Ege Elektromobil SpeedControlManager Komponent Gereksinim Spesifikasyon (SRS) Dokümanı

Ahmet Vedat Özkılıç

http://www.egeelektromobiltakimi.com/
2025

Revizyonlar

Versiyon	Yazar(lar)	Versiyon Açıklaması	Tamamlanma Tarihi
1.0.0-0	Ahmet Vedat Özkılıç	SpeedControlManager komponentinin gereksinim dokümanı oluşturulmuştur. Temel gereksinimler eklenmiştir.	18 Şubat 2025 18.02.2025

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

İÇİNDEKİLER

1. Giriş

1.1 Giriş

Bu doküman, **SpeedControlManager** 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 Dokümanın kapsamı

Bu doküman, SpeedControlManager'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

SpeedControlManager, aracın hız kontrolünü sağlayan bir yazılım bileşenidir. Bu bileşen, hız durumunu ve göstergesini yöneten bir yapı sunar ve çeşitli arayüzler aracılığıyla hız kontrolüne ilişkin bilgileri işler. Sistem, hız ve fren durumlarını dinamik olarak yönetir, günceller ve **IRawSpeed, ISpeedStatus** ve **IIndicatorSpeedStatus** arayüzleri üzerinden güncel bilgileri yayınlar. Ayrıca, yapılandırma ve durum yönetimi için singleton tasarım deseni benimsenmiştir.

2. Genel Açıklamalar

2.1 Temel Fonksiyon tanımlaması

SpeedControlManager'ın temel fonksiyonları:

- Hız durumu okuma ve yönetme: Hız durumu ve göstergesiyle ilgili verileri değerlendirerek sistemde uygun güncellemeler yapar.
- Hız yapılandırmalarını yönetme: Minimum hız adımı, maksimum hız adımı ve hız bölücüsü gibi yapılandırma değerlerini okur ve günceller.
- Hız ve durum bilgilerini yayınlama: **IRawSpeed, ISpeedStatus** ve **IIndicatorSpeedStatus** arayüzleri üzerinden güncel hız ve durum bilgilerini paylaşır.
- Fren durumu yönetimi: Fren durumunu değerlendirerek uygun hız kontrol güncellemelerini gerçekleştirir.
- Durum ve yapılandırmayı dönüştürme: Hız kontrol durumlarını ve göstergelerini string temsiline dönüştürerek bilgi paylaşımını kolaylaştırır.

2.2 Fonksiyonel Hedefler

SpeedControlManager, aracın hız durumunu ve göstergelerini periyodik olarak okuyup güncelleyerek, hız ve fren değişikliklerini algılar ve uygun aksiyonları gerçekleştirir. Minimum, maksimum ve bölücü hız yapılandırmalarını dinamik olarak yönetir. Hız kontrol mekanizması, fren ve hız durumlarıyla entegre çalışarak sistemin kararlı bir şekilde işlemesini sağlar. Ayrıca, hız göstergeleri için durumları string temsiline dönüştürerek hata senaryolarında uygun hata yönetimi mekanizmalarını destekler. Singleton tasarım deseni sayesinde sistem genelinde merkezi bir kontrol sunar.

3. Fonksiyonel Gereksinimler

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

1. **SpeedControlManager**, aracın hız yönetimini sağlayan temel bir bileşendir. Bu bileşen, hız ve fren bilgilerini işler, gerekli güncellemeleri yapar ve **IRawSpeed**, **ISpeedStatus** ve **IIndicatorSpeedStatus** 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 hız ve durum bilgileri ilgili arayüzlere düzenli olarak yazılmalıdır.
   1.4. Hata yönetimi mekanizması aktif olmalıdır.

2. **SpeedControlManager**, sistem başlatıldığında varsayılan olarak **SpeedStatus** ve **RawSpeed** "0", Indicator ise INDICATORSPEEDSTATUS_IDLE olarak ayarlanmalıdır.
   8.1 Başlangıç değerleri aşağıdaki gibi olmalıdır:
      2.1.1. SpeedStatus = 0
      2.1.2. RawSpeed = 0
      2.1.3. indicator = INDICATORSPEEDSTATUS_IDLE
   8.2 Yeni değerler validasyon kurallarına göre güncellenmelidir.
   8.3 Başlangıçta, bu değerler **IRawSpeed**, **ISpeedStatus** ve **IIndicatorSpeedStatus** arayüzlerine yazılmalıdır.

