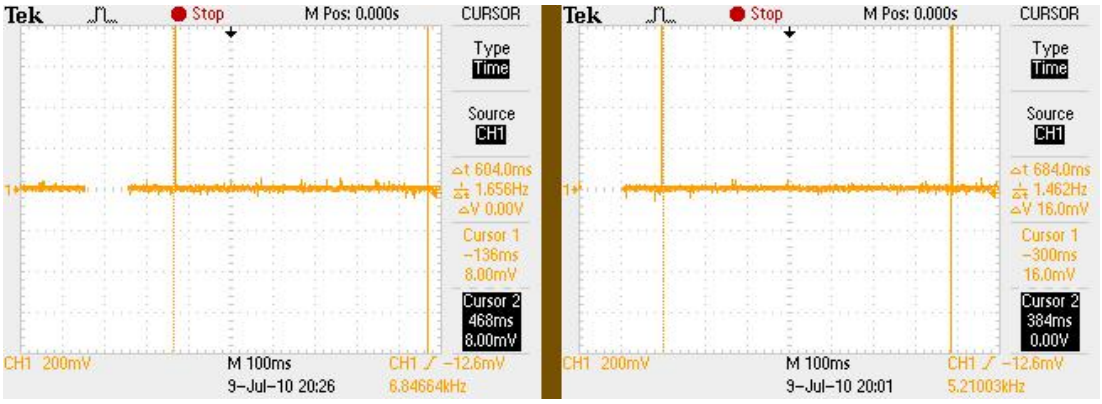
6.3.2.1.1 Gerçek zamanlı olmayan çekirdek, yüksüz

Standart Linux çekirdeği kullanılan sistemde alınan 50 ölçümde iki mesaj arasındaki gecikme süresinin 5-7,5 ms arasında olduğu gözlemlenmiştir(Şekil 6.16). En kötü durum yanıt süresi 7,5 ms olarak ölçülmüştür.

Şekil 6.16: GZ Olmayan Çekirdek, Yüksüz

6.3.2.1.2 Gerçek zamanlı olmayan çekirdek, yük altında

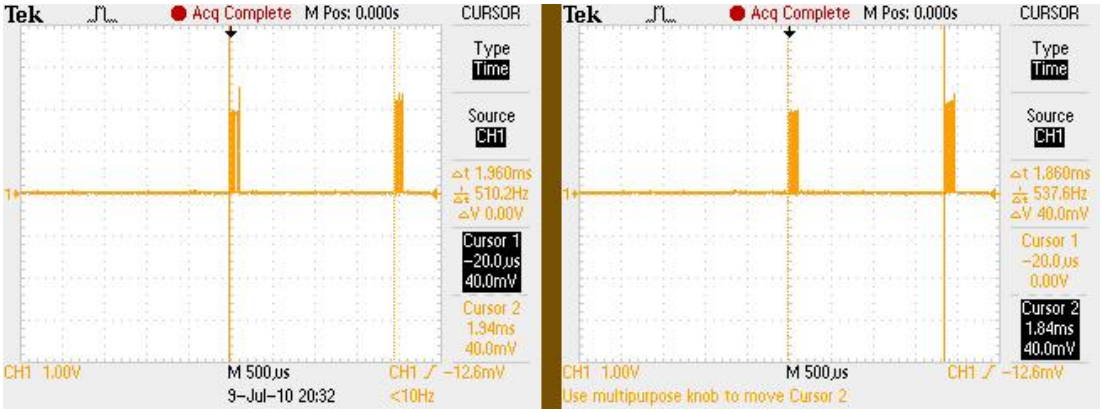
"Stress" uygulaması kullanılarak; 75 işlemci ağırlıklı, 75 giriş/çıkış ağırlıklı ve 3 bellek ağırlıklı işlem ile sistemin zorlanması sonucu gecikme 600-680 ms seviyelerine kadar çıkmıştır(Şekil 6.17). En kötü durum yanıt süresi 684 ms olarak ölçülmüştür.



Şekil 6.17: GZ Olmayan Çekirdek, Yük Altında

6.3.2.1.3 Gerçek zamanlı çekirdek, yüksüz

Sistemin yüksüz olduğu durumda iki mesaj arasındaki sürenin 1,8-1,9 ms arasında değiştiği gözlemlenmiştir(Şekil 6.18). En kötü durum yanıt süresi 1.96 ms olarak ölçülmüştür.



Şekil 6.18: GZ Çekirdek, Yüksüz

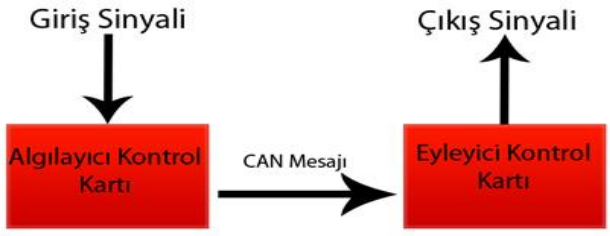
6.3.2.1.4 Gerçek zamanlı çekirdek, yük altında

Gerçek zamanlı çekirdeğe sahip sisteme "Stress" uygulaması ile yük bindirildiğinde, gecikmelerde önemli bir artış olmadı. Gerçek zaman yaması uygulanmamış çekirdeğe göre bu gecikmelerin çok daha düşük olduğu gözlemlendi(1,9-2,1 ms)(Şekil 6.19). En kötü durum yanıt süresi 2,1 ms olarak ölçüldü.

