Ege Elektromobil
MOTOR SÜRÜCÜ
Yazılım Gereksinim Spesifikasyon (SRS) Dökümanı

Atakan Ertekin

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

Gözdengeçiren & Onaylayan

Gereksinim dökümanları onaylanan geçmiş versiyonlar

Onaylayan(lar)	Onaylanan Versiyon	İmza	Tarih

Gereksinim dökümanlarının gözdengeçirenler

Gözdengeçiren(ler)	Onaylanan Gözdengeçirme	İmza	Tarih

İÇİNDEKİLER

1. Giriş

1.1 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.

1.2 Dökümanın kapsamı

1.3 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

2.1 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.

2.2 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.

2.3 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.

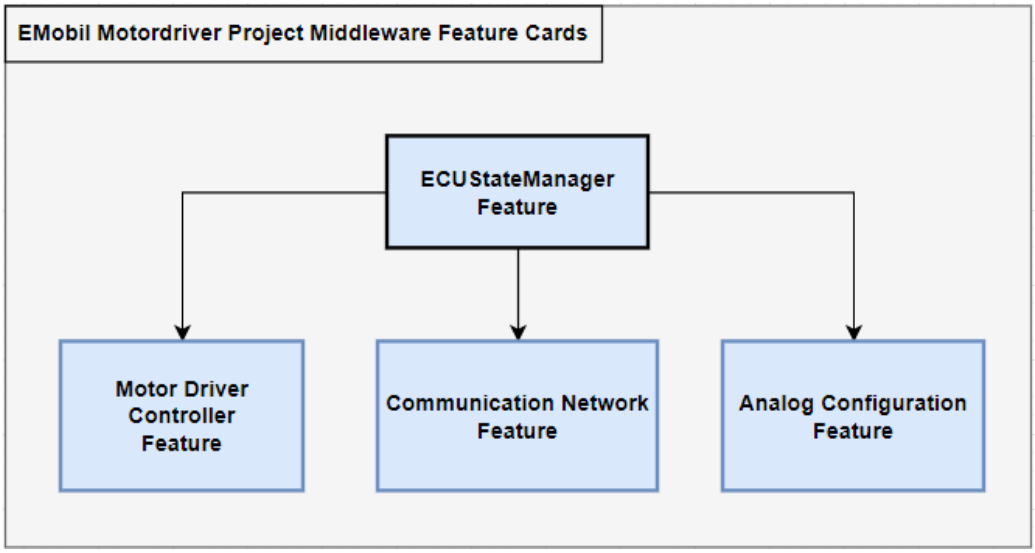


Figure 1. Motor Driver Feature Cards

2. 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.
    - 2.1. Sistem, **bare-metal** bir yapıda çalışır ve tüm süreçler bir **state machine** üzerinden kontrol edilir.
    - 2.2. **Interrupt-driven event** tabanlı bir mimari kullanılarak olaylar hızlı ve güvenilir bir şekilde işlenir
    - 2.3. **State machine** içinde aşağıdaki ana durumlar ele alınır
        - 2.3.1. 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.
        - 2.3.2. INIT_STATE : Sistem, başlatma işlemlerini gerçekleştirir.
        - 2.3.3. REFRESH_STATE : Sistem, rutin işlemleri bu durumda gerçekleştirir. Durumlar arasında döngüsel geçişler yapılır ve işleme alınacak görevler kontrol edilir.
        - 2.3.4. PROCESS_STATE : İşlem sürecinde, veriler işlenir ve gerekli görevler gerçekleştirilir. Sistem, işlem tamamlandığında rutin durumuna geri döner.
    - 2.4. 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.

3. 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.
    3.1. 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.

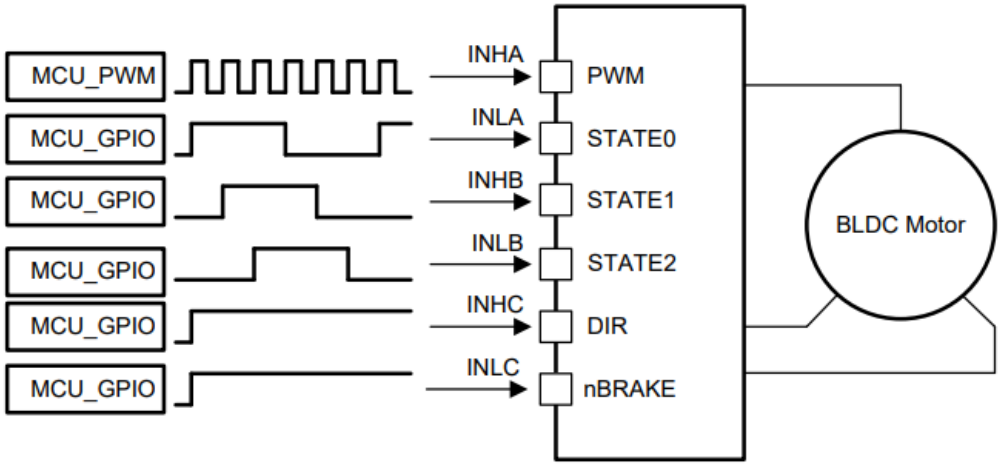


Figure 2. MCU PWM Configuration Logic

    3.2. 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.

Name	10	9	8	7	6	5	4	3	2	1	0	Type	Address
DRV8350S and DRV8350RS													
Fault Status 1	FAULT	VDS_OCP	GDF	UVLO	OTSD	VDS_HA	VDS_LA	VDS_HB	VDS_LB	VDS_HC	VDS_LC	R	0h
VGS Status 2	SA_OC	SB_OC	SC_OC	OTW	GDUV	VGS_HA	VGS_LA	VGS_HB	VGS_LB	VGS_HC	VGS_LC	R	1h
Driver Control	OCP_ACT	DIS_GDUV	DIS_GDF	OTW_REP	PWM_MODE		1PWM_COM	1PWM_DIR	COAST	BRAKE	CLR_FLT	RW	2h
Gate Drive HS	LOCK			IDRIVEP_HS				IDRIVEN_HS				RW	3h
Gate Drive LS	CBC	TDRIVE		IDRIVEP_LS				IDRIVEN_LS				RW	4h
OCP Control	TRETRY	DEAD_TIME		OCP_MODE		OCP_DEG		VDS_LVL				RW	5h
Reserved	Reserved											RW	6h
Reserved	Reserved											RW	7h

Figure 3. Register Map

    3.3. 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.
    3.4. İletişim ve Veri Yönetimi: **MotorDriverCommProxy**, fren ve hız bilgilerini PDU formatında **Communication Stack** üzerinden aktarmalıdır. **MotorDriverDcmProxy**, sistemdeki hata tespitlerini **Diagnostics Stack** üzerinden bildirmelidir.
    3.5. Hata Yönetimi: Her bileşen kendi hata durumlarını kontrol etmeli ve interface arayüzü üzerinden diyagnostik birimine raporlamalıdır.
4. <Communication Network Feature>
5. <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

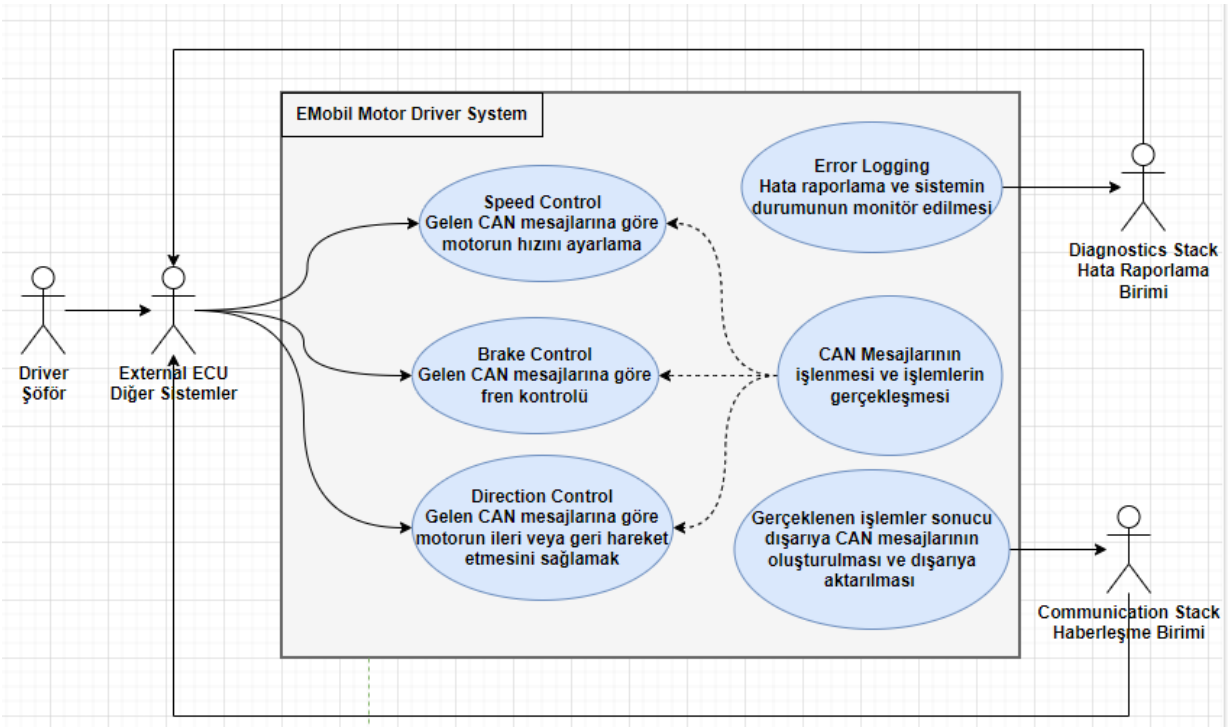
<Gerçekleşebilecek test senaryo açıklaması>