3. SpeedControlManager, **IRawSpeed** arayüzü üzerinden ham hız değerini periyodik olarak okumalıdır:

   3.1 **readRawSpeed**() fonksiyonu çağrılarak ham hız değeri alınmalıdır.
   3.2 Alınan ham hız değeri, maksimum ham hız değeri olan 4095'e göre normalize edilerek bir hedef hız (**TargetSpeed**) hesaplanmalıdır:
   Normalizasyon formülü:
   TargetSpeed = (**RawSpeed** * **1000) / 4095**
   3.3 Hesaplanan hedef
   hız, SpeedControlManager yapılandırmasındaki **TargetSpeed** değişkenine atanmalıdır.

4. SpeedControlManager, **IBrakeStatus** arayüzü üzerinden fren durumunu periyodik olarak okumalıdır:

   4.1 **readBrake**() fonksiyonu çağrılarak fren durumu alınmalıdır.
   4.2 Eğer fren durumu **BRAKE_STATUS_ENABLE** ise:
   CurrentSpeed değeri 0 olarak ayarlanmalıdır.
   4.3 Hız durumu (**SpeedStatus**) sıfırlanmalı ve **ISpeedStatus** arayüzüne yazılmalıdır.

5. SpeedControlManager, hız kontrolünde fren durumu ve hedef hıza göre dinamik bir yapı sergilemelidir:

    5.1 Eğer fren durumu **BRAKE_STATUS_ENABLE** ise:
    CurrentSpeed sıfırlanmalı ve hız durumu güncellenmelidir.
    5.2 Fren durumu devrede değilse:
    Hedef hıza ulaşmak için hız artırımı veya azaltımı yapılmalıdır.

6. SpeedControlManager, mevcut hız (**CurrentSpeed**) ve hedef hız (**TargetSpeed**) arasındaki farkı hesaplayarak dinamik bir hız adımı (**SpeedStep**) belirlemelidir:

    6.1 Hız adımı, minimum ve maksimum hız adımları arasında olmalıdır:
    6.1.1 Minimum hız adımı: minStep = 5.
    6.1.2 Maksimum hız adımı: maxStep = 50.
    6.1.3 Dinamik bölme faktörü: divider = 20.
    6.2 Dinamik hız adımı hesaplama formülü:
    SpeedStep = min(max(minStep, SpeedDiff / divider), maxStep)
    Burada, SpeedDiff hedef hız ile mevcut hız arasındaki farktır:
    SpeedDiff = abs(TargetSpeed - CurrentSpeed)
    6.3 Eğer CurrentSpeed, TargetSpeed'den küçükse:
    SpeedStep kadar hız artırılmalıdır.
    Hız artırımı sonucu CurrentSpeed, TargetSpeed'i aşmamalıdır.
    6.4 Eğer CurrentSpeed, TargetSpeed'den büyükse:
    SpeedStep kadar hız azaltılmalıdır.
    Hız azaltımı sonucu CurrentSpeed, TargetSpeed'in altına düşmemelidir.
    6.5 Mevcut hız (CurrentSpeed), hız durumu (SpeedStatus) olarak güncellenmelidir.
    Güncellenen hız durumu, ISpeedStatus arayüzüne yazılmalıdır:
    writeSpeedStatus() fonksiyonu çağrılarak hız durumu ilgili arayüze iletilmelidir.

7. SpeedControlManager, sistemdeki hız ayarlarını (minimum adım, maksimum adım ve hız bölücüsü) dinamik olarak yönetme işlemini gerçekleştirmelidir.

    7.1 **getMinStep(), getMaxStep()** ve **getDivider()** getter fonksiyonları çağrılarak hız yapılandırma değerleri okunmalıdır.
    7.2 **setMinStep(), setMaxStep()** ve **setDivider()** setter fonksiyonları çağrılarak hız yapılandırma değerleri güncellenmelidir.
    7.3 Periyodik bir döngü içinde bu yapılandırma değerleri kontrol edilmeli ve değişiklikler tespit edilmelidir.
    7.4 Eğer bir değişiklik tespit edilirse, yapılandırma değişiklikleri validasyon kurallarına göre kontrol edilmeli ve uygun değerler diğer bileşenlere entegre edilmelidir

8. SpeedControlManager, **IIndicatorSpeedStatus** arayüzüne gösterge durumunu yazmalıdır:
    8.1 Gösterge durumu (**Indicator**), hız durumuna göre aşağıdaki gibi ayarlanmalıdır:
        8.1.1. INDICATORSPEEDSTATUS_IDLE: Hız sıfır olduğunda
        8.1.2. INDICATORSPEEDSTATUS_LOWSPEED: Hız düşük seviyedeyken
        8.1.3. INDICATORSPEEDSTATUS_MIDSPEED: Hız orta seviyedeyken
        8.1.4. INDICATORSPEEDSTATUS_HIGHSPEED: Hız yüksek seviyedeyken
    8.2 Gösterge durumu, **writeIndicatorSpeedStatus**() fonksiyonu ile **IIndicatorSpeedStatus** arayüzüne yansımalıdır.

