OSEK/VDX Uyumlu Katı Gerçek-Zamanlı İşletim Sistemleri için Zamanlama Mimarilerinin Değerlendirilmesi

Evaluation of Scheduling Architectures for OSEK/VDX Compliant Hard Real-Time Operating Systems

Berkay Saydam
R&D Centre
Red Pine Software
Izmir, Turkey
berkay.saydam@redpine.com.tr

Tolga Ayav
Izmir Institute of Technology
Urla- Izmir, Turkey
tolgaayav@iyte.edu.tr

***Özet*—Gelişen teknoloji ile araçların performansının artırılması, güvenlikten ödün vermeden yeni işlevler eklemek açısından önemlidir. İşlevselliği sağlamak için kullanılan yongalar ECU tasarımının ilk adımında belirlenir. Bu karar projenin kalan kısmını etkileyecektir. Donanım, yazılım tasarımı ve bunların testi otomotiv endüstrisinde uzun bir süreçtir. Bir ECU sahada kullanılmaya başladığında donanım tasarımını değiştirmek zordur. Bu nedenlerden dolayı, mikrodenetleyici seçimi maliyeti ve müşterinin istediği işlevselliği etkiler. İşlevselliği sağlayan görevler performans ve güvenlik arasında bir denge sağlar. Bu çalışmada bu konular OSEK / VDX sertifikalı işletim sistemleri üzerinde değerlendirildi ve zamanlama algoritmaları için sonuçlar sunuldu.**

Anahtar Kelimeler—zamanlama algoritmaları, katı gerçek-zamanlı sistemler, OSEK/VDX sertifikalı işletim sistemi, statik zamanlama analizi, araçlarda güvenlik

***Abstract*— Developing technology is reflected to the vehicles as well. But it brings the challenge of adding new functionalities to vehicles without compromising safety. The chips, which are used to provide the functionalities, are determined in the first step of ECU design. This decision will effect the remaining part of the development. Designing hardware and software together with testing phase is a long process in automotive industry. Changing the design of the hardware is quite costly after an ECU begins to be used in field. For these reasons, the selection of chips is directly related to cost and the functionality which should be provided to customer. Tasks, which fulfill desired functionality, provide a balance between performance and safety. These were evaluated for an OSEK/VDX certified OS and results are presented from the scheduling algorithms point of view.**

Keywords—scheduling algorithms, hard real-time systems, OSEK/VDX certified OS, static scheduling analysis, safety in vehicle

I. GİRİŞ

Günümüzde teknolojinin gelişmesi ile birçok yeni özellik araçlara eklendi. Bu özellikler araçlarda ekstra çalışma yükü oluşturmaktadır. Yeni fonksiyonaliteleri sağlayan bu özellikleri karşılayan görevler, güvenlik konularında negative bir etki oluşturmayacak şekilde doğru ayarlanmalıdır. Bu ayarlamalar sadece zamanlama algoritmalarının çok iyi belirlenmesiyle mümkün olabilmektedir.

Bu makalede zamanlama mimarileri incelenecek ve hangi zamanlama mimarisi ne amaçla kullanılması gerektiği belirlenecektir. Ayrıca bu mimarilerin avantaj ve dezavantajları açıklanacaktır. Böylelikle Electronic Control Unit (ECU) dizaynından sorumlu yazılım mimarı hangi zamanlama mimarisinin mevcut durum için daha uygun olabileceğine ve kriterlere göre uygun mikrodenetleyici seçimine karar verebilecektir. Doğru mikrodenetleyici seçimi, hem zamandan hem de maliyetten kazanç sağlatmış olacaktır. Otomotiv endüstrisine bir yarar olarak, bu çalışma ECU birden fazla mikrodenetleyici içermeli mi, eğer içerirse bu mikrodenetleyicilerin hangi amaca göre seçilmesi gerektiği sorularına cevap olacaktır.

Çalışmanın sonuçlarına göre, bir sabit-öncelikli zamanlama algoritması olan Rate Monotonic (RM) katı gerçek-zamanlı sistemlerde emniyet sağlamak için iyi bir seçimdir. Earliest Deadline First (EDF) bir dinamik-öncelikli zamanlama algoritması olarak katı gerçek-zamanlı sistemlerde performans elde etmek için iyi bir yoldur. Mimari açıdan bakıldığında, olay tetikleyici esnektir ve yüksek kaynak kullanımı karakteristiğine sahiptir. Fakat katı gerçek-zamanlı sistemlerde önemli olan determinizm ve öngörülebilirlik açısından güvenilir değildir. Ayrıca, olay tetikleyicisinin çalışma süresindeki sistem yükü zaman tetikleyicisinden daha yüksektir. Zaman tetikleyici ise determinizm ve öngörülebilirlik sağlar, ancak daha düşük kaynak kullanımı oranı sağlar.

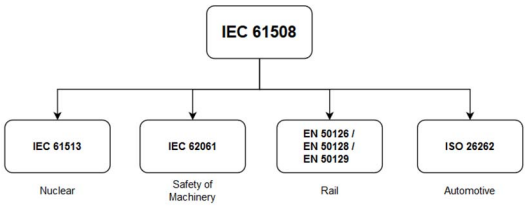
Güvenlik, otomotiv endüstrisinin çok önemli bir yönüdür. Güvenli bir sistem üretimi ve geliştirmesi için otomotiv şirketleri tarafından bazı standartlar oluşturulmuştur. Bu standartlardan bölüm 2'de bahsedilecektir. Araçlarda güvenli yolculuk yapılması için zaman önemli bir parametredir. Gerçek-zamanlı sistemler, tanımlanmış kısıtlı zamanda yanıt vermeyi garanti etmelerinden dolayı zaman parametresinin önemli olduğu alanlarda yaygın olarak kullanılır. Bu sistemler hakkında detaylı bilgi bölüm 3'de bulunacaktır. Sistemdeki görevleri

978-1-7281-8541-5/20/$31.00 ©2020 IEEE

yönetmek için her gerçek-zamanlı sistem Merkezi İşlem Birimi (CPU) içerir. CPU, görevleri belli bir sıraya göre çalıştırmak için kararlar almalıdır. Bu sebeple bazı algoritmalara ihtiyaç duyar. Bu algoritmalara zamanlama algoritmaları denir. Bölüm 4'de konu hakkında bilgi verilecektir. Gerçek-zamanlı sistemler, sistem yanıtı geciktiğinde ortaya çıkan tehlikelere göre sınıflandırılabilir. Bunlar esnek gerçek-zamanlı, sıkı gerçek-zamanlı ve katı gerçek-zamanlı sistemlerdir. Gerçek-zamanlı sistemler ihtiyaca göre özel işletim sistemleri kullanır. Katı gerçek-zamanlı sistemler için kullanılan mevcut işletim sistemleri bölüm 5'te ele alınmıştır. Bölüm 4'te açıklanan zamanlama algoritmaları, bölüm 5'de açıklanan katı gerçek-zamanlı işletim sistemi üzerinde uygulanmıştır. Uygulama adımları, bulgular ve değerlendirmeler bölüm 6'ya eklenmiştir. Bölüm 6'daki sonuçlar üzerinde çıkarımlar yapılıp sonuç bölümünde çıktılar ayrıca değerlendirilmiştir.

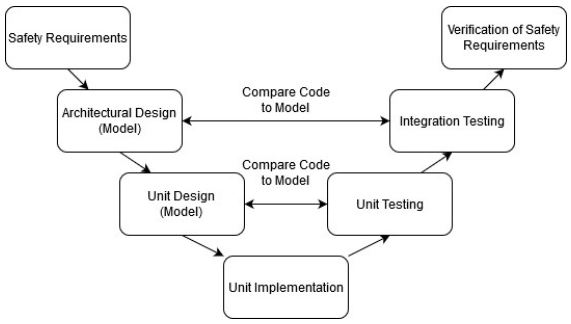
II. OTOMOTİV ENDÜSTRİSİNDE GÜVENLİK

Otomotiv endüstrisindeki büyük topluluklar, insanlara koruma sağlamak için bazı standartlar oluşturdu. Bunlardan biri Uluslararası Standardizasyon Örgütü (ISO) 26262 [1]. Aslında Uluslararası Elektroteknik Komisyonundan (IEC) 61508 [2] türetilmiştir. IEC 61508, her türlü endüstri için geçerli olan temel bir fonksiyonel güvenlik standardıdır. Ayrıca, ISO 26262, Otomotiv Elektrik / Elektronik (E / E) Sistemleri için IEC 61508'in bir uyarlamasıdır. ISO 26262, güvenlik kapsamında otomotiv E / E ürünlerinin tüm ürün geliştirme yaşam döngüsünde süreçleri kısıtlar.



Şekil 1: ISO26262 türetilmesi

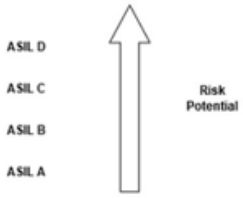
ISO 26262, yazılım geliştirmede ürün geliştirmeyi organize etmek için V modeli çerçevesini kullanmaktadır [1]. Model tabanlı yazılım tasarımıdır, çünkü model tabanlı tasarım ve ISO 26262 birbirini tamamlar. Bu model V-modeli olarak adlandırılır, çünkü fazlar birbirlerini V harfi şeklinde doğrularlar. Şekil 2'de gösterilmektedir.



Şekil 2: V-model

Bu çerçevenin bazı aşamaları vardır. Birinci aşama, güvenli olmayan veya etkisiz olarak donanım veya yazılım hatalarını gidermek için belirlenen birçok güvenlik gereksinimine sahiptir. Bir sonraki aşamada, bir önceki aşamada belirlenen güvenlik gereksinimlerini sağlayan üst mimari, her bileşen için tasarlanmıştır. Diğer aşama yazılım birimi alt sistemleri de benzer şekilde tasarlanmıştır. Bunlar sırasıyla uygulama ve birim test aşaması olarak adlandırılır. Birim testinden sonra, tüm sistem entegre birimlerin davranışından emin olmak için entegrasyon test adımından geçer. Son aşamada, güvenlik gereksinimlerini doğrulamak için sistem gerçek ortamla test edilir.

ISO 26262 gerekliliklerinin yapılandırılmasında, gelişimin derecesini belirlemek için Otomotiv Güvenlik Bütünlüğü Seviyeleri (ASIL'ler) kullanılır [2]. ASIL'ler zararın olasılığı ve kabul edilebilirliğine dayanmaktadır. Standart olarak A, B, C ve D olmak üzere dört seviye vardır. ASIL-A en düşük otomotiv tehlikesini temsil ederken ASIL-D en yüksek dereceyi temsil eder. Örneğin, arka lambalar gibi bileşenler ASIL-A sınıfı gerektirirken, hava yastıkları, kilitlenme önleyici fren sistemleri ASIL-D sınıfı gerektirir.