Senaryo B: MotorDrive SIL TEST

<Gerçekleşebilecek test senaryo açıklaması>

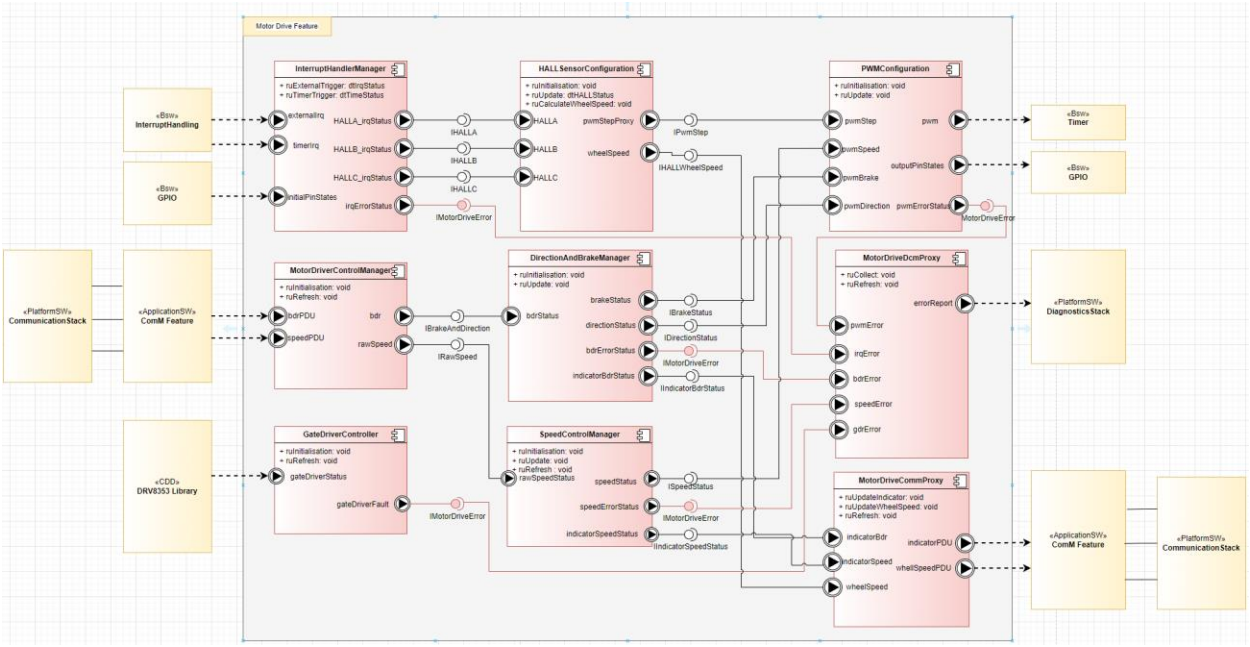
8. Görsel Modeller ve Diyagramlar

8.1 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.

8.2 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.

8.3 Communication Network Feature Diagram

8.4 Analog Configuration Network Feature Diagram

9. Güncelleme Kayıt Defteri

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

10. Güncellenen Bütçe Kaydı

Güncellene versiyondan bütçe gerektiren bir güncelleme ise buraya kayıt tutulmaktadır.

11. Ekler