

Ege Elektromobil MOTOR SÜRÜCÜ

Yazılım Gereksinim Spesifikasyon (SRS) Dokümanı

*Atakan Ertekin*

[*http://www.egeelektromobiltakimi.com/*](http://www.egeelektromobiltakimi.com/) *2025*

**Revizyonlar**

|  |  |  |  |
| --- | --- | --- | --- |
| **Versiyon** | **Yazar(lar)** | **Versiyon Açıklaması** | **Tamamlanma Tarihi** |
| 1.0.0-0 | Atakan Ertekin | Uygulamanın yüksek seviyeli tasarımının ilk versiyonu tamamlanmıştır. | 20 Ocak 2025  20.01.2025 |
| 1.0.0-1 | Ahmet Vedat Özkılıç | Basit düzenleme ve düzeltmeler yapılmıştır. | 13 Şubat 2025  13.02.2025 |
| 1.0.0-2 | Ahmet Vedat Özkılıç | Basit düzenleme ve düzeltmeler yapılmıştır. | 14 Şubat 2025  14.02.2025 |

**Gözden geçiren & Onaylayan**

**Gereksinim dokümanları onaylanan geçmiş versiyonlar**

|  |  |  |  |
| --- | --- | --- | --- |
| **Onaylayan(lar)** | **Onaylanan Versiyon** | **İmza** | **Tarih** |
|  |  |  |  |
|  |  |  |  |

**Gereksinim dokümanlarının gözden geçirenler**

|  |  |  |  |
| --- | --- | --- | --- |
| **Gözden geçiren(ler)** | **Onaylanan Gözden geçirme** | **İmza** | **Tarih** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

1. [Giriş 3](#_bookmark0)

**İÇİNDEKİLER**

1. [Genel Açıklamalar 3](#_bookmark1)
2. [Fonksiyonel Gereksinimler 4](#_bookmark2)
3. Arayüz Sınıf Tanımlamaları 6
4. [Yazılım Komponentleri 6](#_bookmark3)
5. Arayüz Tanımlamaları 6
6. [Test Senaryoları 8](#_bookmark4)
7. [Görsel Modeller ve Diyagramlar 8](#_bookmark5)
   1. [Temel Flowchart Diyagramları 8](#_bookmark6)
   2. [Sıralama Diyagramları 9](#_bookmark7)
8. [Güncelleme Kayıt Defteri 9](#_bookmark8)
9. [Güncellenen Bütçe Kaydı 9](#_bookmark9)
10. [Ekler 9](#_bookmark10)

# 1. Giriş

### Giriş

Bu projede, **bare-metal** bir mimari kullanılarak, **interrupt-driven event** mantığında çalışan bir motor sürücü sistemi tasarlanacaktır. Sistem, HALL sensörlerinden alınan verileri işleyerek motorun konumunu belirleyecek ve bir **smart gate driver** üzerinden motor kontrolü gerçekleştirecektir. Ayrıca, sistem CAN ve Diagnostics stack’i üzerinden iletişim kurarak gerekli haberleşme ve teşhis işlevlerini yerine getirecektir.

### Dokümanın kapsamı

* 1. **Genel Bakış**

Proje, **gerçek zamanlı** bir kontrol ve iletişim sistemi geliştirmeyi hedefler. Sistemin genel yapısı üç ana fonksiyonel bileşenden oluşmaktadır:

* + 1. **Motor Drive Feature:** Motorun konum, hız ve yön bilgilerini kontrol ederek sürüş fonksiyonlarını gerçekleştirir.
    2. **Analog Configuration Feature:** Analog sinyallerin hassas bir şekilde işlenmesi ve yapılandırılması görevini üstlenir.
    3. **Communication Feature:** CAN protokolü ve Diagnostics stack üzerinden iletişim kurarak veri alışverişini ve sistem teşhisini sağlar.

Bu mimari, hem yüksek performans hem de güvenilir bir iletişim altyapısı sunmayı amaçlamaktadır.