Şekil 3: ASIL

Hava yastıklarının arızası, arka lambaların arızasına göre daha yüksek bir hasar olasılığına sahiptir. Her elektronik bileşen için ASIL derecesi ciddiyet, maruziyet ve kontrol edilebilirlik değişkenlerine göre belirlenir. ASIL'in sınıflandırılması, araçlarda en yüksek güvenliği sağlamaya yardımcı olur.

Kıdemli programcılar kolayca hata yapmaktan kurtulabilirler. Ancak bu durum genç programcılar için geçerli değildir. Öte yandan, donanıma kolay erişim, düşük bellek gereksinimleri ve verimli çalışma zamanı performansı, gömülü sistemlerde C programlama dilinin popüler kullanımının nedenleridir. Bununla birlikte C, teknik olarak yasal olan basit hatalara eğilimli bir sözdizimi gibi oldukça sınırlı çalışma zamanı denetimi gibi bazı sorunlara sahiptir.

Bu nedenlerden ötürü, Motor Endüstrisi Yazılım Güvenilirlik Derneği (MISRA), güvenlik açısından kritik sistemlerde C programlama dilinin kullanımı için bir dizi yazılım geliştirme kılavuzu oluşturmuştur [3]. Buna MISRA C denir. Otomotiv endüstrisinde kullanılan gömülü sistemlerde kod güvenliği, taşınabilirliği ve emniyetini sağlamayı amaçlamaktadır.

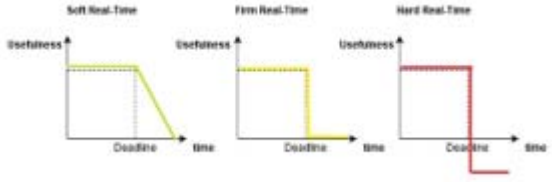
ISO 26262 uyumlu ECU zamanlamaları oluşturmak için görev zamanlama güvenliği doğrulanmalıdır. Zamanlama, araç sistemlerinde güvenilirlik ve güvenlik için kritik bir performans faktörüdür. Yeni teknoloji ile işlevsellik arttıkça zamanlama analizi gittikçe zorlaşmaktadır. Güvenlik gereksinimlerinden biri, kararlı ve öngörülebilir zamanlama davranışına ve gereken bilgi işlem gücü miktarına sahip görevlerdir.

III. GERÇEK-ZAMANLI SİSTEMLER

Bir sistem, bir veya daha fazla giriş kümesini ve girişlerle ilgili bir veya daha fazla çıkış kümesini içeren bir kara kutudur. Sözlükteki determinizmin anlamı, tüm olaylar

önceden var olan nedenlerle tamamen belirlenir [4]. Deterministik bir sistem aynı zamanda belirli bir girdiden her zaman aynı çıktıyı üreten bir sistemdir. Öngörülebilirlik otomotiv endüstrisi için önemli bir özelliktir. Bu nedenle, deterministik sistemler bu endüstride yaygın olarak kullanılmaktadır.

Zamanın güvenli araç sistemi sağlamak için çok önemli bir unsur olduğu önceki kısımlarda belirtildi. Zaman, sistemleri gerçek-zamanlı ve diğer tip sistemler olarak ayıran ana unsurdur. Gerçek-zamanlı sistemler, olaylara belirli bir zaman kısıtlaması içinde tepki vermesi gereken özel sistemlerdir [5]. Bir reaksiyon çok geç ortaya çıkarsa, çok tehlikeli sonuçlara yol açabilir. Sistemler, sebep oldukları tehlikelere göre üç farklı kategoride sınıflandırılmaktadır. Bunlar esnek gerçek zamanlı sistemler, sert gerçek-zamanlı sistemler ve katı gerçek-zamanlı sistemlerdir.



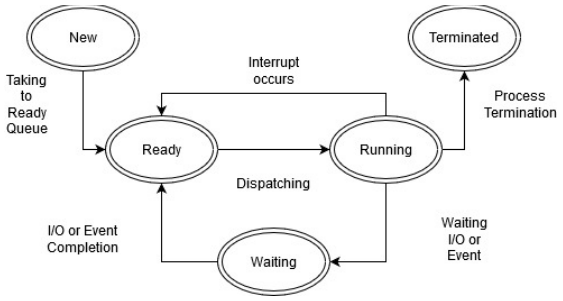
Şekil 4: Gerçek-zamanlı sistemlerin davranışı

Esnek gerçek-zamanlı sistem, sistem zaman sınırından sonra yanıt verdiğinde yalnızca performans düşüşüne neden olan bir sistemdir. Sert gerçek-zamanlı sistemi, esnek ve katı gerçek-zamanlı sisteme göre bir ara sistemdir. Bu sistemdeki seyrek zaman sınırı kaçırmaları tolere edilebilir. En katı sistem katı gerçek-zamanlı sistemdir. Zaman sınırından sonra yanıt verirse, bu sistemde yıkıcı sonuçlara neden olur.

Katı gerçek-zamanlı sistemler, ASIL-D sınıfı gereklilikleri sağlamak için otomotiv endüstrisinde kullanılmaktadır. Bu sistemlerin ASIL-A'dan ASIL-D'ye kadar derecesi olan birçok görevi vardır. Her bilgisayarda çalışan yazılımın içindeki komutları işleyen bir CPU bulunur. Bu CPU, güvenlik sınıfına göre bu görevleri yönetmelidir. Bu amaçla CPU, görev zamanlaması sağlamak için zamanlama algoritmasına sahip zamanlayıcı kullanır.