9. SpeedControlManager, **retVal** değişkenini kullanarak sistemin çalışma durumunu ve hata koşullarını yönetir.

    9.1 retVal değeri, SCM_ERR, SCM_EOK veya SCM_UNKNOWN gibi durumları ifade edebilir:
        9.1.1 SCM_ERR: Sistem bir hata ile karşılaşmışsa atanır.
        9.1.2 SCM_EOK: Sistem normal şekilde çalışıyorsa atanır.
        9.1.3 SCM_UNKNOWN: Sistem durumu bilinmiyorsa atanır.

10. Belirtilmeyen veya bilinmeyen durumlar için uygun hata yönetimi uygulanmalıdır.

    10.1 Örneğin: Bellek tahsisi başarısız olursa, uygun bir hata mesajı döndürülmelidir.
    10.2 Hatalı arayüz bağlantıları için sistemsel bir hata raporlama mekanizması sağlanmalıdır.

4. Arayüz Sınıf Tanımlamaları

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

5. Arayüz Tanımlamaları

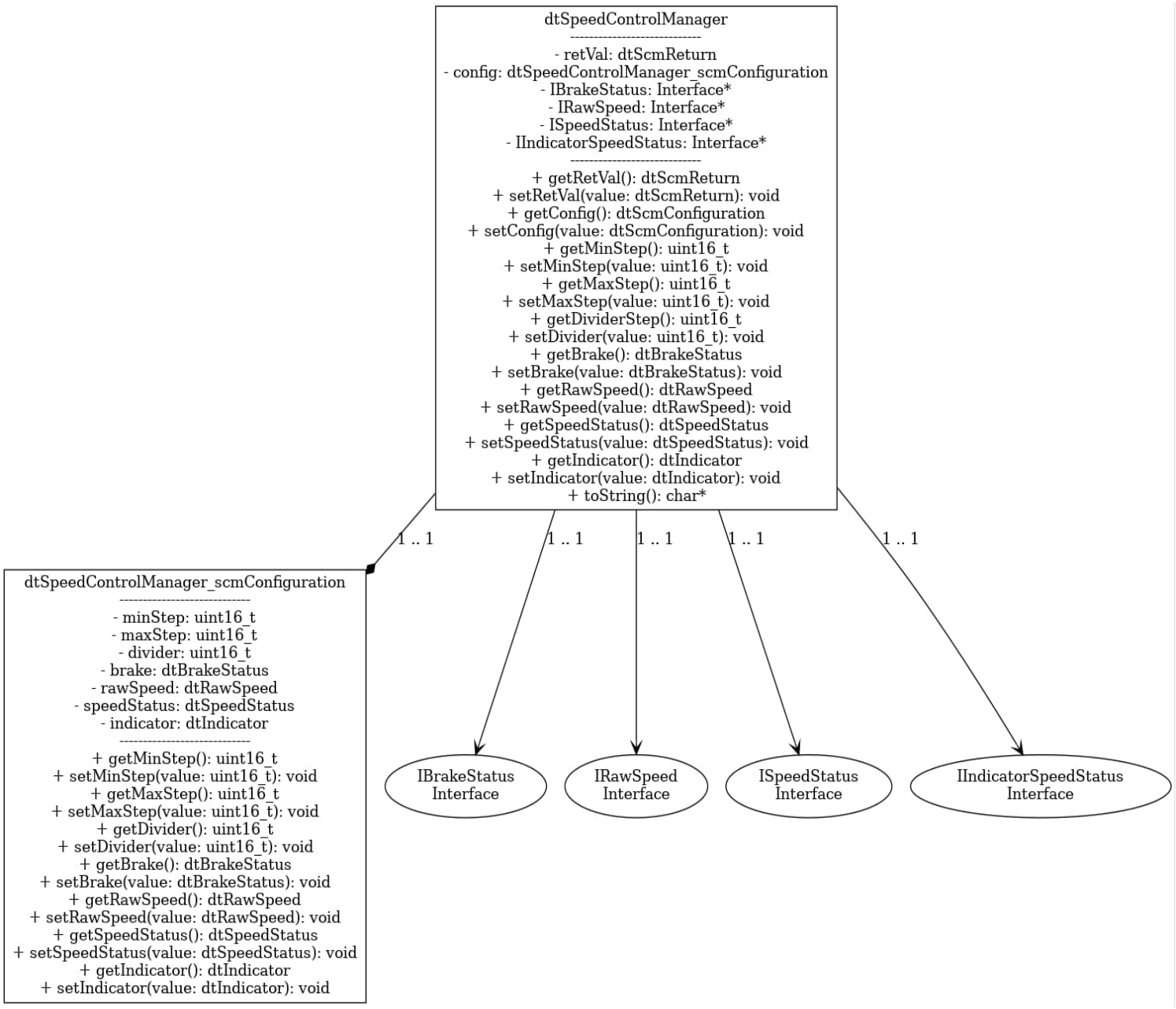
Abstract_MotorDriver

Arayüz	Tanım
IRawSpeed	Ham hız verilerinin okunması ve sistemde işlenmesi için kullanılır.
ISpeedStatus	Hız durumunun okunması ve sistemle paylaşılması işlevini tanımlar.
IIndicatorSpeedStatus	Hız göstergesi durumlarının yönetimi ve diğer bileşenlerle paylaşımı için kullanılır.
IBrakeStatus	Fren durumunun yönetimini tanımlar.

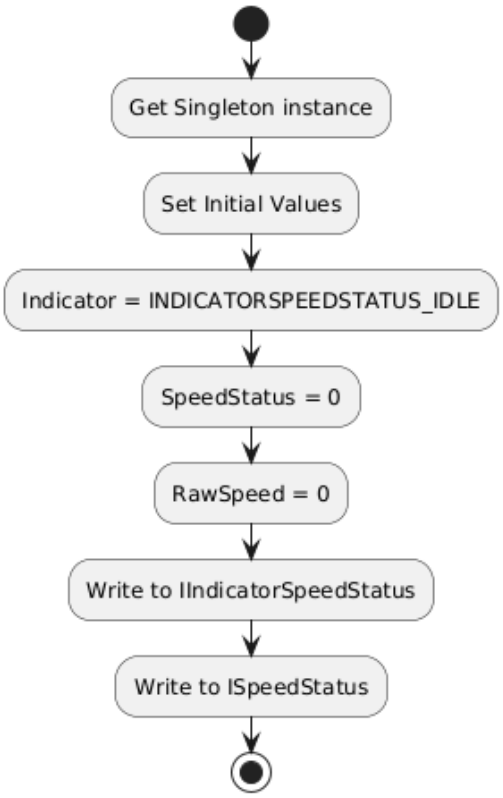
6. Test Senaryoları

7. Görsel Modeller ve Diyagramlar

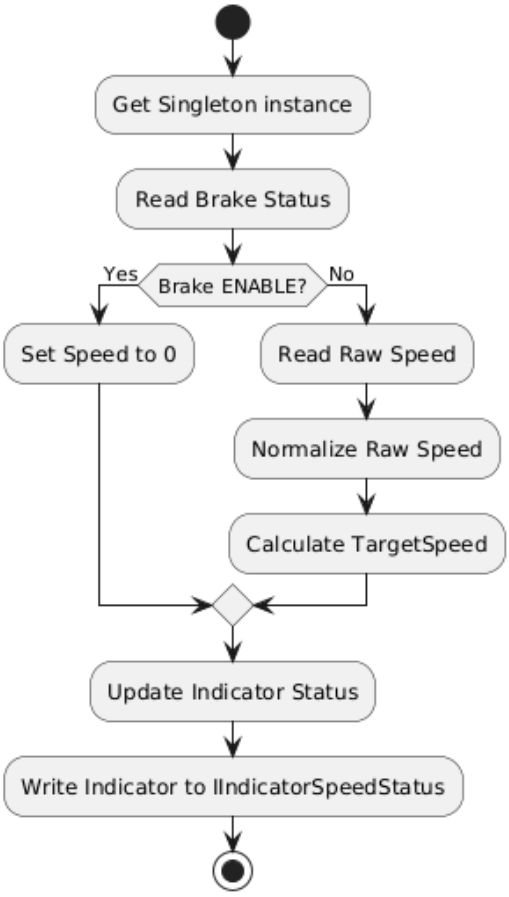
7.1 UML Diagram



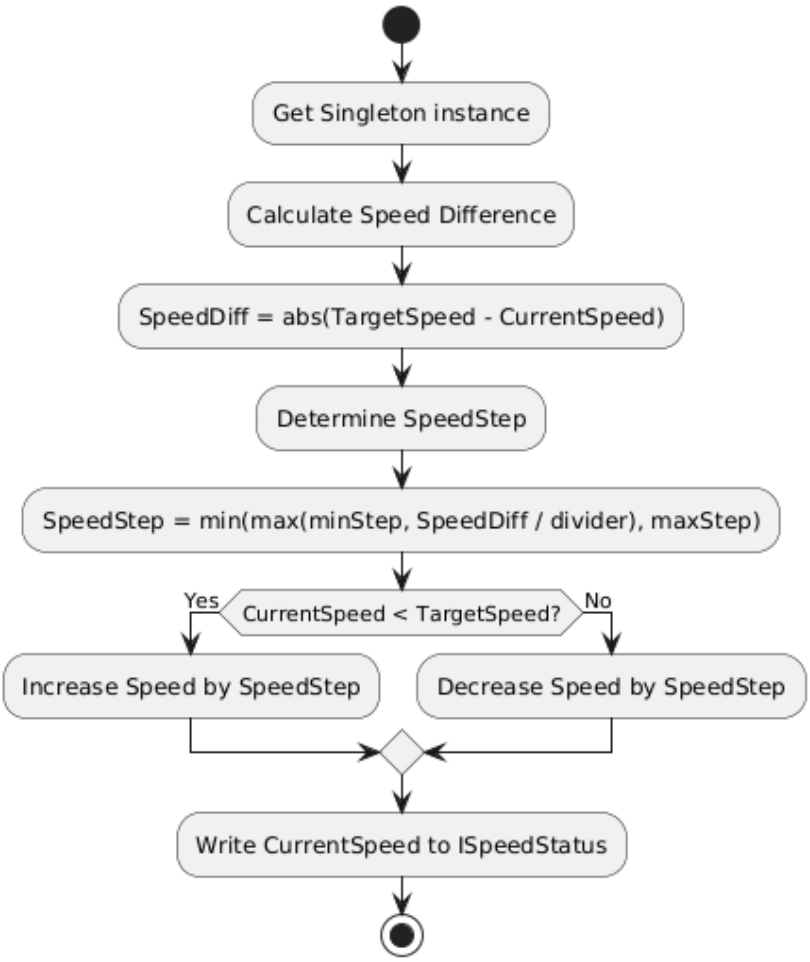
7.2.1 SpeedControlManager_ruInitialisation Activity Diagram



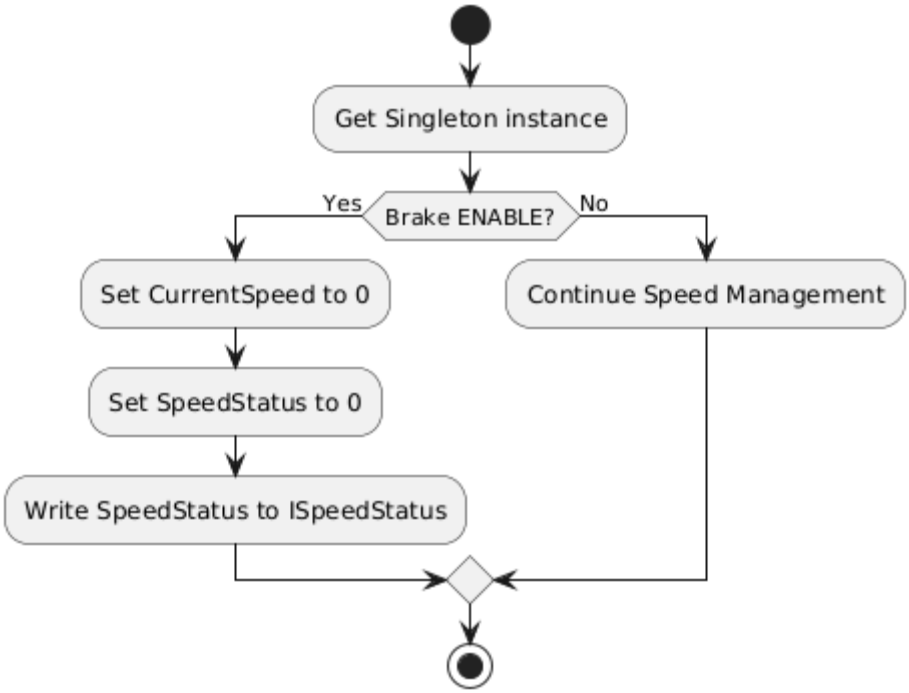
7.2.2 SpeedControlManager_ruRefresh Activity Diagram



7.2.3 SpeedControlManager_ruUpdate Activity Diagram



7.2.4 SpeedControlManager_ruBrakeUpdate Activity Diagram



8. Güncelleme Kayıt Defteri

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

9. Güncellenen Bütçe Kaydı

Güncellenen versiyon bütçe gerektiren bir güncelleme ise burada kayıt tutulmaktadır.

10. Ekler