Şekil 6.19: GZ Çekirdek, Yük Altında

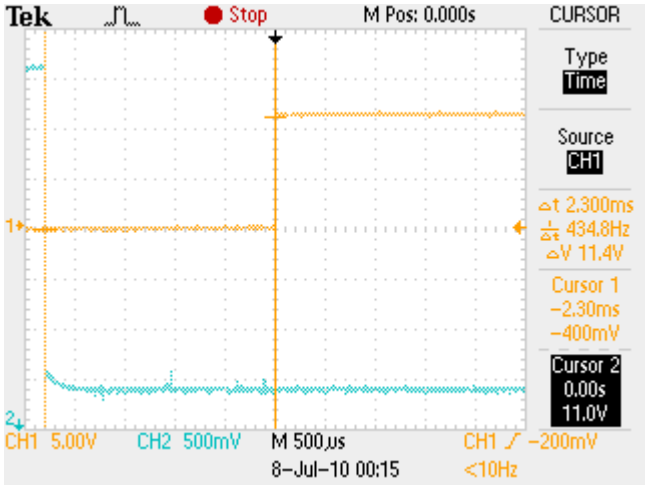
6.3.2.2. Algılayıcı ve eyleyici kartları gecikmeleri

Sadece algılayıcı ve eyleyici kartlarından kaynaklanan gecikmelerin tespit edilebilmesi için algılayıcı kartına verilen bir giriş sinyalinin, eyleyici kontrol kartında bir çıkış sinyali oluşturmasına kadar geçen süre ölçülmüştür



Şekil 6.20: Algılayıcı ve Eyleyici Kartlarındaki Gecikmeleri Ölçümü

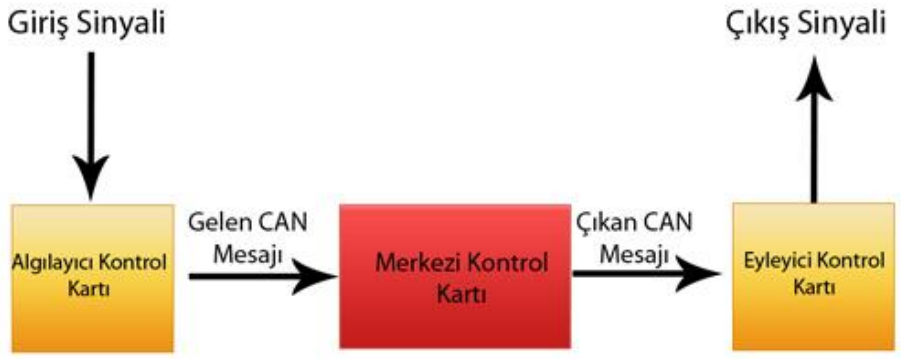
Yapılan ölçümler sonucunda algılayıcı ve eyleyici kontrol kartlarından kaynaklanan gecikmelerin 2-2,3 ms arasında değiştiği gözlemlenmiştir. En kötü durum yanıt süresi 2,3 ms olarak ölçülmüştür. Şekil 6.21'de gösterilen osiloskop çıktısında türkuaz renkli sinyal girişten verilen kontağı, turuncu renkli sinyal ise çıkıştan alınan sinyali göstermektedir.



Şekil 6.21: Algılayıcı ve Eyleyici Kartlarından Kaynaklanan Gecikme

6.3.2.3. Giriş çıkış referanslı ölçümler

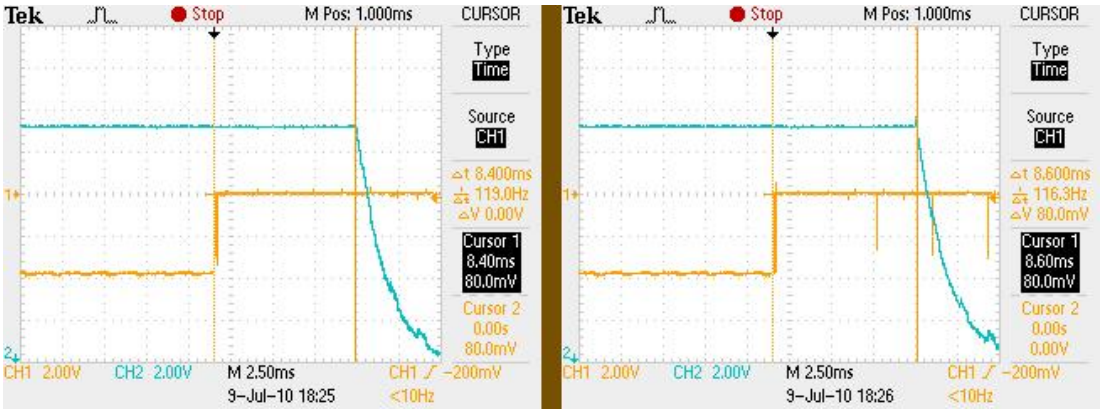
Giriş/çıkış referanslı ölçümler direk olarak ICT ‘nin giriş ve çıkışlarından yapıldığı için sistemin nihai performansı ile ilgili daha net bir sonuç alınmasını sağlamıştır. Grafiklerdeki turuncu renkli sinyal girişten verilen kontağı, türkuaz renkli sinyal ise çıkıştan alınan sinyali göstermektedir. Şekil 6.22 ‘de düzenek gösterilmiştir.



Şekil 6.22: Giriş/Çıkış Sinyal Sürelerinin Ölçümü

6.3.2.3.1 Gerçek zamanlı olmayan çekirdek, yüksüz

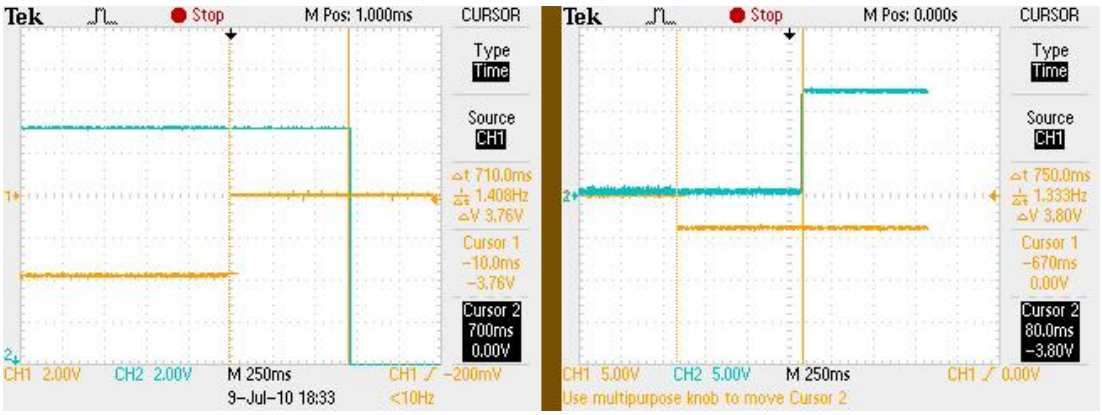
Bu durumda giriş sinyali ve çıkış sinyali arasındaki süre 8-9 ms arasında değişmektedir(Şekil 6.23). En kötü durum yanıt süresi 8,6 ms dir.