CPU, işlemler arasında geçiş yaparak bilgisayarı daha verimli hale getirir. Her zaman aralığında bir işlem yürütülmesi hedeflenmektedir. CPU, işlemlerde oluşacak bekleme durumlarında diğer işlemleri çalıştırır. Bellekte birçok süreç var. Bir işlem herhangi bir şekilde bekleme moduna geçtiğinde, CPU başka bir işleme geçer. İşlem yürütme, CPU yürütme ve Giriş / Çıkış (G / Ç) bekleme döngüsünü içerir. İşlemler bu iki durum arasında geçiş yapar. İşlemler CPU patlaması ile çalışmaya başlar ve I / O patlaması ile devam eder. Hesaplamalar için bir CPU patlaması gerçekleştirilir. Sistemler arasında veri aktarımını beklemek için bir G / Ç patlaması yapılır.

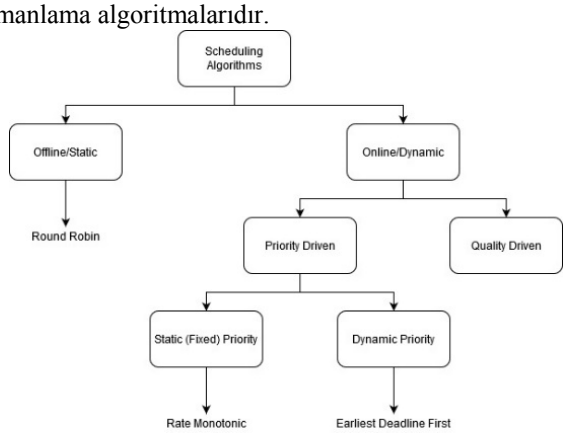
Her CPU Zamanlayıcı'nın görev yürütme durumları için bir durum makinesi vardır. Örnek durum makinesi Şekil 5'te gösterilmiştir.



Şekil 5: Görev durum makinası

Temel olarak, programlama algoritmaları iki ana başlık altında kategorize edilir. Bunlar çevrimdışı / statik ve çevrimiçi / dinamik zamanlamadır [6]. Çevrimdışı zamanlamada, görevlerin listesini ve etkinleştirme zamanlarını içeren bir zamanlama tablosu vardır. Çalışma zamanında, basit bir dağıtıcı tabloda gösterilen kararları yürütür. Round Robin (RR), çevrimdışı zamanlama için kullanılan bir zamanlama algoritmasıdır. Çevrimiçi zamanlamada, önceden tanımlanmış bir dizi kural vardır. Çalışma zamanında, görev dağıtıcı belirli bir görev kümesine uygulamak için bu önceden tanımlanmış kurallara dayalı bir karar alır.

Çevrimiçi planlama iki bölüme ayrılabilir; bunlar öncelik odaklı ve kalite odaklıdır [7]. Öncelik güdümlü, katı gerçek-zamanlı sistemlerde yaygın bir rol oynamaktadır. Sabit-öncelikli bir algoritma, her görevdeki tüm işlere aynı önceliği atar. Öte yandan, dinamik-öncelik algoritması her görevdeki bağımsız işlere farklı öncelikler atar. Rate Monotonic (RM) ve Earliest Deadline First (EDF), sırasıyla sabit-öncelik ve dinamik-öncelik zamanlaması için popüler zamanlama algoritmalarıdır.



Şekil 6: Zamanlama algoritmalarının sınıflandırılması

IV. ZAMANLAMA ALGORİTMALARI

A. Olay Tetiklemeli Zamanlama

1) Rate Monotonic

Rate Monotonic, basit bir kurala sahip olan sabit-öncelikli bir programlama algoritmasıdır [5]. Kural, görevlerinin önceliklerinin derleme zamanında belirlenmesidir. Öncelikleri zamanla değişmez. Öncelikleri, yürütme sıklıkları ile doğru orantılıdır. En kısa süreye sahip olan görev en yüksek önceliğe sahiptir. Aşırı yüklenmelerde deterministik davranış vardır. Böylece görevler öncelik seviyesinden etkilenir. Bu nedenle, RM statik programlamaya EDF'den daha yakındır. Statik tarafa yaklaştıkça performans azalmaktadır. RM için üst sınır kullanımını hesaplama ihtiyacı varsa, Lui ve Layland

kanıtlarından yararlanabiliriz [8]. Bir task çalıştırılma süresini gösteren Ci, çalıştırılma sıklığını gösteren Ti, ve task sayısını gösteren n ile tanımlanır. Lui ve Layland, Şekil 7'de gösterilen hesaplamayı verdiler.

$$U = \sum_{i=1}^{n} \frac{Ci}{Ti} < n(2^{1/n} - 1)$$

Şekil 7: CPU kullanım hesaplamasının formülü

Bu kanıta göre U, n için yaklaşık 0,69 olan ln (2) 'ye yakınlaşır. Yani RM, ancak toplam yük yüzde 69'dan fazla değilse uygulanabilir bir programı garanti edebilir [4].

2) Earliest Deadline First

EDF bir dinamik-öncelik zamanlama algoritmasıdır. Bu algoritmanın kuralı, mutlak süre kısıtlarına göre yürütme görevini seçmesidir. Öncelik değişikliği olduğunda hazır kuyruk azalan önceliklere göre sıralanır. Bu nedenle, bu algoritmanın yükü daha yüksektir. Çekirdek desteği gerektirdiği karmaşık bir uygulamaya sahiptir. Öte yandan iyi ölçeklendirilebilir ve aynı zamanda seyrek gerçekleşen görevlerde kolayca üstesinden getirebilir. Lui ve Layland'ın kanıtlarına göre [8], EDF tam işlemci kapasitesinden yararlanabilir, görev setlerini 1'e kadar kullanımla zamanlayabilir.