# 2. Genel Açıklamalar

### Temel Fonksiyon tanımlaması

Bu proje, HALL sensörlerinden alınan verilerle motorun konum, hız ve yön kontrolünü

gerçekleştiren, **smart gate driver** üzerinden hassas motor sürüşü sağlayan bir sistem tasarımıdır.

Ayrıca, CAN protokolü ve Diagnostics stack kullanılarak güvenilir bir iletişim altyapısı oluşturur. Sistem, analog sinyallerin doğru işlenmesini ve yapılandırılmasını desteklerken, kesme tabanlı olay yönetimi sayesinde hızlı ve gerçek zamanlı bir performans sunar.

### Fonksiyonel Hedefler

Projenin temel hedefi, motor kontrolünde yüksek hassasiyet sağlarken, güvenilir haberleşme ve doğru analog yapılandırmayı bir araya getirerek endüstriyel standartlara uygun, yüksek performanslı bir çözüm geliştirmektir.

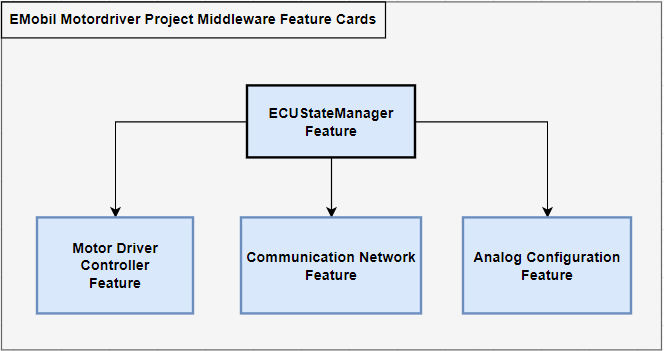
### Genel Kısaltmalar

* + 1. **ECU -** Electronic Control Unit
    2. **PDU -** Protocol Data Unit
    3. **CompReqs -** Component Requirements
    4. **TC -** Test Case
    5. **BSW -** Basic Software
    6. **CAN -** Controller Area Network

# 3. Fonksiyonel Gereksinimler

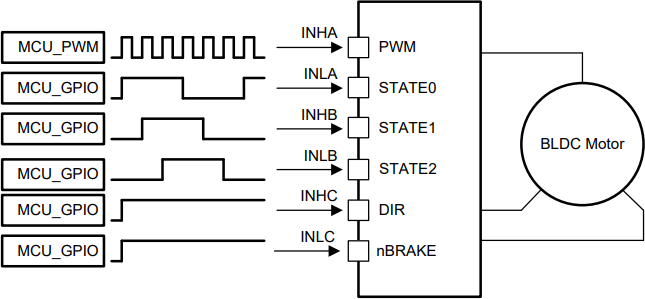
1. Sistem, **ECUStateManager** üzerinden, tüm modüller arasında senkronize bir şekilde çalışan bir **state machine** mimarisi ile gerçek zamanlı ve güvenilir bir kontrol sağlamalıdır. Bu yapı,

### Motor Driver Controller, Communication Network ve Analog Configuration özelliklerini birleştirerek, motor kontrolü, haberleşme ve analog sinyal işleme süreçlerini optimize ederken, yüksek performans ve hata yönetimi mekanizmalarını içermelidir. Her bir modül, sistemin genel çalışma durumlarına uygun olarak, düşük gecikme, yüksek güvenilirlik ve protokollere uygunluk esaslarına dayanarak çalışmalıdır.



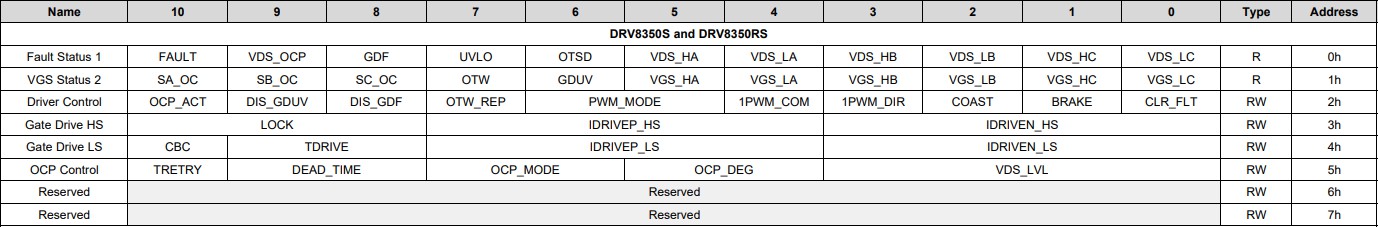
**Figure 1. Motor Driver Feature Cards**

1. ECUStateManager, bare-metal bir yapıda, interrupt-driven state machine yaklaşımıyla sistemin tüm çalışma durumlarını yönetir. STARTUP, ROUTINE ve PROCESS\_STATE gibi temel durumlarla, sistemin başlatılmasından rutin işlemlerin yönetimine ve işlem süreçlerinin tamamlanmasına kadar her adımı kontrol eder. Durumlar arası geçişler, refresh state ve process queue gibi mekanizmalarla senkronize edilir ve sistemin kesintisiz, gerçek zamanlı bir şekilde çalışması sağlanır.
   1. Sistem, **bare-metal** bir yapıda çalışır ve tüm süreçler bir **state machine** üzerinden kontrol edilir.
   2. **Interrupt-driven event** tabanlı bir mimari kullanılarak olaylar hızlı ve güvenilir bir şekilde işlenir
   3. **State machine** içinde aşağıdaki ana durumlar ele alınır
      1. STARTUP (PRE\_INIT, INIT, POST\_INIT) : Sistem başlatıldığında, başlangıç işlemleri **PRE\_INIT**, **INIT**, ve **POST\_INIT** durumları üzerinden yürütülür. Bu süreçte, gerekli bileşenler başlatılır ve ECU, rutin işlemler için hazırlanır.
      2. INIT\_STATE : Sistem, başlatma işlemlerini gerçekleştirir.
      3. REFRESH\_STATE : Sistem, rutin işlemleri bu durumda gerçekleştirir. Durumlar arasında döngüsel geçişler yapılır ve işleme alınacak görevler kontrol edilir.
      4. PROCESS\_STATE : İşlem sürecinde, veriler işlenir ve gerekli görevler gerçekleştirilir. Sistem, işlem tamamlandığında rutin durumuna geri döner.
   4. İşlem sırası bir kuyruğa alınır ve her bir görev sırayla işlenir. processQueue, birden fazla işlem için sıranın saklanmasını sağlar. Her işlem sırasıyla işlenir ve tamamlandıktan sonra PROCESSSTATE\_DEFAULT olarak işaretlenir.
2. MotorDrive Feature, motor kontrolü için gerekli olan tüm alt modülleri bir araya getirerek HALL sensörlerinden alınan verilerin işlenmesini, PWM yapılandırmasının ayarlanmasını, iletişim protokolleri üzerinden veri alışverişini ve hata yönetimini sağlar. Sistem, abstract class'lar ile esneklik ve genişletilebilirlik sunarken, her bileşen özel bir sorumluluk üstlenir.
   1. HALL Sensör Yönetimi: HALL sensörlerinden gelen IRQ sinyalleri **InterruptHandlerManager** tarafından okunmalı ve işlenmelidir. HALL sensör verileri **HALLSensorConfiguration** tarafından toplanmalı ve motor hız, yön ve PWM sıralamasına dönüştürülmelidir.



**Figure 2. MCU PWM Configuration Logic**

* 1. PWM Yönetimi: **PWMConfiguration**, motor kontrolü için PWM adımları, hız, fren ve yön bilgilerini düzenlemelidir. PWM yapılandırması sırasında GPIO ve Timer bileşenleri kullanılarak zamanlama ve çıkış pin durumları kontrol edilmelidir.