Şekil 6.23: GZ Olmayan Çekirdek, Yüksüz

6.3.2.3.2 Gerçek zamanlı olmayan çekirdek, yük altında

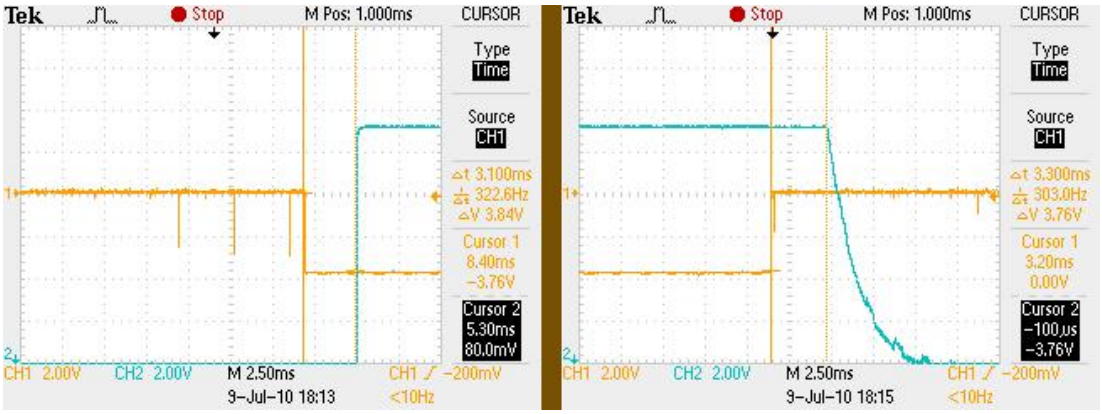
Stress uygulaması çalıştığında sistemde oldukça büyük gecikmeler gözlemlenmiştir. Gecikmeler 750 ms ye kadar çıkmıştır(Şekil 6.24).



Şekil 6.24: GZ Olmayan Çekirdek, Yük Altında

6.3.2.3.3 Gerçek zamanlı çekirdek, yüksüz

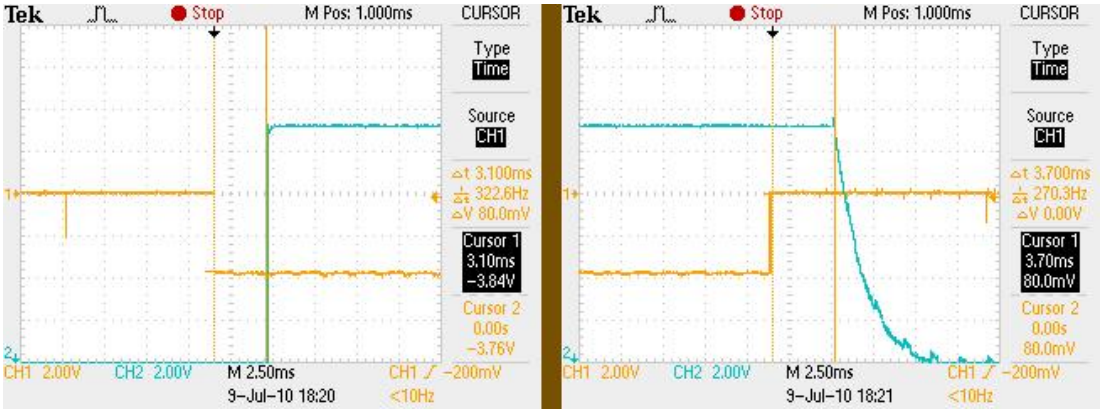
Gerçek zamanlı çekirdek, yüksüz durumda standart çekirdekten biraz daha düşük gecikmeler göstermektedir. Gecikmeler 3-3,5 ms arasında değişmektedir(Şekil 6.25).



Şekil 6.25: GZ Çekirdek, Yüksüz

6.3.2.3.4 Gerçek zamanlı çekirdek, yük altında

Stress uygulaması çalıştırıldığında gerçek zamanlı çekirdekte gecikmelerin oldukça düşük olduğu görülmüştür. Yüksüz durumdakine yakın, 3-4 ms arasında gecikmeler sistemde mevcuttur(Şekil 6.26).



Şekil 6.26: GZ Çekirdek, Yük Altında

6.3.3. Gerçek zaman testi sonucu

Yapılan ölçümler sonucunda, RT-Preempt yamasının en kötü zaman performansı üzerinde çok önemli iyileştirmeler sağladığı görülmektedir. Gerçek zamanlı bir sistemde en kötü durum gecikmesi sistem yükü arttıkça değişmemelidir. Yapılan ölçümlerde gerçek zamanlı çekirdekte, yük arttıkça gecikmelerde önemli bir değişim olmadığı görülmüştür. Bu istenen bir durumdur. Ancak kullanılan PREEMPT_RT yaması, aslında SocketCAN' in katı gerçek zamanlı olarak çalışmasını sağlamaz. Sonuçlara bakıldığında ise SocketCAN' in sistemin katı gerçek zamanlı olarak çalışmasını engellemediği görülmüştür. Bunun sebebi SocketCAN' in yumuşak gerçek zaman kısıtlarını sağlayabilecek bir yapıda olması ve kullanılan uygulama düzeneğinin katı gerçek zamanlı olarak çalışabilmesi için ihtiyaç duyulan en kötü durum gecikme değerinin oldukça yüksek olmasıdır. Eğer bir sistemin kabul edebileceği en kötü durum gecikme değeri daha düşük olursa, CAN haberleşmesinin katı gerçek zamanlı olarak çalıştırılabilmesi de mümkündür. Bir açık kaynak kod

projesi olarak davam ettirilen SocketCAN' in katı gerçek zaman destekli bir sürümü de mevcuttur. RT-Socket-CAN, Xenomai gerçek zaman uzantısında bulunan RTDM(Real-Time Driver Model)[1] tabanlı bir yapıya sahiptir. Bu sebepten standart çekirdek üzerinde CONFIG_PREEMPT_RT yaması ile doğrudan kullanılması mümkün değildir. Xenomai, RTDM üzerinden katı gerçek zamanlı sürücülerin geliştirilebilmesini sağlayan bir katman sağlar. Standart SocketCAN ise Linux ağ yığınını kullanır. Linux ağ yığını, nispeten daha fazla gecikmeye sebep olabilecek dinamik bellek ayırma metodunu kullanır. Bu durumda katı gerçek zamanlı CAN haberleşmesi, gömülü kart üzerine Xenomai gerçek zaman uzantısının eklenmesi ve RT-Socket-CAN kullanılmasıyla gerçekleştirilebilir.

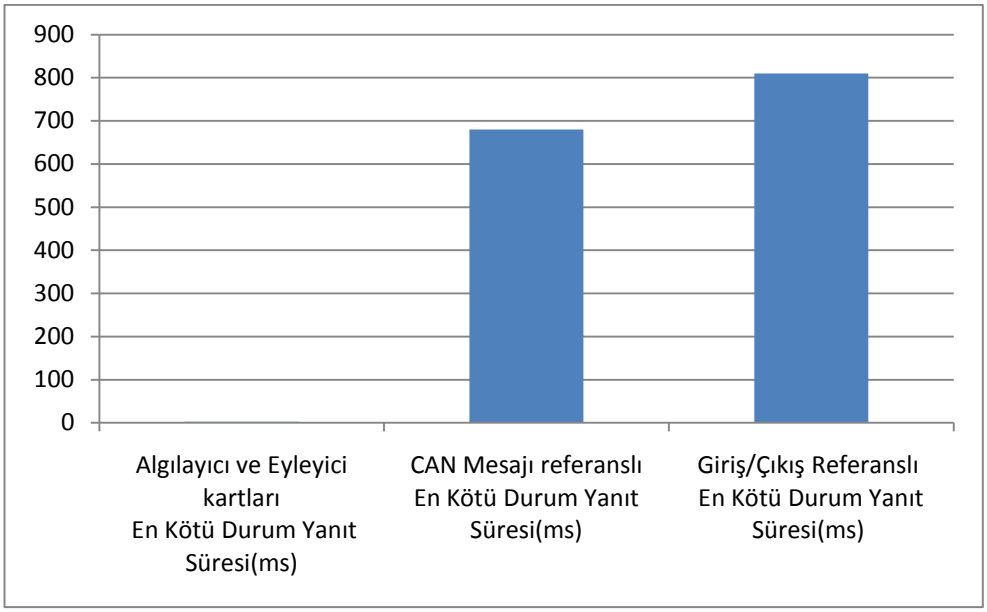
Giriş/çıkış ölçümlerinden şu görülmektedir ki; algılayıcı ve eyleyici kontrol kartları sistem üzerinde belli bir gecikmeye sebep olmaktadır. Bu gecikmenin 2-2,3 ms civarında olmasının sebebi; kartlar üzerindeki yazılımların kesme tabanlı değil, polling(gözlem) tabanlı olarak çalışıyor olmasıdır. Ancak 2,3 ms lik en kötü durum yanıt süresi oldukça deterministiktir. Bu sebepten sistemin gerçek zamanlı olarak çalışmasına engel olmaz; sadece gecikmenin en fazla 2,3 ms kadar artmasına sebep olur.

Tablo 6.3 ,Şekil 6.27, Şekil 6.28, Şekil 6.29 ve Şekil 6.30 'da elde edilen sonuçlar tablo ve grafikler halinde verilmiştir.

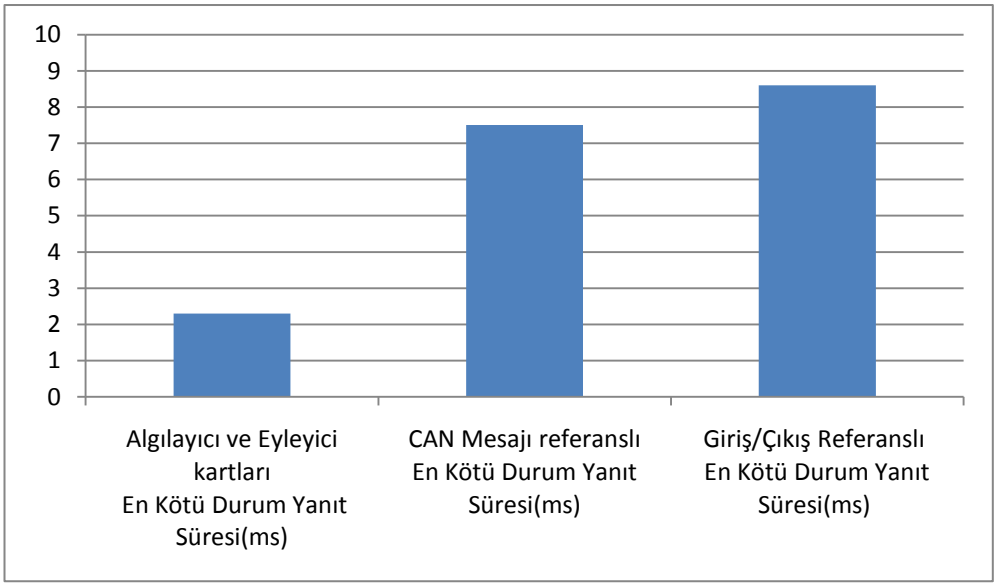
Tablo 6.3: Ölçülen En Kötü Durum Yanıt Süreleri

	Algılayıcı ve Eyleyici kartları En Kötü Durum Yanıt Süresi(ms)	CAN Mesajı referanslı En Kötü Durum Yanıt Süresi(ms)	Giriş/Çıkış Referanslı En Kötü Durum Yanıt Süresi(ms)
Standart kernel ve Stress	2,3	680	810
Standart kernel, Stress olmadan	2,3	7,5	8,6
GZ kernel ve Stress	2,3	2,1	3,7
GZ kernel, Stress olmadan	2,3	1,96	3,6