B. Zaman Tetiklemeli Zamanlama

Zaman tetiklemeli zamanlamada, görevler önceden tanımlanan noktalarda başlatılır. Çalışma zamanı dağıtımı bir dizi kurala göre gerçekleştirilir. Bu kurallar derleme zamanında hazırlanan çizelge tablosunda tanımlanır. Dağıtıcı çizelgeleme kararlarını bu tabloya göre alır. Bu nedenle, çalışma zamanında daha düşük sistem yükü vardır. Bu aynı zamanda esnekliğin olmaması gibi dezavantajı beraberinde getirir.

Bu zamanlama için, çalışma zamanından önce her şey bilinmelidir [9][10]. Bu da bir maliyet oluşturur. Ancak, geliştiriciler için önemli bir nitelik olan determinizmi sağlar. Her şeyi önceden planlayabilirler.

V. KATI GERÇEK-ZAMANLI SİSTEMLER

İşletim sistemi standartları, uygulamanın bir platformdan diğerine taşınabilirliğini sağlayabilir. Ayrıca, tek bir uygulama için birkaç çekirdek sağlayıcısına sahip olma olanağına izin verebilirler. Bu nedenle, bu standartlar çok önemlidir. Kullanıldıkları yere göre sınıflandırılabilirler. Gerçek-Zamanlı Taşınabilir İşletim Sistemi Arabirimi (RT-POSIX), ana genel amaçlı işletim sistemi standardının gerçek-zamanlı uzantısı olan popüler bir arayüzdür. APEX ise aviyonik sistemlerde yaygındır.

Offene Systeme und deren Schnittstellen für die Elektronik in Kraftfahrzeugen (OSEK), bir Alman otomotiv şirketi gömülü sistemler konsorsiyumu tarafından kurulmuş bir standarttır [11]. Bu standartta bir iletişim yığını, bir ağ yönetim protokolü ve diğer ilgili konular hakkında spesifikasyonlar üretilmiştir. ECU'lar için standart yazılım mimarisi olarak tasarlanmıştır. Fransız otomobil imalatçıları, Vehicle Distributed eXecutive (VDX) adında benzer bir projeye sahipti. VDX, konsorsiyuma katıldıktan sonra resmi isim OSEK/VDX oldu.

Birçok İşletim Sistemi (OS) vardır, ancak otomotiv açısından bakıldığında bu işletim sistemlerinin bazı dezavantajları vardır. Oldukça hantal ve gerçek-zamanlı performansları yeterince gerçek değildir. Uygulama Programlama Arabirimleri (API), eşzamanlı ve eşzamansız görev zamanlamayı birlikte kullanmak için uygun değildir. OSEK API, otomotiv endüstrisi için özel olarak tasarlanmıştır. Bu nedenlerden ötürü, OSEK OS otomotiv gömülü sistemlerinde kullanılmaktadır.

Bunlara ek olarak OSEK / VDX'in birçok avantajı vardır. Uygulamadan bağımsız bir mimariye sahip olduğu için uygulama yazılımının taşınabilirliğini ve tekrar kullanılabilirliğini destekler. Arayüzlerin özellikleri donanım ve ağdan bağımsızdır. Bu avantaj, bir uygulamanın farklı ECU'larda kullanılmasını sağlar. Mimarisi verimli bir şekilde tasarlandığından, mevcut işlevler yapılandırılabilir ve ölçeklenebilir. Ayrıca sisteme kolayca yeni işlevler eklenebilir.

Standartlara uygunluk, standardın şartlarını desteklediği anlamına gelir. Erika Enterprise açık kaynaklı bir OSEK / VDX Katı Gerçek-Zamanlı İşletim Sistemidir (RTOS) [12]. AUTomotive Açık Sistem Mimarisi (AUTOSAR) API'sinin bir alt kümesinden esinlenen bir API uygular. Erika 2012 yılında sertifika aldı ve daha sonra Erika'ya AUTOSAR OS spesifikasyonları uygulandı. OSEK / VDX uyumlu sertifikalı ilk açık kaynaklı Ücretsiz RTOS'tur. Olay tetikleyici mimariyi ve zaman tetikleyici mimariyi destekler. Bu işletim sistemi 1-4 Kb Flash ayak izi ve 8 ila 32 bit mikrodenetleyiciler için uygundur. Ayrıca, ayak izini sınırlamak için kullanılan uygunluk sınıflarına sahiptir. En iyi avantajlarından biri, bir OSEK Uygulama Dili (OIL) dosyası veya AUTOSAR Genişletilebilir İşaretleme Dili (XML) ile statik olarak yapılandırılabilir.

VI. UYGULAMA VE DEĞERLENDİRME

Tek işlemcili sistem, programlama algoritmalarını değerlendirmek için bir seçimdir. Görevler arasında bağlam geçişi, kaçırılmış zaman sınırı ve her görev için CPU kullanımı açıkça görülebilir. Böylece, Arduino Uno uygulamak için seçilebilir. Erika OS'nin bu kart için desteğinin bulunması da iyi bir avantajdır.

Arduino Uno, Atmega328 işlemci kullanan basit bir karttır [13]. 14 dijital G / Ç çıkış pini vardır, bunların 6'sı Darbe Genişliği Modülasyonu (PWM) çıkışı olarak kullanılabilir. 16 MHz kristal osilatör, Evrensel Seri Veri Yolu (USB) bağlantısı, 2.1mm güç girişi, Devre İçi Seri Programlama (ICSP) başlığı ve sıfırlama düğmesi vardır. Çalıştırmak için DC 7 ~ 12V güç kaynağına bağlanmak yeterlidir.