**Figure 3. Register Map**

* 1. Gate Driver Kontrolü: **GateDriverController**, sistemin kurulum işlemlerini gerçekleştirmeli ve **fault register** bilgilerini sürekli okuyarak hataları izlemelidir.

Hatalı durumlarda, sistemin güvenli bir duruma geçmesi için fault yönetimi sağlanmalıdır.

* 1. İletişim ve Veri Yönetimi: **MotorDriverCommProxy**, fren ve hız bilgilerini PDU formatında **Communication Stack** üzerinden aktarmalıdır. **MotorDriverDcmProxy**, sistemdeki hata tespitlerini **Diagnostics Stack** üzerinden bildirmelidir.
  2. Hata Yönetimi: Her bileşen kendi hata durumlarını kontrol etmeli ve interface arayüzü üzerinden diyagnostik birimine raporlamalıdır.

1. <Communication Network Feature>
2. <Analog Configuration Feature>

# 4. Arayüz Sınıf Tanımlamaları

|  |  |
| --- | --- |
| **Abstract Sınıf** | **Tanım** |
| Abstract\_CoreManager | Tüm temel interrupt ve GPIO durumlarını yöneten genel bir arayüz sağlar. |
| Abstract\_MotorDriver | Motorun hız, fren ve yön yönetimi için gerekli olan arayüzleri içerir. |
| Abstract\_SmartGateDriver | Gate driver yapılandırması ve fault yönetimi için soyut arayüzleri içerir. |
| Abstract\_VehComManager | Hız, fren ve gösterge verilerinin iletişimini sağlayan arayüzleri tanımlar. |

# 5. Yazılım Komponentleri

|  |  |
| --- | --- |
| **Komponent Sınıfı** | **Tanım** |
| ECUStateManager | IRQ sinyallerini işleyen ve HALL sensör verilerini yöneten modül. |
| InterruptHandlerManager | HALL sensör durumlarını okuyup PWM sıralamasını ayarlayan yapı. |
| HALLSensorConfiguration | PWM adımları, hız, fren ve yön durumlarını düzenleyen modül. |
| PWMConfiguration | Fren ve hız bilgilerinin yönetimini sağlayan ana yapı. |
| MotorDriverControlManager | Fren ve yön verilerinin yönetimini üstlenen bileşen. |
| DirectionAndBrakeManager | Gate driver'ın kurulumu ve fault register okumasını yapan bileşen. |
| GateDriverController | Verileri PDU formatında ileten iletişim bileşeni. |
| MotorDriverCommProxy | Hata tespitlerini Diagnostics Stack'e aktaran bileşen. |
| MotorDriverDcmProxy | Hata tespitlerini Diagnostics Stack'e aktaran bileşen. |

# 6. Arayüz Tanımlamaları

**Abstract\_CoreManager**

|  |  |
| --- | --- |
| **Arayüz** | **Tanım** |
| IExternalInterruptStatus | Dış interrupt sinyallerini okuma ve yönetim işlevlerini tanımlar. |
| IGPIOStatus | GPIO pinlerinin durumlarını kontrol ve okuma işlevlerini sağlar. |
| ITimerInterruptStatus | Timer tabanlı interrupt işlemlerini yönetmek için kullanılır. |

**Abstract\_MotorDriver**

|  |  |
| --- | --- |
| **Arayüz** | **Tanım** |
| IBrakeAndDirection | Fren ve yön sinyallerinin kontrolünü sağlar. |
| IBrakeStatus | Fren durumunun yönetimini tanımlar. |
| IDirectionStatus | Motor yön bilgilerini okuma ve kontrol işlevlerini sağlar. |
| IHALLA, IHALLB, IHALLC | HALL sensörlerinden alınan durum sinyallerini tanımlar. |
| IHALLWheelSpeed | HALL sensöründen alınan tekerlek hız bilgilerini sağlar. |
| IIndicatorBdrStatus | Fren ve yön göstergelerinin durumlarını yönetir. |
| IIndicatorSpeedStatus | Hız göstergelerinin durumlarını okuma ve yönetim işlevlerini tanımlar. |
| IMotorDriveError | Motor sürücü hatalarının tanımlanması ve raporlanması için kullanılır. |
| IPwmStep | PWM adımları için kontrol işlevlerini tanımlar. |
| IRawSpeed | Motorun ham hız bilgilerini sağlar. |
| ISpeedStatus | Motor hız durumunu okuma ve yönetme işlevlerini tanımlar. |