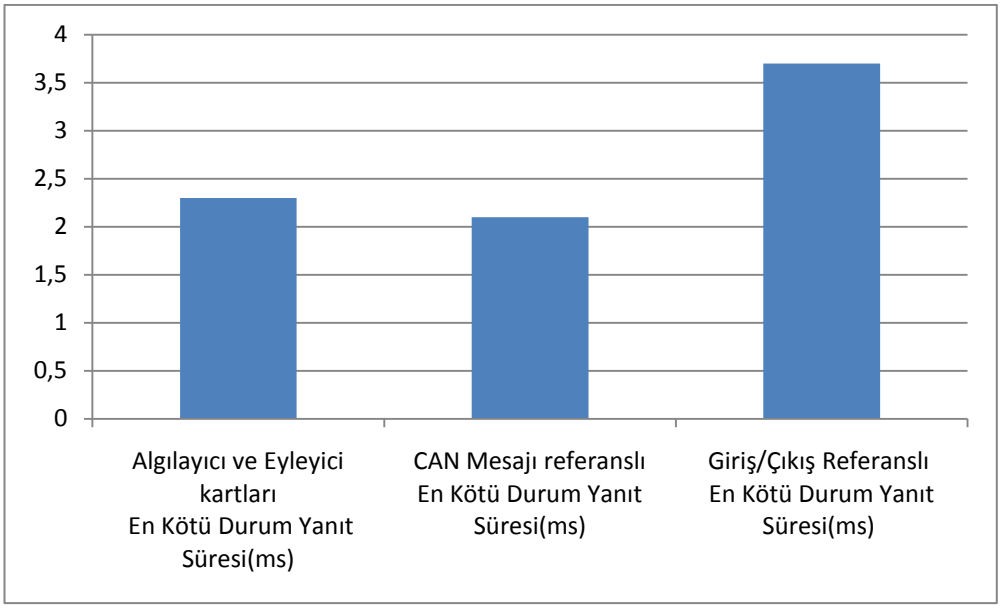
[1] Gerçek Zamanlı Sürücü Modeli



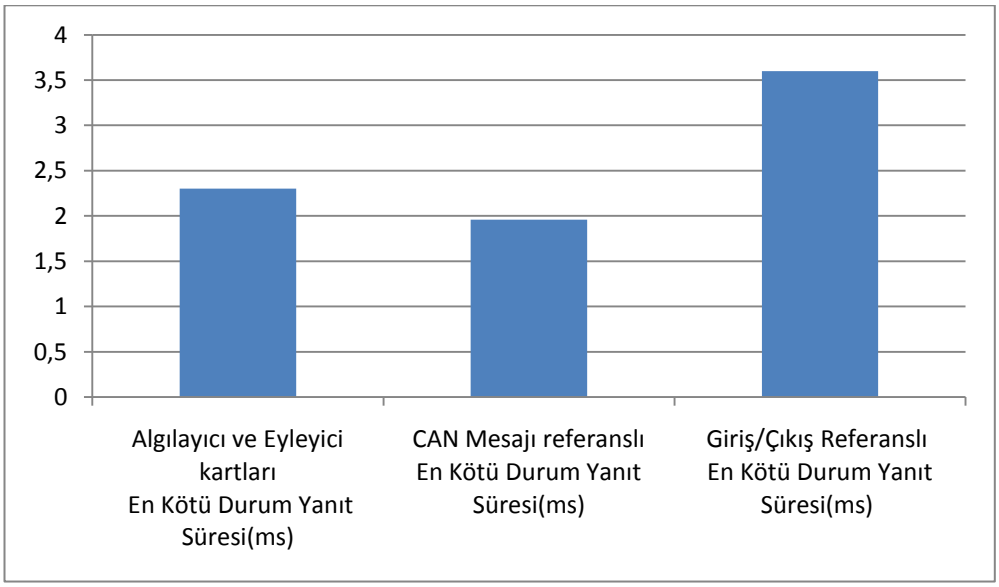
Şekil 6.27: Standart Kernel'de, Stress Uygulaması Çalıştırıldığı Durumdaki En Kötü Durum Yanıt Süreleri



Şekil 6.28: Standart Kernel'de, Stress Uygulaması Çalıştırılmadığı Durumdaki En Kötü Durum Yanıt Süreleri



Şekil 6.29: GZ Kernel'de, Stress Uygulaması Çalıştırıldığı Durumdaki En Kötü Durum Yanıt Süreleri

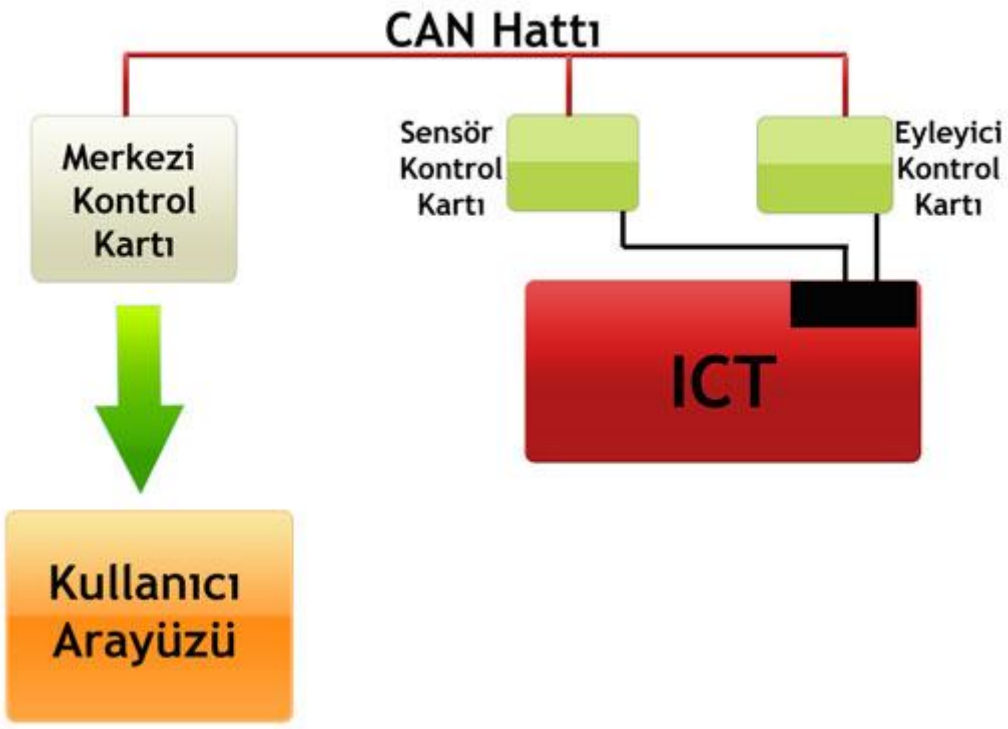


Şekil 6.30: GZ Kernel'de, Stress Uygulaması Çalıştırılmadığı Durumdaki En Kötü Durum Yanıt Süreleri

6.4. Uygulama Düzeneği

6.4.1. Uygulama düzeneğinin yapısı

Uygulama düzeneği Merkezi Kontrol Kartı(ADS512101), algılayıcı kontrol kartı, eyleyici kontrol kartı ve ICT den oluşmaktadır. Merkezi Kontrol Kartı, algılayıcı kontrol kartı ve eyleyici kontrol kartı CAN ağ protokolü kullanılarak birbirlerine bağlanmıştır(Şekil 6.31). Algılayıcı kontrol kartı ve eyleyici kontrol kartı ICT üzerindeki uygun giriş/çıkışlara bağlıdır.



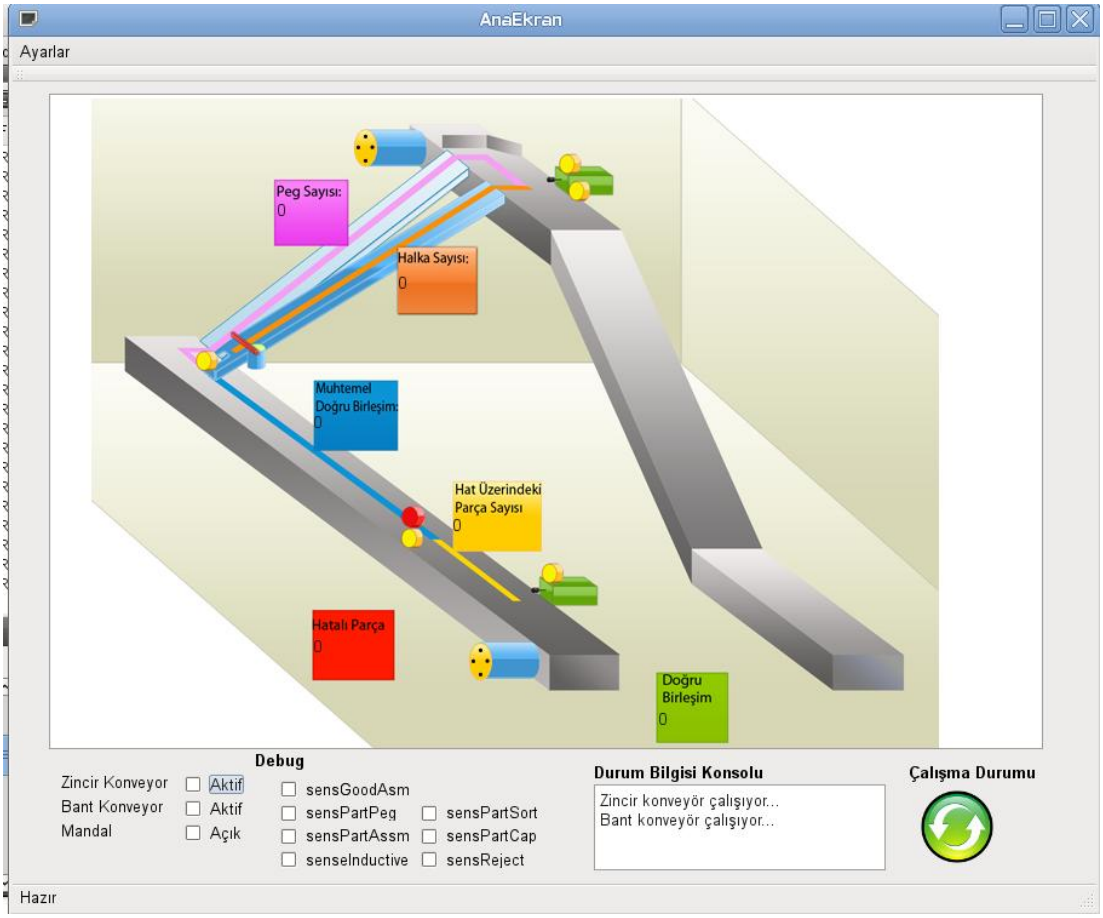
Şekil 6.31: Uygulama Düzeneği

Merkezi Kontrol Kartı üzerinde; Merkezi Kontrol Yazılımının yanısıra, insan-makine arayüzü sağlayan ayrı bir uygulama da çalışmaktadır. Bu arayüz üzerinden kanal ayarları yapılabilmektedir. Bunun yanında, sistemin çalışma anı durumu arayüz üzerinden izlenebilmektedir.

Merkezi kontrol yazılımı, gerçek zamanlı olarak çalışmasına rağmen insan makine arayüzünün böyle bir özelliği yoktur. İnsan-makine arayüzü daha düşük öncelikli bir

işlem olarak çalıştırılır. Bunun sebebi kontrol yazılımının aynı sistem üzerinde daha kritik bir işlem yürütüyor olmasıdır. Bu sayede arayüz yazılımının sisteme yüklemiş olduğu yük, sistemin çalışmasının doğruluğunu etkilemez.

Merkezi kontrol yazılımı ve insan-makine arayüz yazılımı Linux Socket yapısı üzerinden haberleşirler. Socket yapısı Linux' un ağ haberleşme alt yapısını oluşturmanın yanında IPC(işlemler arası haberleşme)'de de kullanılır. Linux Socket yapısı çift yönlü bir iletişim kanalı oluşturur.

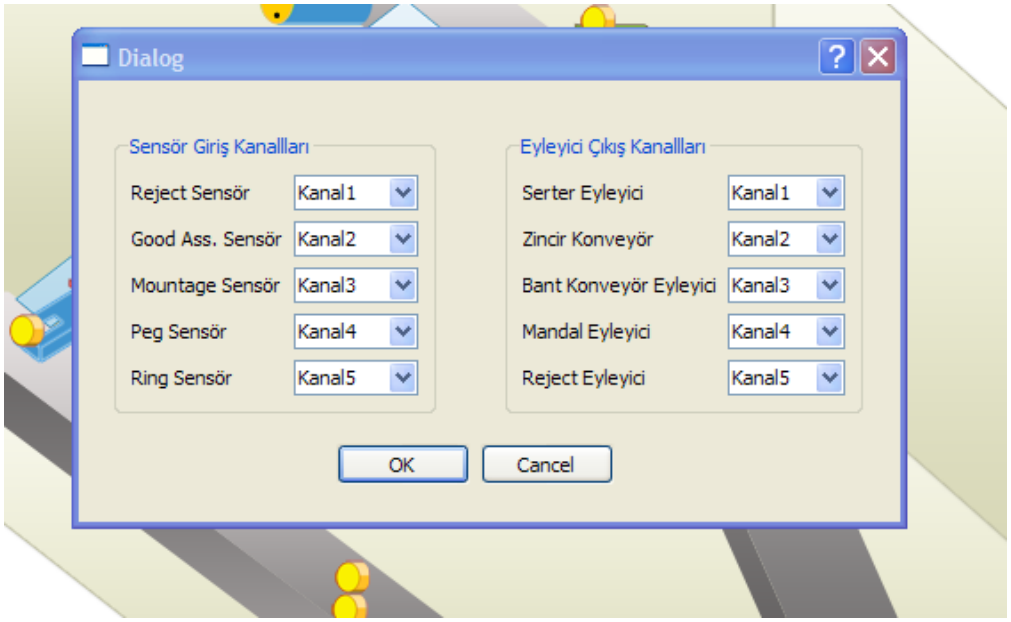


Şekil 6.32: Uygulama Kullanıcı Arayüzü

İnsan-makine arayüzü üzerinde sistemde bulunan eyleyici ve algılayıcıların durumları görselleştirilerek kullanıcıya bildirilir(Şekil 6.32). Gelen algılayıcı verileri ve eyleyicilerin durumları aynı zamanda bir konsol üzerinde loglanır. Eyleyici kumanda paneli üzerinde konveyör hatları çalıştırılabilir veya durdurulabilir. Bobinler de kontrol yazılımından bağımsız olarak el ile çalıştırılabilir. Acil bir

durumda sistemin durdurulması işlemi de arayüz üzerinden yapılabilmektedir. Bunların dışında arayüz uygulamasının kilitlenmeden doğru olarak çalışıp çalışmadığının anlaşılabilmesi için bir çalışma durumu indikatörü arayüze eklenmiştir. Yazılım kilitlenirse bunu indikatör yardımıyla anlamak mümkündür.

Kullanılacak olan giriş ve çıkış kanalları "kanal konfigürasyon" menüsünden değiştirilebilir. Bu sayede sistemin farklı giriş/çıkış kanalları kullanarak da çalıştırılabilmesi sağlanır.(Şekil 6.33)



Şekil 6.33: Kanal Konfigürasyon Menüsü

6.4.2. Uygulama düzeneğinin başarımı

Uygulama düzeneği sistem üzerinde yapmış olduğumuz testlerde olduğu gibi gerçek zamanlı Linux ve standart Linux ile yüklü ve yüksüz durumlarda çalıştırılmıştır.

Aslında ICT' nin yapısı düşünüldüğünde sistemin hatalı çalışmaya başlaması için gecikmelerin 500 ms ye kadar çıkması gerekmektedir. Sistem üzerinde daha önce yapılan ölçümlerde bu oranda bir gecikme görülmemiştir. Ancak "Stress"

uygulamasının başlatacağı işlem sayısını arttırarak, bu gecikmeleri oluşturmak mümkündür.

Gerçek zamanlı ve standart çekirdek üzerinde "Stress" uygulamasının çalıştırılmadığı durumlarda sistemin çalışmasında herhangi bir problem görülmemektedir. İnsan-makine arayüzü üzerinden sistemin çalışması problemsiz şekilde gözlemlenmekte ve ICT birleştirme ve ayıklama işlerini doğru olarak yapmaktadır.

Standart çekirdek üzerinde, "stress" uygulaması 75 işlemci tabanlı, 75 giriş/çıkış tabanlı ve 3 bellek tabanlı işlem başlatacak şekilde ayarlandığında gecikmelerin önemli oranda arttığı gözlemlenmiştir. Algılayıcılardan gelen veriler geç işlendiği ve sorter , reject ve mandal eyleyicileri gecikmeli olarak çalıştığı için birleştirme ve ayıklama işlerinin doğru şekilde yapılamadığı görülmüştür. Aynı zamanda insan-makine arayüzünde de donmalar gözlemlenmiştir.

"Stress" uygulaması aynı parametrelerle gerçek zamanlı çekirdek üzerinde çalıştırıldığında, ICT doğru olarak birleştirme ve ayıklama işlerini yapmaktadır. Bu durumda insan-makine arayüzü üzerinde donmalar görülmektedir. Bunun sebebi aynı anda çalışan yüzlerce işlemin, işlemci zamanını paylaşmaya çalışmasıdır. İşlemci zamanının büyük bölümünü gerçek zamanlı olarak çalışan "Master Kontrol Yazılımı" almaktadır. Bu durum diğer işlemlerin çalışmasında yavaşlamalara sebep olmakla birlikte; kritik bir görev yürüten "Master Kontrol Yazılımının" sorunsuz bir şekilde çalışmasını sağlamaktadır.

7. SONUÇ

Sistemin katı gerçek zamanlı olarak çalışmasını sağlayan RT-Preempt yaması uygulanmış Linux üzerinde, endüstriyel kontrol eğitim setinin(ICT) gözlem ve kontrolünü sağlayan, insan-makine arayüzü tasarlanmıştır. Linux'un üzerinde çalıştığı bir gömülü bilgisayara(merkezi kontrol kartı), CAN ağ mesajları ile algılayıcı bilgileri gönderilmiştir. Bu gelen mesajlar gerçek zamanlı olarak çalışan yazılım(merkezi kontrol yazılımı) tarafından işlendikten sonra ilgili eyleyicilerin kontrolü için eyleyici kontrol kartına, CAN mesajlarıyla komutlar gönderilmiştir. Bu esnada insan-makine arayüzü yazılımı aracılığıyla, merkezi kontrol yazılımından gelen veriler doğrultusunda sistemin durumu insan-makine arayüzü üzerinde gösterilmiştir.