Erika OS resmi web sitesinden indirildi. Bu işletim sistemi, Erika'nın geliştiricileri tarafından Erika resmi web sitesinde yayınlanan talimat kılavuzuna göre Arduino Uno kartına inşa edildi ve gömüldü. Daha sonra Işık Yayan Diyot (LED) yanıp sönmesi gibi bazı örneklerle test edildi. Programlama algoritmalarının artılarını ve eksilerini göstermek için bir senaryo belirlenmiştir. Bu senaryoda sabit öncelikleri olan 3 görev vardır. Spesifikasyonları Tablo 1'de gösterilmiştir.

Tablo 1: Senaryoda kullanılan parametreler

Task	Execution Time (s)	Period (s)
Task1	1	4
Task2	2	6
Task3	3	8

Bu görevlerin CPU Kullanımı Şekil 7'de belirtilen formüle göre hesaplanır. Hesaplama Şekil 8'de gösterilmiştir.

$$U = \frac{1}{4} + \frac{2}{6} + \frac{3}{8} = 0.96$$

Şekil 8: Senaryoya göre CPU kullanım hesaplaması

İşletim sistemini yapılandırmak için Bölüm 5'te belirtilen bir OIL dosyasına sahiptir. Bu dosya istenen ve gerekli konfigürasyona göre yazılmıştır. Her konfigürasyon algoritması için bazı konfigürasyonlar kolayca değiştirilebilir. RM ilk analiz zamanlama algoritmasıydı. Sabit Öncelikli (FP) bir algoritmadır. OIL dosyası Şekil 9'daki gibi yapılandırılmıştır.

```
    KERNEL_TYPE = FP;

};

TASK TaskL1 {
    PRIORITY = 3;
    STACK = SHARED;
    SCHEDULE = FULL;
    AUTOSTART = TRUE;
};

TASK TaskL2 {
    PRIORITY = 2;
    STACK = SHARED;
    SCHEDULE = FULL;
    AUTOSTART = TRUE;
};

TASK TaskL3 {
    PRIORITY = 1;
    STACK = SHARED;
    SCHEDULE = FULL;
    AUTOSTART = TRUE;
};
```

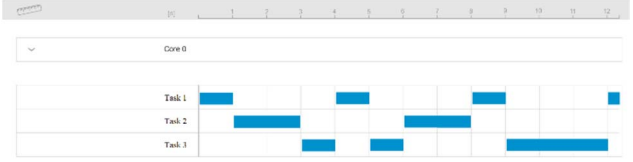
Şekil 9: RM için OIL dosyasının ayarlanması

Bu görevlerin içeriği, araçlardaki ECU uygulamaları gibi uygulama sürelerine göre ayarlandı. Görevlerin yürütme sürelerini göstermek için görevlere zaman damgalı Evrensel Asenkron Alıcı Verici (UART) günlükleri eklendi. RM için UART günlükleri Şekil 10'da bulunur.

```
14:32:13.313 -> Cpu Utilization: 0.96
14:32:13.313 -> HyperPeriod: 24 For 3 Task
14:32:14.319 -> TASK1_started
14:32:15.311 -> Task1_finished
14:32:15.311 -> Task2_started
14:32:16.321 -> Task2_continue
14:32:17.318 -> Task2_finished
14:32:17.318 -> Task3_started
14:32:18.339 -> Task3_continue
14:32:18.542 -> TASK1_started
14:32:19.524 -> Task1_finished
14:32:20.310 -> Task3_continue
14:32:20.644 -> Task2_started
14:32:21.632 -> Task2_continue
14:32:22.644 -> Task2_finished
14:32:22.745 -> TASK1_started
14:32:23.763 -> Task1_finished
14:32:24.293 -> Task3_finished
14:32:24.293 -> Task3_started
14:32:25.291 -> Task3_continue
14:32:26.301 -> Task3_continue
14:32:26.963 -> TASK1_started
14:32:27.971 -> Task1_finished
14:32:27.971 -> Task2_started
14:32:28.954 -> Task2_continue
14:32:29.983 -> Task2_finished
14:32:30.256 -> Task3_finished
14:32:31.167 -> TASK1_started
14:32:32.191 -> Task1_finished
14:32:32.191 -> Task3 started
```

Şekil 10: RM UART çıktısı

Bu senaryonun sonuçlanan davranışı, RM zamanlama algoritmasının artılarını ve eksilerini göstermek için görselleştirildi. İlgili görsel Şekil 11'de verilmiştir.



Şekil 11: RM çıktısının görselleştirilmesi

Şekil 11'e bakıldığında, Görev 3 zaman kısıtını kaçırıyor. Görev 3 en düşük önceliğe sahiptir ve yürütme süresi 3 saniyedir. 8'inci saniyede tamamlanmalıydı. Bu, RM için bir dezavantajdır, ancak en düşük öncelikli görev için zaman kısıtı kaçırılmıştır. Bu görev ASIL-A, araçiçi ışığının kapı durumunu göstermesi görevi gibi düşünülebilir. Bu görevin gecikmesi, bu sistemde yıkıcı sonuçlara neden olmaz. Bu zaman kısıtının kaçırılması, CPU kullanımını azaltarak ortadan kaldırılabilir. Bu analiz, bir FP zamanlama algoritması olan RM'nin performansı reddederek güvenli bir sistem oluşturmak için kullanılabileceğini gösteriyor.

Bir sonraki zamanlama algoritması, Dinamik-Öncelik Zamanlama Algoritması olan EDF'dir. FP'den daha az çaba gerektirir, çünkü görevleri zaman kısıtlarına göre otomatik olarak zamanlar. OIL dosyası EDF için yeniden düzenlenmiştir. KERNEL_TYPE değişkeni EDF olarak ayarlandı. Bu, Şekil 12'de gösterilmiştir.

```
    KERNEL_TYPE = EDF;

};

TASK TaskL1 {
    PRIORITY = 3;
    STACK = SHARED;
    SCHEDULE = FULL;
    AUTOSTART = TRUE;
};

TASK TaskL2 {
    PRIORITY = 2;
    STACK = SHARED;
    SCHEDULE = FULL;
    AUTOSTART = TRUE;
};

TASK TaskL3 {
    PRIORITY = 1;
    STACK = SHARED;
    SCHEDULE = FULL;
    AUTOSTART = TRUE;
};
```

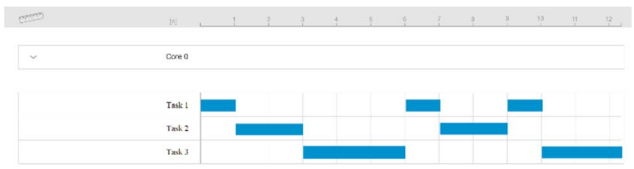
Şekil 12: EDF için OIL dosyasının ayarlanması

Bu kez, EDF için aynı senaryo uygulanmaktadır. EDF için UART günlükleri Şekil 13'te bulunur.

```
14:35:19.063 -> Cpu Utilization: 0.96
14:35:19.063 -> HyperPeriod: 24 For 3 Task
14:35:19.063 -> TASK1_started
14:35:20.060 -> TASK1_finished
14:35:20.060 -> TASK2_started
14:35:21.056 -> TASK2_continue
14:35:22.086 -> TASK2_finished
14:35:22.086 -> TASK3_started
14:35:23.082 -> TASK3_continue
14:35:24.078 -> TASK3_continue
14:35:25.073 -> TASK3_finished
14:35:25.073 -> TASK1_started
14:35:26.103 -> TASK1_finished
14:35:26.103 -> TASK2_started
14:35:27.098 -> TASK2_continue
14:35:28.095 -> TASK2_finished
14:35:28.095 -> TASK1_started
14:35:29.091 -> TASK1_finished
14:35:29.091 -> TASK3_started
14:35:30.121 -> TASK3_continue
14:35:31.117 -> TASK3_continue
14:35:32.114 -> TASK3_finished
14:35:32.114 -> TASK1_started
14:35:33.122 -> TASK1_finished
14:35:33.122 -> TASK2_started
14:35:34.151 -> TASK2_continue
14:35:35.129 -> TASK2_finished
14:35:35.129 -> TASK1_started
14:35:36.126 -> TASK1_finished
14:35:36.126 -> TASK3_started
14:35:37.155 -> TASK3_continue
14:35:38.150 -> TASK3_continue
14:35:39.147 -> TASK3 finished
```

Şekil 13: EDF UART çıktısı

Bu senaryonun sonuçlanan davranışı EDF zamanlama algoritmasının artılarını ve eksilerini göstermek için görselleştirildi. Şekil 14'te verilmiştir.



Şekil 14: EDF çıktısının görselleştirilmesi

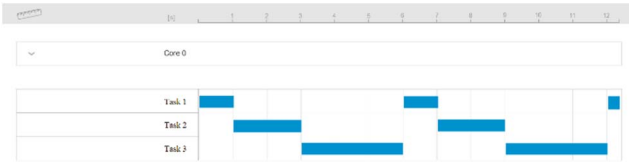
Şekil 14'e bakıldığında, kaçırılmış bir zaman kısıtı yoktur. Bu bize EDF'nin FP'den daha yüksek CPU kullanımı sağlayabildiğini gösteriyor. Ancak bu EDF'nin dezavantajları olmadığını göstermez. Zaman kısıtı zamanlayıcı, her görev için zaman kısıtı içerisinde yerine getirilmesini garanti eder, ancak herhangi bir görev için minimum yanıt süresi sağlamak mümkün değildir. En yüksek öncelikli görev her zaman RM cinsinden minimum yanıt süresine sahiptir, ancak EDF için garanti vermek mümkün değildir. Yeni bir görev veya yanlış En Kötü Durum Yürütme Süresi (WCET) tahmini nedeniyle sistem aşırı yüklenirse, domino etkisi oluşabilir. Domino etkisi, bir görev beyan edilen çalışma süresinden daha fazlasını yürütmek için zaman kısıtını kaçırdıktan sonra diğer tüm görevlerin zaman kısıtlarını kaçırmasıdır. Bu, çarpışma anında hava yastıklarını açan bir görev gibi ASIL-D görevinin zaman kısıtını da kaçırması olasılığıdır. Bu durum, bu sistemde yıkıcı sonuçlara neden olur. Bu nedenle EDF, yüksek performans ve düşük güvenlik gerektiren görevler için kullanılabilir.

