Ege Elektromobil BrakeAndDirection Komponent 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	BrakeAndDirectionManager komponentinin gereksinim dökümanı oluşturulmuştur. Temel gereksinimler eklenmiştir.	10.Şubat 2025
1.0.1-0	Ahmet Vedat		

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 doküman, **BrakeAndDirectionManager** komponentinin fonksiyonlarını, gereksinimlerini, kullanılan arayüzleri ve test senaryolarını açıklamaktadır. Komponent, fren ve yön bilgilerini okuyarak yönetir ve ilgili arayüzler aracılığıyla sistemin diğer bileşenlerine iletir.

1.2 Dökümanın kapsamı

Bu döküman, BrakeAndDirectionManager'ın işleyişini tanımlayan işlevsel gereksinimleri, kullanılan arayüzleri ve doğrulama için test senaryolarını içermektedir. Sistem içindeki konumu ve diğer bileşenlerle olan etkileşimleri açıklanmaktadır.

1.3 Genel Bakış

BrakeAndDirectionManager, aracın fren ve yön kontrolünü sağlayan bir yazılım bileşenidir. Bu bileşen, **MotorDriverControlManager** üzerinden gelen fren ve yön bilgilerini işler, sistemde uygun güncellemeleri yapar ve **IBrakeStatus** ve **IDirectionStatus** arayüzleri aracılığıyla güncel bilgileri yayınlar.

2. Genel Açıklamalar

2.1 Temel Fonksiyon tanımlaması

BrakeAndDirectionManager'ın temel fonksiyonları:

- **Fren durumu okuma ve yönetme: MotorDriverControlManager**'dan gelen fren durumunu değerlendirerek sistemde güncellemeler yapar.
- **Yön durumu okuma ve yönetme: MotorDriverControlManager**'dan gelen yön bilgisini değerlendirerek güncel yön bilgisini belirler.
- **Fren ve yön durumlarını yayınlama: IBrakeStatus** ve **IDirectionStatus** arayüzleri üzerinden güncellenmiş durum bilgilerini paylaşır.
- **Hız yönetimi ile entegrasyon: SpeedControlManager_ruBrakeUpdate()** çağrılarak fren durumundaki hız yönetimi sağlanır.

2.2 Fonksiyonel Hedefler

BrakeAndDirectionManager, aracın fren ve yön durumlarını periyodik olarak okuyup güncelleyerek, fren değişikliklerini algılar ve uygun aksiyonları gerçekleştirir. Yön değişikliklerini yalnızca durma veya geri vites senaryolarında uygular, hız kontrol yönetimiyle entegre çalışarak fren durumunu günceller ve hata senaryolarında uygun hata yönetimi mekanizmalarını sağlar.

3. Fonksiyonel Gereksinimler

Bu bölümde, **BrakeAndDirectionManager** komponentinin işlevlerini yerine getirebilmesi için gerekli olan gereksinimler detaylandırılmıştır. Gereksinimler, komponentin temel yapı taşlarını oluşturan **base class** tanımına dayanarak belirlenmiştir.

1. BrakeAndDirectionManager, aracın **fren ve yön yönetimini sağlayan** temel bir bileşendir. Bu bileşen, gelen **fren ve yön bilgilerini işler**, gerekli güncellemeleri yapar ve **IBrakeStatus** ile **IDirectionStatus** arayüzleri aracılığıyla sistemin diğer bileşenlerine bilgi sağlar.

1.1. Komponent, **başlangıç değerleriyle başlatılmalı** ve çalışma sürecinde bu değerleri yönetmelidir.
1.2. Tüm **getter/setter fonksiyonları** ve **arayüz bağlantıları** eksiksiz çalışmalıdır.
1.3. Güncellenen yön ve fren bilgileri **ilgili arayüzlere düzenli olarak yazılmalıdır**.
1.4. **Hata yönetimi mekanizması** aktif olmalıdır.