**Abstract\_SmartGateDriver**

|  |  |
| --- | --- |
| **Arayüz** | **Tanım** |
| IDRV8353 | DRV8353 gate sürücüsü ile ilgili kontrol ve durum okuma işlevlerini sağlar. |

**Abstract\_VehComManager**

|  |  |
| --- | --- |
| **Arayüz** | **Tanım** |
| IBdr | Fren (brake) bilgileri için iletişim arayüzünü tanımlar. |
| IIndicator | Göstergelerle ilgili iletişim işlemlerini sağlar. |
| ISpeed | Hız bilgilerini iletişim için düzenler. |
| IWheelSpeed | Tekerlek hız bilgilerini işleme ve iletişim için tanımlar. |

# 7. Test Senaryoları

## Senaryo A: ECUStateManager SIL TEST

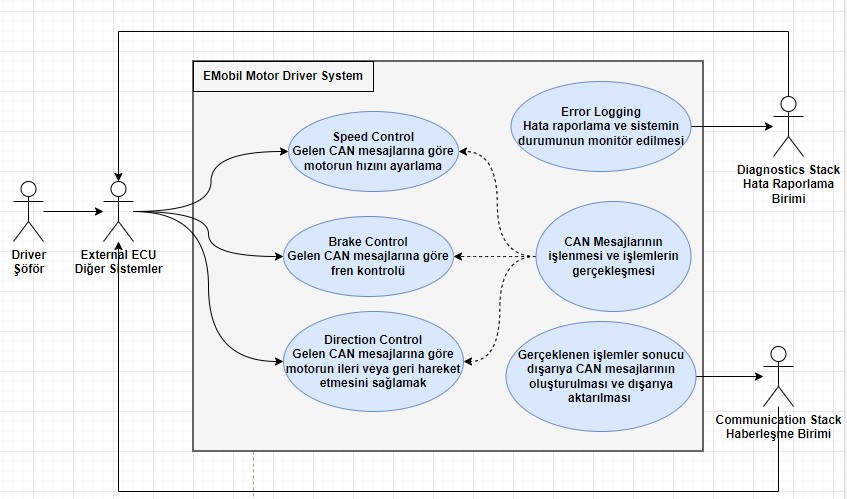
<Gerçekleşebilecek test senaryo açıklaması>