Bunlar çevrimiçi zamanlama algoritmalarının artıları ve eksileriydi. Otomotiv endüstrisinin performans ve güvenlik olarak sadece iki parametresi yoktur. Determinizm ve öngörülebilirlik, otomotiv endüstrisinde çalışan geliştiriciler için diğer önemli parametrelerdir. Bu nedenle, zaman tetikleyici planlama mimarisi bu sektörde önemlidir [10]. Bu mimaride çevrimdışı tanımlanmış görevleri ve bunların tetiklenme zamanlarını içeren zamanlama tablosu bulunur. Bu zamanlama tablosu OIL dosyasında hazırlanmıştır ve her görev belirli bir zamanda çağrılmaktadır. Zaman Tetikleyicisi (TT) için UART günlükleri Şekil 15'te bulunur.

```
15:09:22.443 -> Cpu Utilization: 0.96
15:09:22.443 -> HyperPeriod: 24 For 3 Task
15:09:22.941 -> TASK1_started
15:09:23.969 -> TASK1_finished
15:09:23.969 -> TASK2_started
15:09:24.966 -> TASK2_continue
15:09:25.962 -> TASK2_finished
15:09:25.962 -> TASK3_started
15:09:26.958 -> TASK3_continue
15:09:27.988 -> TASK3_continue
15:09:28.985 -> TASK3_finished
15:09:28.985 -> TASK1_started
15:09:29.982 -> TASK1_finished
15:09:29.982 -> TASK2_started
15:09:30.977 -> TASK2_continue
15:09:32.007 -> TASK2_finished
15:09:32.007 -> TASK3_started
15:09:33.003 -> TASK3_continue
15:09:33.999 -> TASK3_continue
15:09:34.995 -> TASK3_finished
15:09:34.995 -> TASK1_started
15:09:36.024 -> TASK1_finished
15:09:36.024 -> TASK2_started
15:09:37.020 -> TASK2_continue
15:09:38.016 -> TASK2_finished
15:09:38.016 -> TASK3_started
15:09:39.011 -> TASK3_continue
15:09:40.040 -> TASK3_continue
15:09:41.023 -> TASK3_finished
```

Şekil 15: TT UART çıktısı

Bu senaryonun sonuçlanan davranışı TT zamanlama algoritmasının artılarını ve eksilerini göstermek için görselleştirildi. Bu görsel Şekil 16'da verilmiştir.



Şekil 16: TT çıktısının görselleştirilmesi

Olayla tetiklenen zamanlama mimarisi ile zamanla tetiklenen zamanlama mimarisi arasında bir karşılaştırma yapmak gerekirse, olayla tetiklenen esnektir ve yüksek kaynak kullanımı ile karakterize edilir. Ancak, gerçek-zamanlı katı sistemlerde önemli olan determinizm ve öngörülebilirlik açısından güvenilir değildir. Ayrıca, tetiklenen olay, tetiklenen süreden daha yüksek çalışma zamanı yüküne sahiptir. Tetiklenen zamanın periyodik bir dünyası vardır. Sporadik görevler için esnek değildir. Determinizm ve öngörülebilirlik sağlar, ancak kaynak kullanımı oranı daha düşüktür.

VII. SONUÇ

Bu makalede, programlama algoritmaları üç kategoride sınıflandırılmıştır. Bunlar sabit-öncelikli zamanlama, dinamik-öncelikli zamanlama ve zaman tetikleyici zamanlamadır. Bu programlama algoritmaları otomotiv endüstrisinde önemli olan parametrelere göre analiz edilmiştir. Analiz sonucu, her zamanlama algoritmasının avantajları ve dezavantajları vardır. Gereksinimlere göre seçilmelidir.

Çıkarımlardan bir diğeri, iki farklı zamanlama algoritması için bir kartta iki CPU kullanılabilir. Biri performans için, diğeri güvenlik için seçilebilir. Görevler ASIL düzeylerine göre ilgili CPU'lara dağıtılabilir. Başka bir çıkarım ise birden fazla zaman tablosu tanımlanması ve bu zaman tabloları arasında bir karar veren yapay zekanın geliştirilmesidir.Bu yapay zeka çalışma esnasında sporadik görevlerin üstesinden gelmek için gerekli kararları vermesidir.

Gelecekteki çalışmalar için bu analiz çok çekirdekli sistemler için yapılabilir. Her bir çekirdek farklı ASIL seviyelerini ilerletmek için ayrılabilir.

REFERENCES

[1] ISO: 26262 – “Road vehicles-Function safety Part 6: Product development at the software level” (2018).

[2] Redmill (1998). "IEC 61508: Principles and Use in the Managementof Safety". Computing & Control Engineering Journal, 9, 5, 1998.IEE, London.

[3] Jones, Nigel. "Introduction to MISRA C" Embedded Systems Programming, July 2002.

[4] Phillip Laplante, Real-Time Systems Design and Analysis - An Engineer's Handbook, IEEE Press,1993.

[5] Buttazzo, G. C.: HARD REAL-TIME COMPUTING SYSTEMS: Predictable Scheduling Algorithms and Applications (3rd ed.), Kluwer Academic Publishers,Boston, 2011.

[6] Weirong Wang, Aloysius K. Mok, Gerhard Fohler: Pre-Scheduling: Integrating Offline and Online Scheduling Techniques. EMSOFT 2003: 356-372.

[7] D. Rajesh, "Real-time scheduler design for safety-critical systems : A supervisory control approach", 2018.

[8] C.Liu, J.Layland, “Scheduling Algorithms for Multiprogramming in a Hard Real-time Environment”, JACM 20(1), 1973.

[9] Hermann Kopetz. Real-Time Systems - Design Principles for Distributed Embedded Applications. Springer, 2011.

[10] H. Kopetz, K. Ki, “Event triggered versus timed triggered real-time systems” Technical report 8/91, Insitut fur Technische Informatik TU Vienna, Austria, 1991.

[11] Joseph Lemieux, "The OSEK/VDX Standard: Operating System and Communication”, March 2000, p. 90.

[12] Erika-enterprise.com. 2020. [online] Available at: <http://www.erika-enterprise.com/index.php/community.html/> [Accessed 11 Apr 2020].

[13] Arduino.cc. 2020. [online] Available at: <https://www.arduino.cc/> [Accessed 19 May 2020].