2. BrakeAndDirectionManager, sistem başlatıldığında **varsayılan olarak ileri yön (DIRECTION_FORWARD) ve fren devre dışı (BRAKE_DISABLE) durumunda başlatılmalıdır**.
   2.1. Başlangıç değerleri aşağıdaki gibi olmalıdır:
      2.1.1. direction = DIRECTION_FORWARD
      2.1.2. brake = BRAKE_DISABLE
   2.2. Yeni değerler validasyon kurallarına göre güncellenmelidir.
   2.3. Başlangıçta, bu değerler **IBrakeStatus** ve **IDirectionStatus** arayüzlerine yazılmalıdır.

3. BrakeAndDirectionManager, **MotorDriverControlManager** bileşeninden **IBrakeAndDirection** arayüzü üzerinden **fren ve yön durumlarını belirli periyotlarla okuma** işlemi gerçekleştirmelidir.
   3.1. **readBrake()** fonksiyonu çağrılarak **fren durumu**
   3.2. **readDirection()** fonksiyonu çağrılarak **yön durumu** okunmalıdır.
   3.3. **Periyodik bir döngü içinde** bu kontroller yapılmalı ve **değişiklikler tespit edilmelidir**.
   3.4. Eğer bir değişiklik tespit edilirse, ilgili setter fonksiyonları kullanılarak güncelleme yapılmalıdır.

4. MotorDriverControlManager, fren durumu **BRAKE_UNKNOWN** olarak raporladığında, bu durum bir hata olarak değerlendirilmelidir.
   4.1. Eğer **readBrake()** fonksiyonu **BRAKE_UNKNOWN** döndürürse:
      4.1.1. Hata durumu **retVal** değişkenine kaydedilmelidir.
      4.1.2. **Error handling mekanizması devreye girmeli** ve hata kodu saklanmalıdır.
      4.1.3. **Sistemsel hata bildirimi** sağlanmalıdır.

5. MotorDriverControlManager, yön durumu **DIRECTION_UNKNOWN** olarak raporladığında, bu durum bir hata olarak değerlendirilmelidir.
   5.1. Eğer **readDirection()** fonksiyonu **DIRECTION_UNKNOWN** döndürürse:
      5.1.1. Hata durumu **retVal** değişkenine kaydedilmelidir.
      5.1.2. **Sistemsel hata bildirimi** sağlanmalıdır.

6. Aracın yön değişikliği **yalnızca fren aktifken (BRAKE_ENABLE) yapılabilir**.
   6.1. Eğer **brake == BRAKE_DISABLE** ise, yön değişikliğine **izin verilmemelidir**.
   6.2. Eğer **brake == BRAKE_ENABLE** ise, yön değişikliğine izin verilmelidir.
   6.3. Yön değişikliği yapıldığında, yeni değer **IDirectionStatus** arayüzüne yazılmalıdır.

7. MotorDriverControlManager'dan alınan ve güncellenen fren ve yön bilgileri, **IBrakeStatus** ve **IDirectionStatus** arayüzlerine yazılmalıdır.
   7.1. Güncellenen **fren durumu**, **IBrakeStatus** arayüzüne yazılmalıdır.
   7.2. Güncellenen **yön durumu**, **IDirectionStatus** arayüzüne yazılmalıdır.

8. Fren durumundaki herhangi bir değişiklik, **SpeedControlManager_ruBrakeUpdate()** fonksiyonuna bildirilmelidir. Frenleme durumu, hız kontrolünü doğrudan etkileyen bir faktördür. Fren etkinleştirildiğinde veya devre dışı bırakıldığında, hız yönetimi bunu dikkate alarak uygun değişiklikleri yapmalıdır.
    8.1. **SpeedControlManager_ruBrakeUpdate()** çağrılmalıdır.
    8.2. Güncellenmiş hız durumu ilgili bileşenlere iletilmelidir.

4. Arayüz Sınıf Tanımlamaları

Abstract Sınıf	Tanım
Abstract_MotorDriver	Motorun hız, fren ve yön yönetimi için gerekli olan arayüzleri içerir.