## Senaryo B: MotorDrive SIL TEST

<Gerçekleşebilecek test senaryo açıklaması>

# 8. Görsel Modeller ve Diyagramlar

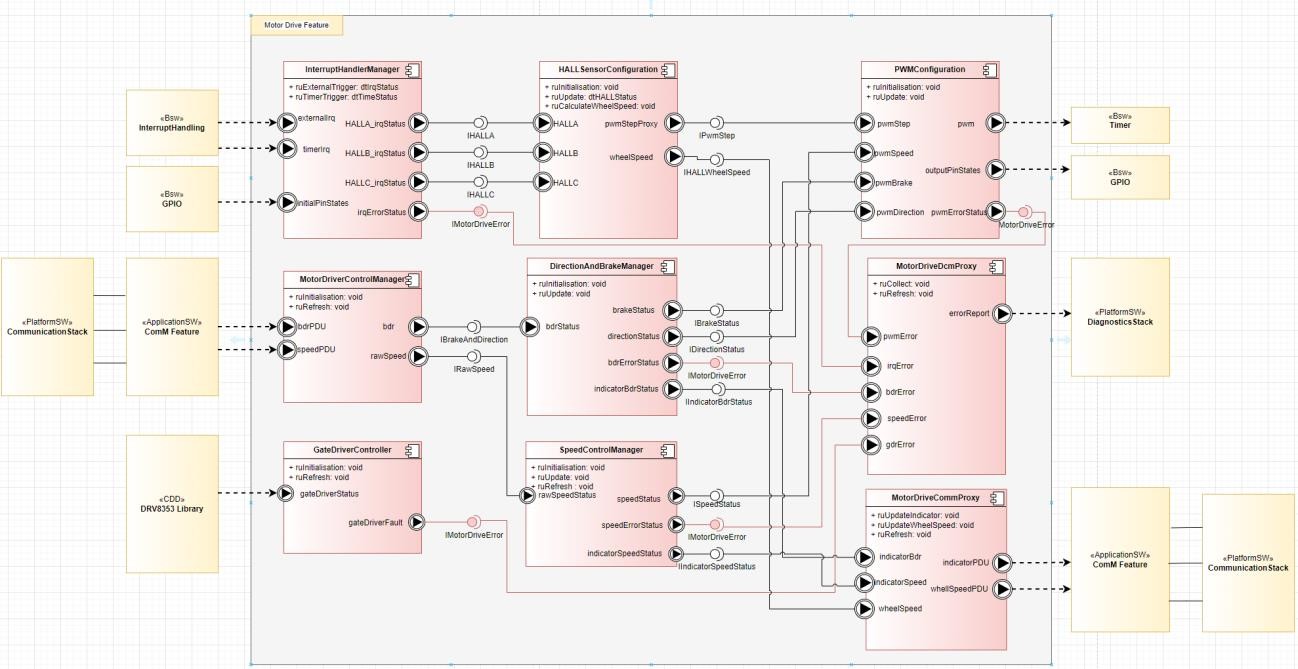
* 1. **Use Case Diagram**

****

E-Mobil Motor Driver System’in temel işlevlerini ve dış aktörlerle etkileşimini göstermektedir.

Driver ve External ECU tarafından gönderilen CAN mesajları, sistem tarafından işlenerek motorun hız, fren ve yön kontrolü sağlanır. Error Logging mekanizması hata durumlarını izler ve Diagnostics Stack’e raporlar, aynı zamanda sistemden gelen veriler Communication Stack aracılığıyla dış sistemlere aktarılır. Tüm bu işlemler, güvenilir ve gerçek zamanlı bir motor kontrolü için optimize edilmiştir.

* 1. **Motor Driver Feature Diagram V1.0.0 [20/1/2025]**

****

Bu diagram, **Motor Drive Feature**’ın temel yapı taşlarını ve bileşenler arası etkileşimleri özetlemektedir. Sistem, HALL sensörlerinden alınan verileri **InterruptHandlerManager** ile işleyerek motorun hız, yön ve fren kontrolü için gerekli PWM sinyallerini **PWMConfiguration** üzerinden üretir. **MotorDriverControlManager**, hız ve fren bilgilerini **DirectionAndBrakeManager** ve **SpeedControlManager** aracılığıyla yönetirken, **GateDriverController** motor sürücüsünün fault durumlarını izler ve yönetir.

**MotorDriverCommProxy**, üretilen PDU’ları iletişim protokolü üzerinden dış sistemlere aktarırken, **MotorDriverDcmProxy** hata tespitlerini **Diagnostics Stack**’e iletir. Bu yapı, modüler ve güvenilir bir motor kontrol sistemi sağlar.

* 1. **Communication Network Feature Diagram**
  2. **Analog Configuration Network Feature Diagram**

# 9. Güncelleme Kayıt Defteri

Versiyon güncellemeleri sonunda hangi versiyon ne amaçla değiştiği buraya madde madde yazılır.

V1.0.0-1 Basit düzenleme ve düzeltmeler yapılmıştır.

V1.0.0-2 Basit düzenleme ve düzeltmeler yapılmıştır.

# 10. Güncellenen Bütçe Kaydı

Güncellenen versiyon bütçe gerektiren bir güncelleme ise buraya kayıt tutulmaktadır.

# 11. Ekler