Merkezi kontrol yazılımı; gerçek zamanlı çekirdek ve standart çekirdek kullanılarak, sistem yük altında ve yüksüz iken test edilmiştir. Yapılan ölçümlere göre; standart çekirdek yüksüz durumda, sistemin en kötü durum gecikmesi 8,6 ms iken; yük altında 810 ms' ye kadar çıkmıştır. Gerçek zamanlı çekirdek kullanıldığında ise yüksüz durumda en kötü durum gecikmesi 3,6 ms iken; yük altında 3,7 ms olarak ölçülmüştür. Bu sonuçlar göstermektedir ki gerçek zamanlı çekirdek, yük altındaki sistemdeki gecikmeleri yaklaşık 200 kat azaltmıştır. Yüksüz sistemdeki gecikmeler üzerinde az da olsa bir iyileştirme sağlamıştır. İnsan-makine arayüz yazılımı gerçek zamanlı bir işlem olarak çalıştırılmadığı için yük altında kısmi donmalar görülmüş; ancak uygulama doğru bir şekilde çalışmaya devam etmiştir.

Üzerinde birden çok yazılımın çalıştırıldığı bir platformda zaman kısıtlı bir işlemin çalıştırılması gerekiyorsa; işletim sisteminin gerçek zamanlı olması ve bu işlemin gerçek zamanlı olarak çalıştırılmasının sistemin doğru çalışabilmesi için hayati bir önem arzettiği görülmüştür. Gerçek zamanlı Linux 'un, üzerinde insan-makine arayüzü ve kontrol yazılımı bulunan böyle bir sistemi başarıyla yönetebildiği tespit edilmiştir.

Bir uygulamanın tam olarak gerçek zamanlı çalışabilmesi için uygulamaya bağlı modüllerin ve sürücülerin de gerçek zamanlı olarak çalıştırılması gerekir. Bu kısıtı olabildiğince minimize etmek için çalıştırılan gerçek zamanlı kod parçasının olabildiğince bağımsız ve basit olmasında fayda vardır. Karmaşıklık ve farklı işlemlere bağımlılık arttıkça gerçek zamanlı çalışmanın bozulma ihtimali artabilir. Gerçekleştirilen uygulamada kullanılan SocketCAN, katı gerçek zaman kısıtlarını sağlayan bir API değildir. Ancak SocketCAN' in sağladığı yumuşak gerçek zaman performansı, çalışmada kullanılan sistemin süre kısıtları oldukça büyük olduğu için sistemin katı gerçek zamanlı olarak çalışmasını bozmamıştır. Gelecek çalışmalarda SocketCAN yerine, katı gerçek zamanlı sürücü desteği sağlayan Xenomai ile beraber, katı gerçek zamanlı CAN haberleşmesi sağlayan RT-SocketCAN kullanılabilir.

KAYNAKLAR

[1] Aroca R., Caurin G., "A Realtime Operating Systems Comparison" , ***Anais do Workshop de Sistemas Operacionais***,(2009)

[2] Rick Lehrbaum, 2001, "Realtime Linux Software Quick Reference Guide", http://www.linuxfordevices.com/c/a/Linux-For-Devices-Articles/Realtime-Linux-Software-Quick-Reference-Guide(**Ziyaret tarihi: 15 Nisan 2010**)

[3] İrez Yiğit, 2009, "Gerçek Zamanlı İşletim Sistemleri Savaşları", http://www.teknovole.com/sizin-icin-sectiklerimiz/gercek-zamanli-isletim-sistemleri-savaslari(**Ziyaret tarihi: 3 Nisan 2010**)

[4] INTEGRITY, 2010, "INTEGRITY Real-Time Operating System", http://www.ghs.com/products/rtos/integrity.html(**Ziyaret tarihi: 15 Nisan 2010**)

[5] Ismael Ripoll, Pavel Pisa, Luca Abeni, Paolo Gai, Agnes Lanusse, Sergio Saez, Bruno Privat, "WP1-RTOS State of the Art", ***OCERA Organization Whitepaper***,3-5,(2008)

[6] David Beal , "Linux As a Real-Time Operating System", ***Freescale Whitepaper***,(2006)

[7] Hafsal M. Awesome,"Real-Time Operating Systems", VLSI and Embedded System Technical Library, http://vetechlib.blogspot.com/2009/10/real-time-operating-systems.html(**Ziyaret tarihi: 7 Mayıs 2010**)

[8] Markus Franke, 2007, "A Quantitative Comparison of Realtime Linux Solutions", ***Technische Universitat Chemnitz, Department Of Computer Science***, Seminar Paper,

[9] Hallinan C., "Embedded Linux Primer: A practical Real-World Approach", ***Prentice Hall***, 32-34,(2006)

[10] Bailey D., Wright E.., "Practical SCADA for Industry", ***Elseiver***, 13-14,(2003)

[11] Xenomai Dökümantasyonu, "A Tour of the Native Api", http://www.xenomai.org/documentation/branches/v2.0.x/pdf/Native-API-Tour.pdf (**Ziyaret tarihi: 16 Mayıs 2010**)

[12] Nokia, “Qt Technical Overview Whitepaper”, http://qt.nokia.com/files/pdf/qt-4.6-whitepaper (**Ziyaret tarihi: 17Mayıs 2010**)

[13] Blanchette J., Summerfield M, “C++ GUI Programming with Qt 4”, ***Prentice Hall,***(2006)

[14] Peter Laurich, 2004, “A Comparison of Hard Real Time Alternatives”, http://www.linuxfordevices.com/c/a/Linux-For-Devices-Articles/A-comparison-of-hard-realtime-Linux-alternatives/ (**Ziyaret tarihi: 2 Haziran 2010**)

[15] Paul McKenney, 2005, “A realtime preemption overview”, http://lwn.net/Articles/146861/ (**Ziyaret tarihi: 1 Haziran 2010**)

ÖZGEÇMİŞ

1982 yılında Erzurumda doğdu. İlk ve orta öğrenimini Sakarya'da, lise öğrenimini Balıkesirde tamamladı. 2000 yılında girdiği Orta Doğu Teknik Üniversitesi Elektrik ve Elektronik bölümünden, 2006 yılında Elektrik ve Elektronik mühendisi olarak mezun oldu. 2008 yılından bu yana, Kocaeli Üniversitesi Fen Bilimleri Enstitüsü, Elektronik ve Haberleşme Mühendisliği Anabilim Dalı'nda Yüksek Lisans öğreniminine devam etmektedir. 2008 yılından beri, TÜBİTAK Bilişim Teknolojileri Enstitüsünde araştırmacı olarak çalışmaktadır.