1. Arayüz Tanımlamaları

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.
IMotorDriveError	Motor sürücü hatalarının tanımlanması ve raporlanması için kullanılır.

7. Test Senaryoları

Senaryo A: Direction SIL TEST

Test Amacı: Bu testin amacı, yön değiştirme komutlarının doğru şekilde işlendiğini kontrol etmektir. Yön değişikliklerinin her iterasyonda doğru şekilde yapıldığını ve fren durumunun bu değişiklikleri etkilemediğini test ederiz.

Test Senaryosu:

1. Başlangıçta BrakeAndDirectionManager başlatılır.
2. **2. İterasyon:** Yön değiştirme komutu (DIRECTION_BACKWARD) gönderilir. Bu, yönün geri (backward) olarak ayarlandığını simüle eder. Bu komut sürekli gönderilir, yani yön sürekli olarak geri yönüne ayarlanır.
    - Komut: BrakeAndDirectionInterface.writeDirection(DIRECTION_BACKWARD);
3. **7. İterasyon:** Yön değişir ve ileri (DIRECTION_FORWARD) yönü seçilir.
    - Komut: BrakeAndDirectionInterface.writeDirection(DIRECTION_FORWARD);
4. **10. İterasyon:** Yön tekrar geri (backward) olarak değiştirilir.
    - Komut: BrakeAndDirectionInterface.writeDirection(DIRECTION_BACKWARD);
5. **20. İterasyon:** Yön tekrar ileri (forward) olarak değiştirilir.
    - Komut: BrakeAndDirectionInterface.writeDirection(DIRECTION_FORWARD);
6. Yön değişikliklerinin doğru şekilde uygulanıp uygulanmadığı her iterasyonda toString() fonksiyonu ile yazdırılır ve izlenir.

Özet: Bu test, yön değiştirme komutlarının doğru bir şekilde gönderildiğini ve her yön değişikliğinin etkili olduğunu doğrular.

Senaryo B: Brake SIL TEST

Test Amacı: Bu testin amacı, fren komutunun doğru şekilde işlenip işlenmediğini ve fren komutunun sistemin diğer işlevleriyle uyumlu çalışıp çalışmadığını test etmektir.

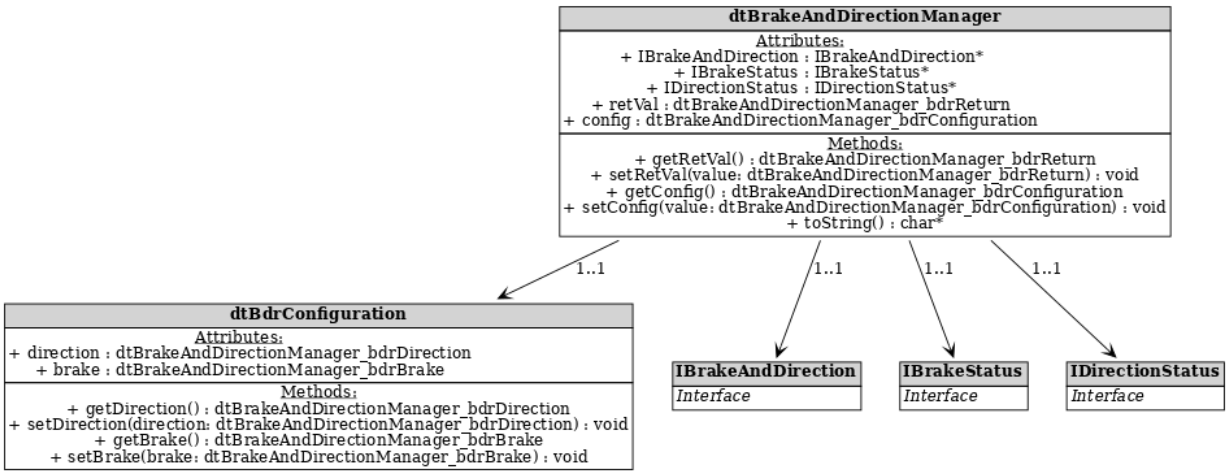
Test Senaryosu:

