Bu projede yapılacak işlemler aşağıda listelenmiştir

Proje Adı:

Dağıtık Sonar Güvenlik Sistem Tasarımı:

Proje Tanımı ve Çalışma Prensibi:

Bu projede ultrasonik mesafe sönsörü ve servo motor yardımıyla belirlenen uzaklıktaki alanı tarayan bir sonar sistemi geliştirilecek ve bu tarama sonucunda belirlenen mesafede herhangi bir nesne varsa ses ve ışık ile uyarı verecektir. Projede 3 adet mcu kullanılması amaçlanmaktadır. Herbir mcu’nun kendine ait görevleri bulunmaktadır. Bu mikrodenetleyicilerden 1.’si mesafe sensörü ile ölçüm yapacak ikinci mikrodenetleyici servo motorun çalışmasını kontrol ederek belirli bir açıda dönmesini sağlayacak 3. Mikrodenetleyici ise gelen ölçümleri ekranda gösterecek ve ses ve ışık uyarılarını verecektir.

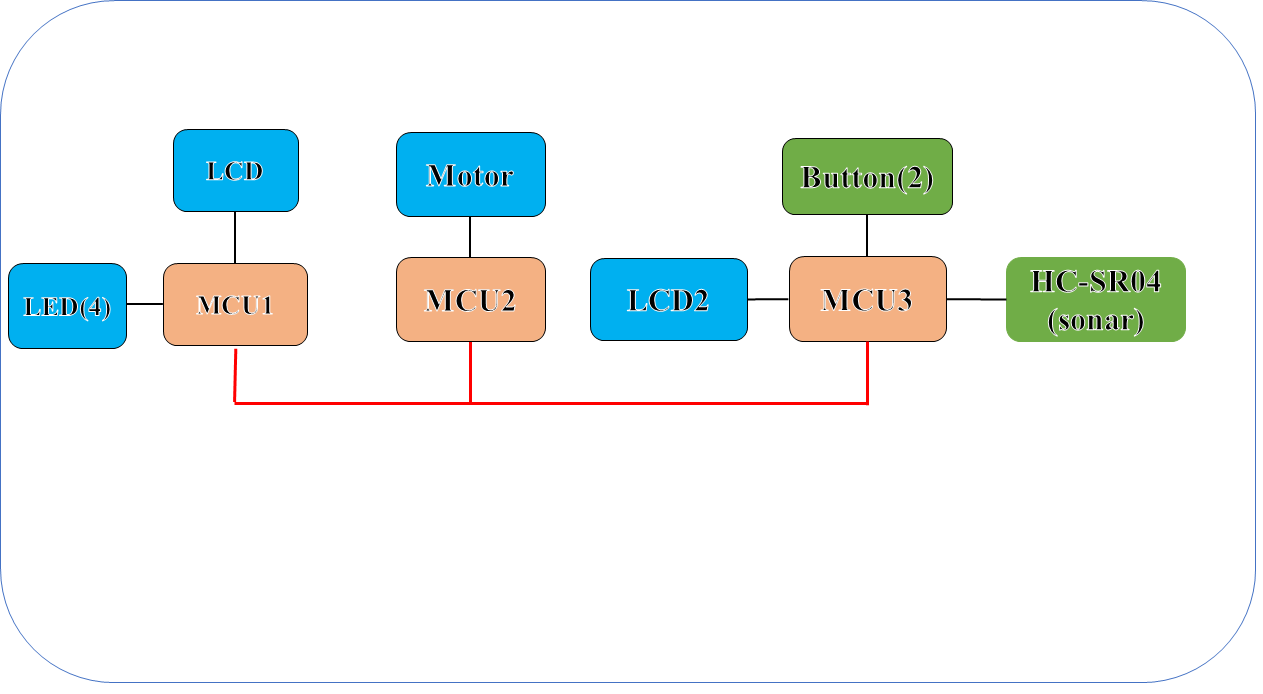
Haberleşme sistemi can bus üzerinden yapılacaktır.

Projede Kullanılacak Cihazlar ve Planlanan Kurgu:

* 3 adet MCU
* HC-SR04
* 2x16 LCD Ekran
* Servo Motor
* Ledx4
* Buttonx2

Proje Blok Diyagramı:

Blok diyagram aşağıdaki şekilde gösterildiği gibidir.



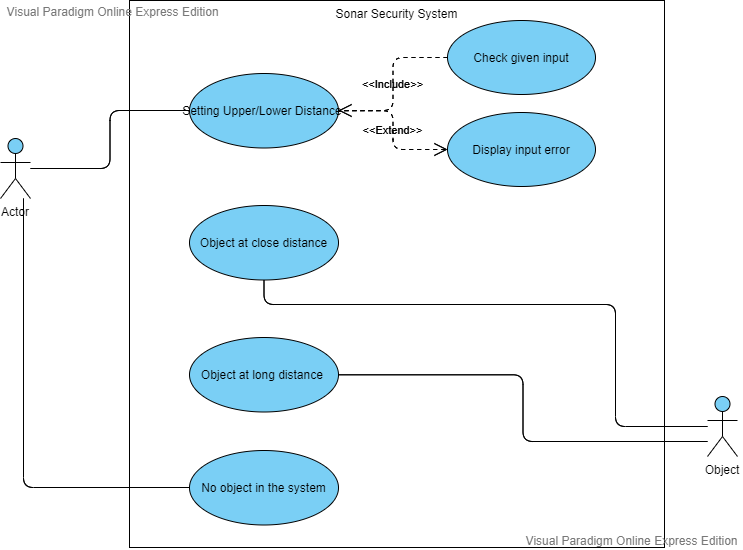
CAN BUS

System model

Proje Zamanı

Tabloda gösterildiği aralıklarla işlemler gerçekleştirilecektir

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Elements** | **26 Ekim-1**  **Kasım** | **2 -8**  **Kasım** | **9 -15**  **Kasım** | **16-22**  **Kasım** | **23-29**  **Kasım** | **30 Kasım-6 Aralık** | **7-13**  **Aralık** | **14-20**  **Aralık** | **21-27**  **Aralık** | **28 Aralık-3 Ocak** | **4-10**  **Ocak** |
| Requirements Elicitaiton | x | x |  |  |  |  |  |  |  |  |  |
| Architecture Creation |  |  | x | x |  |  |  |  |  |  |  |
| Scenarios/Use Cases Creation |  |  |  |  | x | x |  |  |  |  |  |
| Squence Diagram Preparation |  |  |  |  |  |  | x | x |  |  |  |
| State Chart |  |  |  |  |  |  |  |  | x |  |  |
| Implementation |  |  |  |  |  |  |  |  |  | x | x |



Şekil 1. Use Cases

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Text Requirements** | | | | | | |
| **Use Cases** | R1 | R2 | R3 | R4 | R5 | R6 | R7 |
| U1.  Kullanıcı alt ve üst sınır limitini buton ile belirler | X | X |  |  |  | X |  |
| U2.  Alt sınıra yakın bir mesafede nesne vardır |  |  | X |  | X | X |  |
| U3.  Üst sınıra yakın bir mesafede nesne vardır |  |  | X | X |  |  |  |
| U4.  Hiçbir nesne yoktur |  |  | X |  |  |  | X |

USE CASES

U1. Kullanıcı, buton ile alt, üst sınır güvenlik mesafesini belirler

U2. Alt sınıra yakın bir mesafede nesne vardır

U3. Üst sınıra yakın bir mesafede nesne vardır

U4. