Makale Gönderilme Tarihi: 22-12-2021
Makale Kabul Tarihi: 13-12-2021

Otomotiv Endüstrisinde Zamanlama Mimarilerinin Değerlendirilmesi

Evaluation of Scheduling Architectures in Automotive Industry

Berkay Saydam[1,2], Tolga Ayav[2]

[1]Ar-Ge Merkezi
TTTech Auto Turkey
berkay.saydam@tttech-auto.com

[2]Bilgisayar Mühendisliği Bölümü
İzmir Yüksek Teknoloji Enstitüsü
tolgaayav@iyte.edu.tr
berkaysaydam@iyte.edu.tr

Özet

Teknolojik gelişmeler araçlara yansırken güvenlikten ödün vermeden araçlara yeni işlevler ekleme zorluğunu da beraberinde getirmektedir. Araçlarda çeşitli işlevleri yerine getiren görevler farklı karakteristiklere sahiptir. Güvenlik ve performans, bu görevlerin karakteristiğini belirlemek için kullanılan iki temel kriterdir. Görevlerin karakteristikleri, Otomotiv Güvenlik Bütünlük Seviyeleri olarak bilinen güvenlik seviyelerine göre sınıflandırılabilir. Donanım ve yazılım tasarımıyla birlikte bu tasarımların doğrulanması ve testi otomotiv endüstrisinde uzun soluklu bir süreçtir. Bir Elektronik Kontrol Birimi sahada kullanılmaya başlandıktan sonra donanım tasarımındaki herhangi bir değişiklik oldukça maliyetlidir. Bu makalenin hipotezine göre, uygun görev çalıştırma sıralarını belirlemek için kullanılan zamanlama algoritmaları Merkezi İşlem Birimi tarafından özenle seçilmelidir. Ayrıca donanım ve yazılım tasarımında bu karakteristik özellikler ve algoritmalar da dikkate alınmalıdır. Aksi takdirde görevler, bir kritik bileşen için zaman kısıtının kaçırılması gibi güvenlik açısından önemli sorunlara neden olabilmektedir. Bu makalede, zamanlama mimarileri değerlendirilerek hangi zamanlama mimarilerinin hangi amaçla kullanılması gerektiği belirtilmektedir. Algoritmaların avantaj ve dezavantajları sunulmaktadır.
Anahtar kelimeler: zamanlama algoritmaları, katı gerçek zamanlı sistemler, OSEK/VDX sertifikalı işletim sistemi, statik zamanlama analizi, araçlarda güvenlik

Abstract

Technological advancements are reflected to the vehicles as well, but it brings the challenge of adding new functionalities to vehicles without compromising safety. Tasks are used to provide functionalities which are used in car. These tasks have different characteristics. Safety and performance are two main criteria to determine the characteristics of tasks. These characteristics can be classified according to their safety levels which are known as Automotive Safety Integrity Levels. Design of hardware and software together with their verification and testing is a long progress in automotive industry. Any changes on either hardware or software may be quite costly when an ECU began to be used in field. According to the hypothesis of this article, scheduling algorithms run by Central Processing Unit to determine the proper sequence of task executions, should be well known. Besides, the design of hardware and software should be done according to the characteristics and algorithms. Otherwise, tasks may cause serious problems like missing deadlines for safety-critical component. In this article, the scheduling architectures are evaluated distinguishing the purposes of them. The advantages and disadvantages of the algorithms are also explained.
Keywords: scheduling algorithms, hard real-time systems, OSEK/VDX certified operating system, static scheduling analysis, safety in vehicle

1. Giriş

Gelişen teknoloji farklı sektörlerde birçok ihtiyacı beraberinde getirdi. Bu gelişmeler aynı zamanda akıllı araçların ortaya çıkmasında önemli rol oynamaktadırlar. Güvenlikten ödün vermeden araçlara yeni işlevsellik kazandırmak için araç performansının iyileştirilmesi önem arz etmektedir. Araçlarda elektrikli sistemleri kontrol eden gömülü sistemlere Elektronik Kontrol Birimi (EKB) adı verilmektedir. Araçlara görevler aracılığıyla işlevsellik sağlayan mikrodenetleyiciler EKB tasarımının ilk adımında belirlenmektedir. Bu karar proje geliştirme sürecinin sonraki adımlarını önemli derecede etkilemektedir. Donanım, yazılım tasarımı ve test süreci otomotiv endüstrisinde oldukça uzun bir süreçtir. EKB'ler sahada kullanılmaya başlandıktan sonra donanımda yapılacak değişiklikler oldukça maliyetlidir. Bu nedenlerden dolayı mikrodenetleyici seçimi müşteriye sağlanacak işlevsellikleri ve maliyeti etkilemektedir.

Otomotiv sistemlerinde görevler araçlarda kullanılan işlevleri sağlamak için kullanılmaktadır. Bu görevler farklı karakteristiklere sahiptir. Güvenlik ve performans görevlerin karakteristiklerinin belirlenmesinde iki ana kriter olmaktadır. Karakteristikleri, Otomotiv Güvenliği Bütünlük Seviyeleri (ASIL) olarak bilinen güvenlik seviyelerine göre sınıflandırılabilir. ASIL-A seviye görevler performans sağlamak için birden fazla prosese ihtiyaç duyarken ASIL-D seviye görevler katı yanıt süresine ihtiyaç duyarlar. Makalede savunulan hipoteze göre, Merkezi İşlem Birimi (MİB) tarafından görev yürütme sıralarını belirlemek için kullanılan

zamanlama algoritmaları iyi seçilmelidir. Ayrıca donanım ve yazılım tasarımı bu görevlerin karakteristiklerine ve algoritmalara göre yapılmalıdır. Aksi durumda güvenlik-kritik görevlerin zaman sınırını kaçırması gibi ciddi sorunlara neden olacaktır. Birden fazla donanım ve yazılım tasarım süreci geçireceği için ek maliyete de sebep olacaktır.

Bu makalede zamanlama mimarileri değerlendirilmiş ve hangi zamanlama mimarisinin hangi amaçla kullanılması gerektiği belirlenmiştir. Ayrıca avantaj ve dezavantajları açıklanmıştır. Böylelikle EKB tasarımından sorumlu bir yazılım mimarı, amacına yönelik olarak hangi zamanlama mimarisini kullanması gerektiğini bilecek ve doğru mikrodenetleyici seçim kararını sonuç kısmında bulunan kriter ve tasarım fikirlerine göre verebilecektir. Otomotiv endüstrisine katkı olarak bu çalışma bir EKB'nin hangi durumda birden fazla mikrodenetleyici içereceği, ve bu mikrodenetleyicilerin hangi amaçlara göre seçilmesi gerektiği konularına da ışık tutmaktadır.

Analiz aşamasında zamanlama algoritmalarının avantaj ve dezavantajlarını göstermek için gerekli ortam kurulmuştur. Offene Systeme und deren Schnittstellen für die Elektronik in Kraftfahrzeugen/Vehicle Distributed eXecutive (OSEK/VDX) sertifikalı ve aynı zamanda ücretsiz olan Erika işletim sistemi seçilmiştir. Erika işletim sistemi derlenmiş ve Arduino Uno devre kartına gömülmüştür. Sabit Öncelikli Zamanlama Algoritması olan Oransal Monoton (OM) analiz edilecek ilk zamanlama algoritması olarak seçilmiştir. Bu senaryodan ortaya çıkan davranış görselleştirilmiştir. Zaman sınırını aşan bir göreve rastlanmıştır fakat bu görev düşük önceliğe sahiptir. Bu, OM için bir dezavantaj olsa da süre aşımı düşük öncelikli bir görevde gerçekleşmiştir. Bu görev araç kapısı açık olduğunda araç içerisinde yanan ışığın işlevselliğini sağlayan bir ASIL-A görevi olabilir. Bu görevdeki gecikme sistemde katastrofik sonuçlara sebep olmayacaktır. MİB kullanımı azaltılarak bu süre aşımı elimine edilebilir. Bu analiz sabit öncelikli zamanlama algoritması olan OM'nin performanstan feragat ederek güvenli bir sistem oluşturmak için kullanılabileceğini göstermektedir.

Bir sonraki algoritmaysa Dinamik Öncelikli Zamanlama Algoritması olan En Yakın Zaman Sınırı Önce (EYZSÖ) olmuştur. Görevleri zaman sınırlarına göre otomatik yürütmesinden dolayı sabit öncelikli algoritmalara göre daha az efor ihtiyacı vardır. Aynı senaryo EYZSÖ için yürütülmüştür. Herhangi bir zaman sınır aşımına rastlanmamıştır. Bu bize EYZSÖ'nin sabit öncelik algoritmalarından daha fazla MİB kullanımı sağladığını göstermektedir. Fakat bu EYZSÖ'nin dezavantaja sahip olmadığını göstermez. EYZSÖ her görev için zaman sınırını yerine getirmeyi garanti etsede verilen herhangi bir görev için minimum yanıt zamanını sağlamak mümkün değildir. OM'de en yüksek öncelikli görev her zaman minimum yanıt zamanına sahiptir, fakat bunu EYZSÖ'de garanti etmek mümkün değildir. Yeni bir görev eklenmesinden dolayı ya da yanlış bir En Kötü Durum Yürütme Süresi (EKDYS) tahmininden dolayı sistem aşırı yüklenirse domino etkisi oluşabilir. Domino etkisi, bir görev bildirilen çalışma süresinden daha fazlasını yürütmekle son tarihini kaçırdıktan sonra diğer tüm görevlerin son tarihlerini kaçırmasıdır. Bu da kaza anında airbagleri açan ASIL-D seviyesinde bir görevin zaman sınırını kaçırmasını mümkün kılabilir. Bu görevdeki gecikme sistemde katastrofik sonuçlara sebep olacaktır. Bu nedenle EYZSÖ yüksek performans ve düşük güvenlik gerektiren görevlerde kullanılabilir.

Otomotiv endüstrisi sadece performans ve güvenlik parametrelerine sahip değildir. Deterministik ve öngörülebilirlik de otomotiv sektöründeki geliştiriciler için önemlidir. Bu nedenle Zaman Tetiklemeli (ZT) zamanlama mimarisi önemli bir yere sahiptir. OSEK Uygulama Dili (OUD) dosyası işletim sistemlerinin temel parametrelerini belirlemek için sıkça kullanılır. Bu mimari çevrimdışı tanımlanmış görevler ve bu görevlerin tetiklenme zamanlarını içeren zamanlama tablosuna sahiptir. Bu zamanlama tablosu OUD dosyasında hazırlandı ve her görev verilen zamanlarda tetiklendi. Analizler sonucunda olay tetiklemeli ve zaman tetiklemeli mimarileri karşılaştırmak gerekirse olay tetiklemeli zamanlama mimarisi esnek ve yüksek kaynak kullanıma izin vermektedir. Fakat katı gerçek-zamanlı sistemlerde önemlilik arz eden deterministik ve öngörülebilirlik açısından güvenilir değildir. Ayrıca, olay tetiklemeli mimari daha yüksek çalışır-zaman yüküne sahiptir. Zaman tetiklemeli mimari periyodik bir dünyaya sahiptir. Düzensiz görevler için esnek değildir. Deterministik ve öngörülebilirliği düşük kaynak kullanımı sağlar.

Ulusal Yazılım Mühendisliği Sempozyum'unda yayınlanan bildirinin [1] aksine özet ve giriş kısmı makalenin değindiği problemi, amacı ve yazılma motivasyonunu daha anlaşılır bir şekilde anlatmak amacıyla tekrar yazılmıştır. Otomotiv endüstrisinde yapılan güvenlik çalışmaları standartlar dahilinde detaylandırılmıştır. Daha anlaşılır hale getirmek amacıyla gerçek-zamanlı sistemlerin ve algoritmaların kıyaslanması tablolar halinde gösterilip anlatımları detaylandırılmıştır. Zamanlama algoritmalarının kullanım kriterleri verilmiştir. Oransal monoton algoritmasının geçerlilik hesabı yapılıp adım adım gösterilmiştir. Önceki yayınlanan bildirinin senaryosu dışında farklı MİB kullanım oranları ve farklı task sayıları ile analizler yapılmış tablo halinde çıktılar verilmiş ve sonuçları yorumlanmıştır.

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'te bulunacaktır. Sistemdeki görevleri yönetmek için her gerçek zamanlı sistem Merkezi İşlem Birimi içerir. MİB, görevleri belli bir sıraya göre çalıştırmak için kararlar almalıdır. Bu kararları almak adına bazı algoritmalara ihtiyaç duyar. Bu algoritmalara zamanlama algoritmaları denir. Bölüm 4'te 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'te 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.

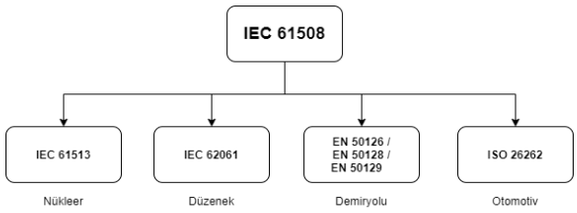
2. Otomotiv Endüstrisinde Güvenlik

Bu bölümde, otomotiv endüstrisinde emniyetin önemi vurgulanmakta ve emniyetle ilgili mevcut standartlar sunulmakta ve gözden geçirilmektedir. İlk olarak ISO 26262 standardının IEC 61508'den türetilmesi ve sonrasında da ISO26262 standardında tanımlı olan V-model ve ASIL seviyeleri hakkında detaylı bilgi verilecektir. Son olarak güvenli yazılım geliştirmek için yararlanılan MISRA'ya değinilecektir.

Güvenliğin sözlük anlamı tehlike, risk ya da yaralanma durumlarından korunma durumudur. İnsanlara zarar vermeyen bir sistem güvenli bir sistem olarak nitelendirilebilmektedir. Tamamen güvenli bir sistem yoktur, bu nedenle güvenli sistemler potansiyel riski kabul edilebilir bir düzeye indirmeye çalışır.

2.1. Güvenlik Standartları

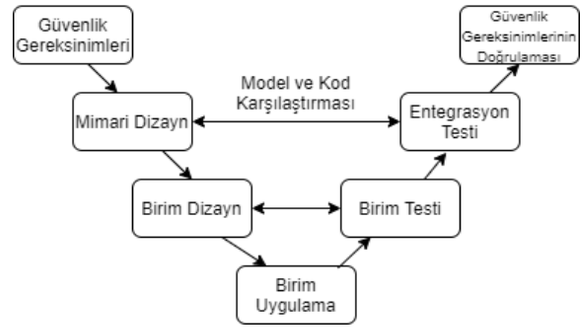
Otomotiv endüstrisinde güvenlik standartlarını belirleyen büyük topluluklar bulunmaktadır. Bunlardan biri Uluslararası Standardizasyon Örgütü'dür (ISO) 26262 [2]. Aslında Uluslararası Elektroteknik Komisyonundan (IEC) 61508 [3] türetilmiştir. IEC 61508, her türlü endüstri için geçerli olan temel bir 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.

2.2. V-model

ISO 26262, yazılım geliştirmede V modeli üzerinden tanımlamaktadır [2]. 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. Bu model Şekil 2'de gösterilmektedir.

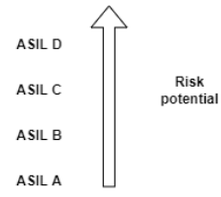


Şekil 2: V-model.

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 aşamada yazılım birimi alt sistemleri de benzer şekilde tasarlanmaktadır Tasarım aşamaları paralelde test de edilmektedirler. Birim testinden sonra, tüm sistem bütünleşmiş birimlerin davranışından emin olmak için entegrasyon test adımından geçer. Son aşamada, güvenlik gereksinimlerini doğrulamak için sistem gerçek ortamda test edilir.

2.3. Otomotiv Güvenlik Bütünlük Seviyeleri

ISO 26262 gerekliliklerinin yapılandırılmasında, gelişimin derecesini belirlemek için Otomotiv Güvenlik Bütünlüğü Seviyeleri (ASIL) kullanılır [3]. ASIL 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 [4]. 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.

2.4. MISRA C

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 MISRA C adı verilen bir dizi yazılım geliştirme kılavuzu oluşturmuştur [5]. Otomotiv endüstrisinde kullanılan gömülü sistemlerde kod güvenliği, taşınabilirliği ve emniyetini sağlamayı amaçlamaktadır.

ISO 26262 uyumlu EKB 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.

3. Gerçek Zamanlı Sistemler

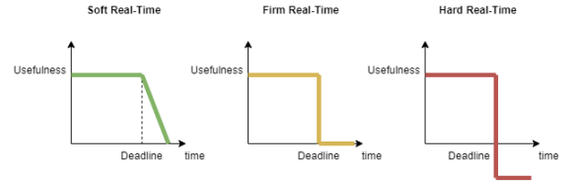
Bu bölüm gerçek zamanlı sistemlerden zamanlama algoritmalarına kadar detaylı bilgilendirmenin yapıldığı bölümdür. Gerçek zamanlı sistemlerin farklarından bahsedilecek ve gerçek zamanlı sistemlerde sınıflandırma yapılacaktır. Sonrasında zamanlama ile ilgili genel konsept anlatılıp zamanlama algoritmalarının sınıflandırılması ile tamamlanacaktır.

Bir sistem, bir veya daha fazla giriş kümesini ve girişlerle ilgili bir veya daha fazla çıkış kümesini içeren bir kara kutudur. Determinizmin sözlükteki anlamı, tüm olayların önceden var olan nedenlerle tamamen belirlenmesidir [6]. 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.

Belirli bir zamanda yanıt veren ve görevleri zamanlayan sistemler gerçek zamanlı sistemlerdir [7][8]. Sürücüsüz araçları düşündüğümüzde kameralar ve sensörler aracılığıyla yayaları saptarlar. Bu araçlar hızlarını tespit edilen yayalara göre ayarlarlar. Aracın yavaşlaması için acil bir istek gelebilir. Bu istek hızlı bir şekilde MİB ile haberleştirmeyi gerektirir. Gerçek zamanlı sistemler istenilen zamanda bu isteğe yanıt verebilme yeteneğine sahiptir. Şekil 4'te sistemlerin kullanım alanları gerçek-zamanlı olmayandan katı gerçek-zamanlıya spektrum şeklinde gösterilmektedir.

Şekil 4: Kullanım alanlarının spektrumu.

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 [9]. 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, sıkı gerçek zamanlı sistemler ve katı gerçek zamanlı sistemlerdir.



Şekil 5: Gerçek zamanlı sistemlerin davranışı.

Esnek gerçek zamanlı sistem, sistem zaman sınırından sonra yanıt verdiğinde yalnızca performans düşüşüne neden olan bir sistemdir. Sıkı gerçek zamanlı sistemi, esnek ve katı gerçek zamanlı sisteme göre bir ara sistemdir. Bu sistemdeki seyrek zaman sınırı kaçırmaları tolere edilebilir. Toleransı en düşük sistem katı gerçek zamanlı sistemdir. Zaman sınırından sonra verilen bir yanıt sistemde yıkıcı sonuçlara neden olabilir. Katı ve esnek gerçek zamanlı sistemler arasındaki genel karşılaştırma Tablo 1'de verilmiştir.

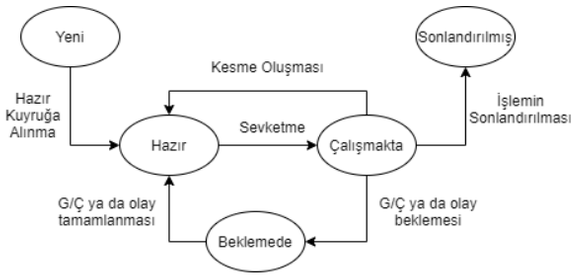
Tablo 1: Katı ve esnek gerçek-zamanlı sistem karşılaştırması

Parametre	Esnek Gerçek-Zamanlı Sistem	Katı Gerçek-Zamanlı Sistem
Yanıt Zamanı	Esnek-Tolere edilebilir	Sıkı-Gerekli
Güvenlik	Kritik değildir	Kritik
Zaman kısıt kaçırıldığında ki sonuçlar	Sistem kalitesinde tolere edilebilir düşme	Sisteme ciddi hasar oluşturma

Katı gerçek zamanlı sistemler [10], 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 MİB bulunur. Bu MİB, güvenlik sınıfına göre bu görevleri yönetmelidir. Bu amaçla MİB, görev zamanlaması sağlamak için zamanlama algoritmasına sahip zamanlayıcı kullanır.

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

Her MİB Zamanlayıcısı'nın görev yürütme durumları için bir durum makinesi vardır. Örnek durum makinesi Şekil 6'da gösterilmiştir.



Şekil 6: Görev durum makinası.

MİB bekleme durumuna girdiğinde işletim sistemi hazır kuyruğundan çalıştırmak için bir işlem seçmek zorundadır. Bu seçim MİB zamanlayıcısı tarafından yapılmaktadır. Durumlar arasında geçiş için karar veren iki farklı zamanlama tipi vardır. Bunlar öncelikli ve önceliksiz zamanlamadır. Öncelikli zamanlamada MİB bir işlem için ayrıldıysa çalışır durumdan hazır duruma veya bekleme durumundan hazır duruma geçme kararı verebilir. Önceliksiz zamanlamada MİB bir işlem için ayrıldıysa işlem bitene kadar beklenir.

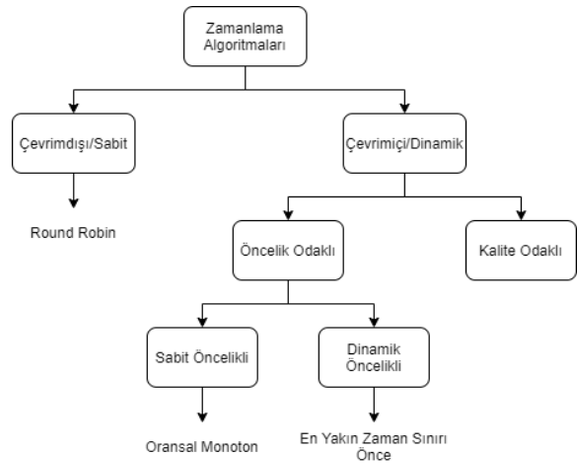
Görevlendirici MİB'ye atanacak işlemi seçen kısımdır. Görevlendiricinin çok hızlı geçiş yapması zorunludur. İşlemler arası geçiş süresine gönderim gecikmesi denir. MİB zamanlama algoritmaları, hazır kuyruğunda bekleyen işlemlerden hangisinin MİB'ye atanacağını belirler. MİB zamanlama algoritmaları birçok farklı kritere göre seçilir. Bunlar:

- MİB kullanımı
- Veri hacmi
- Bekleme süresi
- Yanıtlama süresi
- Dönüş süresi

MİB kullanımı ve veri hacmini maksimuma, bekleme ve yanıtlama süresini minimuma çekmek hedeflenir.

Temel olarak, programlama algoritmaları iki ana başlık altında kategorize edilir. Bunlar çevrimdışı / statik ve çevrimiçi / dinamik zamanlamadır [11]. Çevrimdışı zamanlamada, görevlerin listesini ve etkinleştirme zamanlarını içeren bir zamanlama tablosu vardır. Çalışma zamanında, basit bir dağıtıcı tabloda gösterilen kararları yürütür. Round Robin, 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 [12]. Öncelik odağı, 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. OM ve EYZSÖ, sırasıyla sabi öncelik ve dinamik öncelik zamanlaması için popüler zamanlama algoritmalarıdır.



Şekil 7: Zamanlama algoritmalarının sınıflandırılması.

4. Zamanlama Algoritmaları

Zamanlama algoritmalarının sınıflandırılması önceki bölümde yapılmıştır. Bu zamanlama algoritmaları ve ayrıntıları bu bölümde verilecektir. Şekil 7'ye baktığımızda iki ana kısım bulunmaktadır. Bunlardan biri dinamik kısımdır. Kurallar olayların özelliklerine göre çalışır zamanda işlenmesinden dolayı bu kısma olay tetiklemeli zamanlama mimarisi adını verebiliriz. Diğer kısım statiktir. Görevlerin EKDYS'leri derleme zamanında belirlendiği ve çalışır zamanda değişmediğinden bu bölüme de zaman tetiklemeli zamanlama mimarisi diyebiliriz.

4.1. Olay Tetiklemeli Zamanlama

Olay tetiklemeli yaklaşımda bir olay diğer bir olay tarafından tetiklenerek başlar. Olay tetiklemeli ve zaman tetiklemeli sistemlerde temel fark olay tetiklemeli sistemlerin hangi görevi işleyeceğine çalışır zamanda karar vermesidir. Kötümser yaklaşımlar her zaman sistemin işlevselliğini doğru bir şekilde ele almayı sağlar. Fakat bu yaklaşım olay-tetiklemeli sistemlerin davranışına kıyasla kaynak gereksinimlerinin gereğinden fazla tahmin edilmesine sebep olur. Bunun yanısıra en kötü durum senaryosunda doğru sistem davranışını garantilemek için zamanlama analiz metodları da olay tetiklemeli sistemlerde uygulanmaktadır. Zamanlama analiz yöntemlerinin temelinde kötü durum varsayımı ve periyodik işleme vardır. Böylelikle olay tetiklemeli sistemler kaynakları zaman tetiklemeli sistemlere nazaran daha etkili kullanmaktadır. Ancak tahmini kaynak gereksinimleri, katı gerçek zamanlı sistem gereksinimlerini karşılamak konusunda genellikle kötümserdir. Sonuç olarak güvenlik-kritik sistemler için olay tetiklemeli sistemler daha yüksek kaynak kullanımı konusunda tercih edilmemektedir. Buna rağmen olay tetiklemeli sistemler zaman tetiklemeli sistemlerden işlevsellik sağlayan olaylar için daha hızlı reaksiyon göstermektedir.

4.1.1. Oransal Monoton

Oransal Monoton, basit bir kurala sahip olan sabit öncelikli bir programlama algoritmasıdır [13]. Kural, görevlerin önceliklerinin derleme zamanında belirlenmesidir. Öncelikler 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, OM statik programlamaya EYZSÖ'den daha yakındır. Statik tarafa yaklaştıkça performans azalmaktadır. OM için üst sınır kullanımınının hesaplamasında, Liu ve Layland kanıtlarından yararlanabiliriz [14]. Bir task çalıştırılma süresini gösteren C_i, çalıştırılma sıklığını gösteren T_i, ve task sayısını gösteren n ile tanımlanır. Liu ve Layland, Şekil 8'de gösterilen hesaplamayı formülize etmişlerdir:

$$\left(U = \sum_{i=1}^{n} \frac{C_i}{T_i}\right) < n(2^{\frac{1}{n}} - 1)$$

Şekil 8: MİB kullanımı üst sınır değeri.

Bu kanıta göre MİB kullanım oranı U, n büyüdükçe ln(2)'ye yakınsar. Yani OM, ancak yaklaşık toplam yük yüzde 69'dan fazla değilse uygulanabilir bir planlamayı garanti edebilir [14]. Hesaplama adımları Şekil 9'da verilmiştir.

$$\begin{aligned}
\lim_{n\to\infty} n\left(2^{\frac{1}{n}} - 1\right) &= \lim_{n\to\infty} \left(\frac{-1 + 2^{\frac{1}{n}}}{\frac{1}{n}}\right) \\
&= \lim_{n\to\infty} \left(\frac{-\frac{ln\,2\ .\ 2^{\frac{1}{n}}}{n^2}}{-\frac{1}{n^2}}\right) \\
&= ln\,2\ .\ \lim_{n\to\infty} \left(2^{\frac{1}{n}}\right) \\
&= \lim_{n\to\infty} (ln\,2\ .\ 2^{\frac{1}{n}}) \\
&= ln\,2\ .\ \lim_{n\to\infty} \left(e^{\frac{1}{n} ln\,2}\right) \\
&= ln2 = 0.69314
\end{aligned}$$

Şekil 9: OM geçerlilik hesabının adımları.

4.1.2. En Yakın Zaman Sınırı Önce

EYZSÖ bir dinamik öncelik zamanlama algoritmasıdır [15]. 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 iyi performans gösterir. Liu ve Layland'ın kanıtlarına göre [14], EYZSÖ tam işlemci kapasitesinden yararlanabilir.

4.1.3. OM ve EYZSÖ karşılaştırılması

Dinamik zamanlama altındaki öncelik odaklı zamanlama algoritmalarına baktığımızda statik ve dinamik olarak ikiye ayrılmaktadır. OM ve EYZSÖ sırasıyla statik ve dinamik için popüler zamanlama algoritmalardır. Bu iki algoritma arasındaki karşılaştırma [16] Tablo 2'de verilmiştir.

Tablo 2: OM ve EYZSÖ karşılaştırması.

Parametre	OM	EYZSÖ
Öncelik	Statik	Dinamik
Uygulama Eforu	Düşük	Yüksek
Kural Kriteri	Periyot	Zaman Kısıtı
MİB Kullanımı	Düşük (~0.69)	Tam

4.2. Zaman Tetiklemeli Zamanlama

Zaman tetiklemeli zamanlamada [17], 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 bir dezavantajı da beraberinde getirir.

Bu zamanlama için, çalışma zamanından önce her şey bilinmelidir [18][19]. Bu da bir maliyet oluşturur. Ancak, geliştiriciler için önemli bir nitelik olan determinizmi sağlar.

5. Katı Gerçek-Zamanlı Sistemler

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, 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), Alman otomotiv şirketi gömülü sistemler konsorsiyumu tarafından kurulmuş bir standarttır [20]. Bu standartta bir iletişim yığını, bir ağ yönetim protokolü ve diğer ilgili konular hakkında spesifikasyonlar üretilmiştir. EKB'ler 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 olmuştur.

Piyasada yer alan çok sayıda işletim sisteminin bir kısmı otomotiv açısından bazı dezavantajlara sahiptir. Genellikle gerçek zamanlı performansları yeterli değildir. Uygulama Programlama Arabirimleri (UPA), eşzamanlı ve eşzamansız görev zamanlamayı birlikte kullanmak için uygun değildir. OSEK UPA, otomotiv endüstrisi için özel olarak tasarlanmıştır. Bu nedenlerden ötürü, OSEK İşletim Sistemi 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 EKB'lerde 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 işletim sistemidir [21]. AUTomotive Açık Sistem Mimarisi (AUTOSAR) UPA'sinin bir alt kümesinden esinlenen bir UPA uygular. Erika 2012 yılında sertifika almıştır ve daha sonra Erika'ya AUTOSAR İşletim Sistemi spesifikasyonları uygulanmıştır. OSEK/VDX uyumlu sertifikalı ilk açık kaynaklı ve ücretsiz gerçek zamanlı işletim sistemidir. Olay tetikleyici mimariyi ve zaman tetikleyici mimariyi destekler. Bu işletim sistemi 1-4 Kb Flash ayak izi ve 8-32 bit mikrodenetleyiciler için uygundur. Ayrıca, ayak izini sınırlamak için kullanılan uygunluk sınıflarına sahiptir. En önemli avantajlarından biri, bir OSEK Uygulama Dili (OUD) dosyası [22] veya AUTOSAR Genişletilebilir İşaretleme Dili ile statik olarak yapılandırılabilir.

6. Uygulama ve Değerlendirme

Tek işlemcili sistemler, programlama algoritmalarını değerlendirmek için iyi bir seçimdir. Görevler arasındaki bağlam geçişleri, kaçırılmış zaman sınırları ve her görev için MİB kullanımları bu sistemlerde açıkça görülebilir. Böylece, Arduino Uno uygulamak için seçilebilir. Erika İşletim Sistemi'nin bu kart için desteğinin bulunması da iyi bir avantajdır.

Arduino Uno, Atmega328 işlemci kullanan ve kullanımı oldukça kolay bir karttır [23]. 14 sayısal giriş çıkış pini vardır, bunların 6'sı darbe genişliği modülasyonu çıkışı (PWM) olarak kullanılabilir. 16 MHz kristal osilatör, evrensel seri veri yolu bağlantısı, 2.1mm güç girişi, devre içi seri programlama başlığı ve sıfırlama düğmesi vardır. Çalıştırmak için DC 7 ~ 12V güç kaynağına bağlanması yeterlidir.

Erika İşletim Sistemi resmi web sitesinden indirilmiştir. Bu işletim sistemi, Erika'nın geliştiricileri tarafından Erika resmi web sitesinde yayınlanan talimat kılavuzuna göre Arduino Uno kartına inşa edilerek gömülmüştür. Daha sonra ışık yayan diyotun yanıp sönmesi gibi bazı örneklerle test edilmiştir. Programlama algoritmalarının artılarını ve eksilerini göstermek için bir senaryo belirlenmiştir. Bu senaryoda sabit öncelikleri olan 3 görev vardır. Spesifikasyonları Tablo 3'te gösterilmiştir.

Tablo 3: Senaryoda kullanılan parametreler

Görev	Çalışma Süresi (s)	Periyot (s)
Task1	1	4
Task2	2	6
Task3	3	8

Bu görevlerin MİB Kullanımı Şekil 8'de belirtilen formüle göre hesaplanır. Hesaplama Şekil 10'de gösterilmiştir.

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

Şekil 10: Senaryoya göre MİB kullanım hesaplaması.

İşletim sistemini yapılandırmak için Bölüm 5'te belirtilen bir OUD dosyasına sahiptir. Bu dosya istenen ve gerekli konfigürasyona göre yazılmıştır. Her konfigürasyon algoritması için bazı konfigürasyonlar kolayca değiştirilebilir. OM ilk analiz zamanlama algoritmasıydı. Sabit öncelikli bir algoritmadır. OUD dosyası Şekil 11'deki 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;
};
```

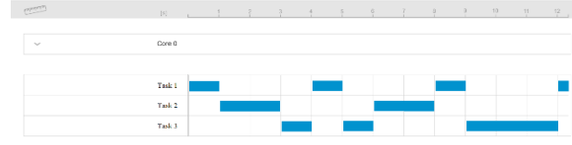
Şekil 11*:* OM için OUD dosyasının ayarlanması.

Bu görevlerin içeriği, araçlardaki EKB uygulamaları gibi uygulama sürelerine göre ayarlanmıştır. Görevlerin yürütme sürelerini göstermek için görevlere zaman damgalı Evrensel Asenkron Alıcı Verici (UART) günlükleri eklenmiştir. OM için UART günlükleri Şekil 12'de görülmektedir.

```
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 12: OM UART çıktısı.

Bu senaryonun sonuçlanan davranışı, OM zamanlama algoritmasının artılarını ve eksilerini göstermek için görselleştirilmiştir. İlgili görsel Şekil 13'te verilmiştir.



Şekil 13: OM çıktısının görselleştirilmesi.

Şekil 13'e göre, Görev 3 en düşük önceliğe sahiptir ve yürütme süresi 3 saniyedir ve 8. saniyede tamamlanması gerekirken zaman kısıtını kaçırmaktadır. Bu, OM 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, MİB kullanımını azaltarak ortadan kaldırılabilir. Bu analiz, bir sabit öncelikli zamanlama algoritması olan OM'nin performanstan fedakarlık ederek güvenli bir sistem oluşturmak için kullanılabileceğini gösteriyor.

Bir sonraki zamanlama algoritması, Dinamik-Öncelik Zamanlama Algoritması olan EYZSÖ'dür. Sabit öncelikliden daha az çaba gerektirir, çünkü görevleri zaman kısıtlarına göre otomatik olarak zamanlamaktadır. OUD dosyası EYZSÖ için yeniden düzenlenmiştir. KERNEL_TYPE değişkeni EYZSÖ olarak ayarlanmış olup Şekil 14'te 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;
};
```

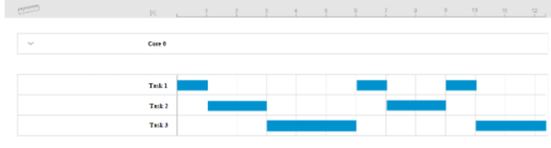
Şekil 14: EYZSÖ için OUD dosyasının ayarlanması.

Bu kez, EYZSÖ için aynı senaryo uygulanmaktadır. EYZSÖ için UART günlükleri Şekil 15'te verilmektedir.

```
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 15*: EYZSÖ* UART çıktısı.

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



Şekil 16: EYZSÖ çıktısının görselleştirilmesi.

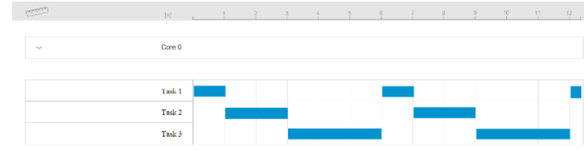
Şekil 16'ya bakıldığında, kaçırılmış bir zaman kısıtı yoktur. Bu bize EYZSÖ'nin sabit öncelikliden daha yüksek MİB kullanımı sağlayabildiğini göstermektedir. Ancak bu EYZSÖ'nin dezavantajları olmadığını göstermez. Zaman kısıtı zamanlayıcı, her görev için zaman kısıtı içerisinde yerine getirilmesini garanti eder, ancak herhangi bir görev için minimum yanıt süresi sağlamak mümkün değildir. En yüksek öncelikli görev her zaman OM cinsinden minimum yanıt süresine sahiptir, ancak EYZSÖ için garanti vermek mümkün değildir. Yeni bir görev veya yanlış EKDYS 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 EYZSÖ, 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 [19]. Bu mimaride çevrimdışı tanımlanmış görevleri ve bunların tetiklenme zamanlarını içeren zamanlama tablosu bulunur. Bu zamanlama tablosu OUD dosyasında hazırlanmıştır ve her görev belirli bir zamanda çağrılmaktadır. ZT için UART günlükleri Şekil 17'de görülmektedir.

```
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 17: ZT UART çıktısı.

Bu senaryonun sonuçlanan davranışı ZT zamanlama algoritmasının artılarını ve eksilerini göstermek için görselleştirilmiş olup Şekil 18'de verilmektedir.



Şekil 18: ZT çıktısının görselleştirilmesi.

Olay tetiklemeli zamanlama mimarisi ile zaman tetiklemeli zamanlama mimarisi arasında bir karşılaştırma yapmak gerekirse; Olay tetiklemeli mimari daha esnektir ve yüksek kaynak kullanımı doğurur. Öte yandan bu, katı gerçek zamanlı 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.

Benzer işlem farklı MİB kullanımları ile daha fazla görev için denendi. Sonuçlar Tablo 4'te listelenmiştir.

Tablo 4: Farklı parametrelere göre çıktılar.

	MİB Kullanımı (%)	Görev Sayısı	Başarısız Görev Sayısı	Başarılı Görev Sayısı	Başarı Oranı (%)
OM	<%85	5	0	5	%100
EYZSÖ			0	5	%100
ZT			0	5	%100
OM		10	0	10	%100
EYZSÖ			0	10	%100
ZT			0	10	%100
OM		15	0	15	%100
EYZSÖ			0	15	%100
ZT			0	15	%100
OM		20	0	20	%100
EYZSÖ			0	20	%100
ZT			0	20	%100
OM		25	0	25	%100
EYZSÖ			0	25	%100
ZT			0	25	%100
OM		30	0	30	%100
EYZSÖ			0	30	%100
ZT			0	30	%100
OM	%85-%98	5	1	4	%80
EYZSÖ			0	5	%100
ZT			0	5	%100
OM		10	2	8	%80
EYZSÖ			0	10	%100
ZT			0	10	%100
OM		15	3	12	%80
EYZSÖ			0	15	%100
ZT			0	15	%100
OM		20	3	17	%85
EYZSÖ			0	20	%100
ZT			0	20	%100
OM		25	4	21	%84
EYZSÖ			0	25	%100
ZT			0	25	%100
OM		30	4	26	%86.7
EYZSÖ			0	30	%100
ZT			0	30	%100
OM	%98-%100	5	1	4	%80
EYZSÖ			0	5	%100
ZT			0	5	%100
OM		10	3	7	%70
EYZSÖ			0	10	%100
ZT			0	10	%100
OM		15	4	11	%73.3
EYZSÖ			0	15	%100
ZT			0	15	%100
OM		20	4	16	%80
EYZSÖ			0	20	%100
ZT			0	20	%100
OM		25	5	20	%80
EYZSÖ			0	25	%100
ZT			0	25	%100
OM		30	5	25	%83.3
EYZSÖ			0	30	%100
ZT			0	30	%100

Önceki deneyde olduğu gibi OM MİB kullanımı arttıkça başarı oranı düşmeye başlayan ilk zamanlama algoritması olmuştur. Sonuçlara göre görev sayısı arttıkça zaman kısıtını aşma eşiği düşmektedir. Bu aynı zamanda Şekil 8'de verilen formülü kanıtlamaktadır. EYZSÖ olay tetiklemeli zamanlama algoritması olarak daha yüksek MİB kullanımında OM'den daha başarılı bir algoritmadır. Daha önce de bahsettiğimiz gibi ZT bazı dezavantajlara sahip olsa da periyodik dünyada en başarılı algoritmadır.

Olay tetiklemeli ve zaman tetiklemeli zamanlama mimarileri karşılaştırıldığında olay tetiklemeli mimarinin daha esnek olduğu ve yüksek kaynak kullanımı doğurması göze çarpmaktadır. Fakat katı gerçek zamanlı sistemler açısından önemli olan determinizm ve öngörülebilirlik açısından güvenilir değildir. Ayrıca olay tetiklemeli mimari zaman tetiklemeli mimariden daha fazla çalışır zaman yüküne sahiptir. Zaman tetiklemeli periyodik bir dünyaya sahiptir. Sporadik görevler için esnek değildir. Düşük kaynak kullanım oranlarında determinizm ve öngörülebilirlik sağlar.

7. 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 olduğu ve gereksinimlere göre seçilmesi gerektiğidir.

Güvenlik açısından kritik sistemler gibi sıkı güvenlik gereksinimlerine sahip sistemler için Oransal Monoton algoritması iyi bir seçim olacaktır. Daha fazla MİB kullanımı sağlamasından dolayı işlevsellik sağlamayı amaçlayan sistemlerin performans ihtiyacını karşılaması için en yakın zaman sınırı algoritmasını kullanması doğru seçim olacaktır. Deterministik sistemler öngörülebilirlik sağlamak adına zaman tetiklemeli algoritmayı tercih edebilirler.

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

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

8. Kaynaklar

[1] B. Saydam and T. Ayav, "Evaluation of Scheduling Architectures for OSEK/VDX Compliant Hard Real-Time Operating Systems," 2020 Turkish National Software Engineering Symposium (UYMS), 2020, pp. 1-6, doi: 10.1109/UYMS50627.2020.9247064.

[2] ISO: 26262 – "Road vehicles-Function safety Part 6: Product development at the software level", 2018.

[3] Redmill, Felix. "IEC 61508-principles and use in the management of safety." Computing & Control Engineering Journal, 9.5, pp. 205-213, 1998.

[4] Wang, Dafang & Song, Peng & Xu, Zexu & Dong, Guanglin & Wei, Hui, "Conceptual Design of Functional Safety of Motor Control System Based on ISO26262", MATEC Web of Conferences, 173. 02045. 10.1051/matecconf/201817302045, 2018.

[5] Introduction to MISRA C, URL: https://www.embedded.com/introduction-to-misra-c/ (Last accessed: Jan, 5, 2022).

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

[7] John A. Stankovic, Krithi Ramamritham, and Marco Spuri, Deadline Scheduling for Real-Time Systems: Edf and Related Algorithms, Kluwer Academic Publishers, USA, 1998.

[8] M. Caccamo, T. Baker, A. Burns, and G. Buttazzo, "Real-time scheduling for embedded systems," in Handbook of Networked and Embedded Systems, pp. 173-195, Birkhäuser Boston, 2005.

[9] Buttazzo, Giorgio C. Hard real-time computing systems: predictable scheduling algorithms and applications. Vol. 24. Springer Science & Business Media, 2011.

[10] A. Burns, "Scheduling hard real-time systems: a review", Software Engineering Journal, 6(3), 116-128, 1991.

[11] Weirong Wang, Aloysius K. Mok, Gerhard Fohler, "Pre-Scheduling: Integrating Offline and Online Scheduling Techniques", EMSOFT 2003, Volume 2855, pp. 356-372, 2003.

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

[13] Bini, E., Buttazzo, G. C., & Buttazzo, G. M. "Rate monotonic scheduling: The hyperbolic bound", IEEE Transactions on Computers, Volume 52, Issue 7, 2003, pp. 933–942, https://doi.org/10.1109/TC.2003.1214341.

[14] C. Liu and J. Layland, "Scheduling Algorithms for Multiprogramming in a Hard Real-time Environment", JACM, vol. 20, no. 1, 1973.

[15] Zhang F, "Analysis for EDF scheduled real-time systems", PhD thesis, Dept of Computer Science, University of York, UK, 2009.

[16] Buttazzo, G. "Rate monotonic vs. EDF: Judgment day", EMSOFT 2003, Volume 2855, pp. 67-83, 2003.

[17] Kopetz, Hermann, and Günther Bauer. "The time-triggered architecture." Proceedings of the IEEE 91.1 : 112-126, January 2003.

[18] Kopetz, Hermann. Real-time systems: design principles for distributed embedded applications. Springer Science & Business Media, 2011.

[19] Kopetz, Hermann. "Event-triggered versus time-triggered real-time systems." Operating Systems of the 90s and Beyond. Springer, Berlin, Heidelberg, pp. 86-101, 1991.

[20] Lemieux, Joseph. "The OSEK/VDX Standard: Operating System and Communication." Embedded Systems Programming 13.3: pp. 90-109, 2000.

[21] Erika-enterprise.com. URL:http://www.erika-enterprise.com/index.php/community.html/ (Last accessed: Apr, 11, 2020].

[22] Feiler, Peter H. (2018): Real-Time Application Development with OSEK: A Review of the OSEK Standards. Carnegie Mellon University. Report. https://doi.org/10.1184/R1/6582899.v1.

[23] Arduino.cc, URL: https://www.arduino.cc/ (Last accessed: May, 19, 2020].

Özgeçmişler

Berkay SAYDAM, İzmir doğumlu olan Berkay SAYDAM, Dokuz Eylül Üniversitesi Elektrik-Elektronik Mühendisliği bölümü lisans eğitimini 2017 yılında onur öğrencisi olarak tamamlamıştır. 2020 yılında İzmir Yüksek Teknoloji Enstitüsü Bilgisayar Mühendisliği bölümü yüksek lisans eğitimini yüksek onur öğrencisi olarak tamamlamıştır ve aynı bölümde doktora eğitimini sürdürmektedir. Lisans öğrenimi mezuniyetinden itibaren özel sektör firmalarının araştırma geliştirme departmanlarında çeşitli pozisyonlarda görev almış ve sektörde çalışmalarına devam etmektedir. Multidisipliner yaklaşım benimsemesinden kaynaklı gömülü sistemler ve nesnelerin interneti alanlarında çalışmasının yanı sıra araştırma alanları arasında veri madenciliği ve makine öğrenmesi de bulunmaktadır.

Tolga AYAV 1995 yılında Dokuz Eylül Üniversitesi Elektrik-Elektronik Mühendisliği Bölümü'nden mezun olmuştur. Endüstriyel otomasyon alanında özel sektörde çalıştıktan sonra 1999'da İzmir Yüksek Teknoloji Enstitüsü Bilgisayar Mühendisliği'nden yüksek lisans ve 2004'te Ege Üniversitesi Bilgisayar Mühendisliği'nden doktora derecelerini almıştır. 2004-2006 süresince INRIA Rhone-Alpes'te doktora sonrası araştırmacı olarak çalışmıştır. Dr. Ayav 2006 yılından itibaren İYTE Bilgisayar Mühendisliği Bölümü'nde öğretim üyesi olarak akademik çalışmalarını sürdürmektedir. Akademik araştırmaları ağırlıklı olarak gerçek zamanlı ve gömülü sistemler, hataya dayanıklılık, yazılım test ve sınama üzerinedir.