1. Başlangıçta BrakeAndDirectionManager başlatılır.
2. **5. İterasyon:** Fren aktif edilir. Bu, BRAKE_ENABLE komutuyla yapılır.
   - Komut: BrakeAndDirectionInterface.writeBrake(BRAKE_ENABLE);
3. **15. İterasyon:** Fren devre dışı bırakılır. Bu, BRAKE_DISABLE komutuyla yapılır.
   - Komut: BrakeAndDirectionInterface.writeBrake(BRAKE_DISABLE);
4. Frenin aktif olduğu ve devre dışı bırakıldığı durumlarda, sistemin yön kontrolüyle birlikte nasıl tepki verdiği gözlemlenir.
5. Fren durumlarının doğru şekilde değiştiği her iterasyonda toString() fonksiyonu ile yazdırılır ve izlenir.

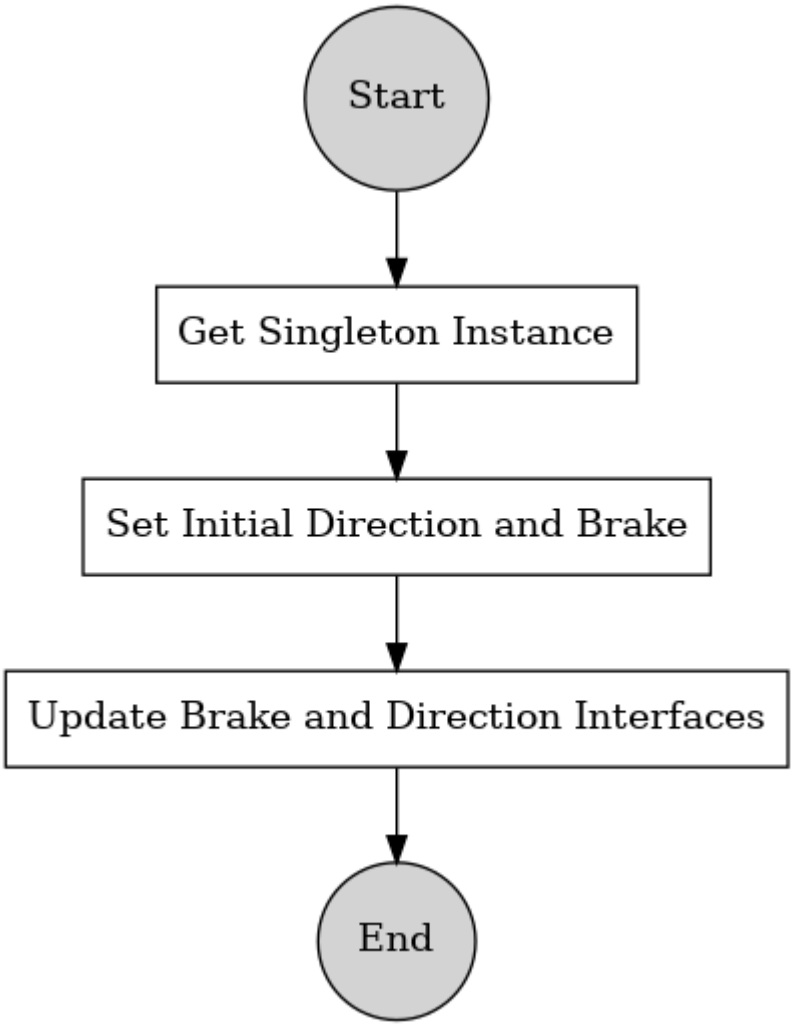
Özet: Bu test, fren komutunun doğru şekilde işlenip işlenmediğini ve frenin doğru bir şekilde etkinleştirilip devre dışı bırakıldığını doğrular

8. Görsel Modeller ve Diyagramlar

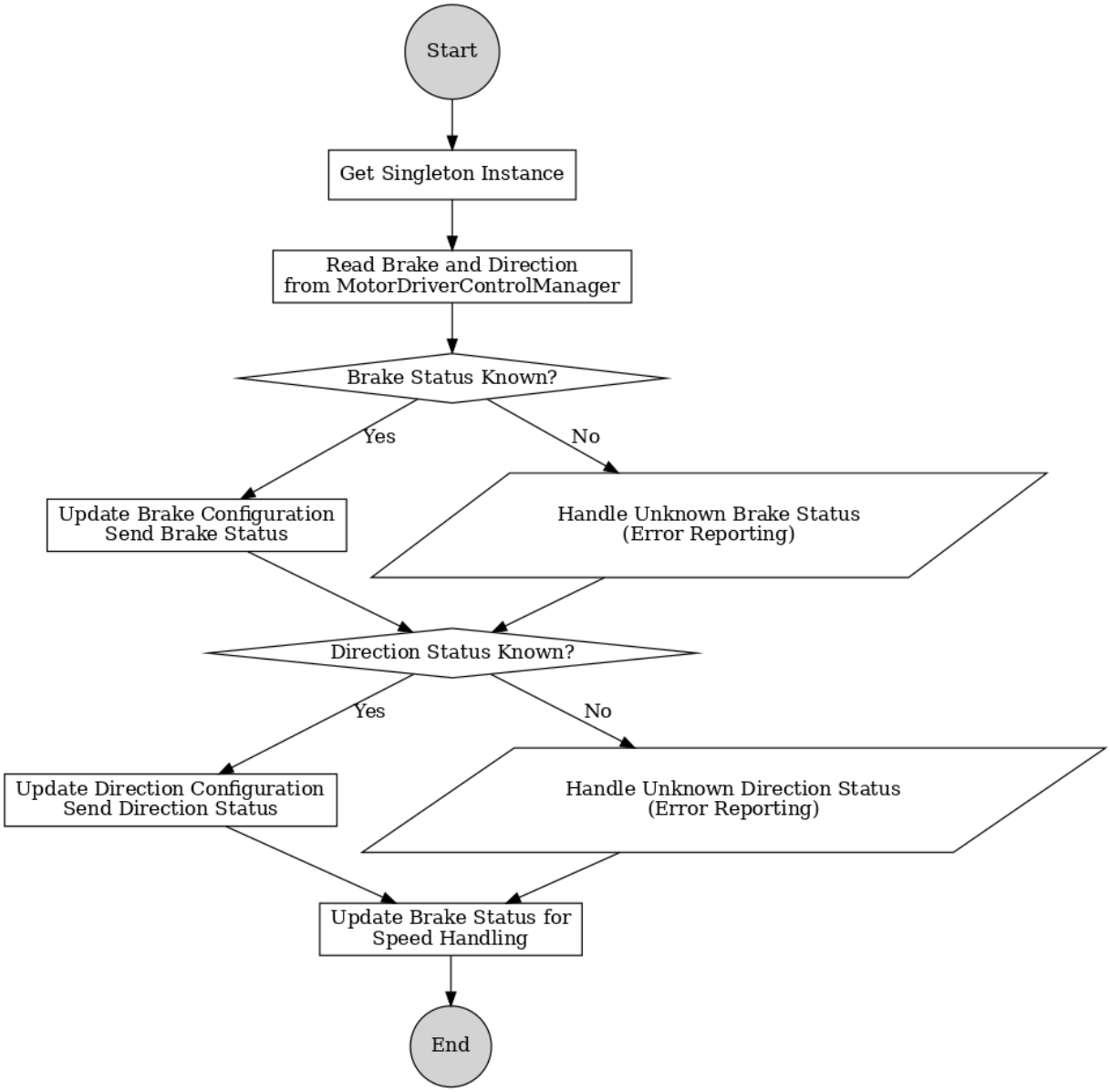
8.1 UML Diagram



8.2 BrakeAndDirectionManager_ruInitialisation Activity Diagram



8.3 BrakeAndDirectionManager_ruRefresh Activity Diagram



11. Ekler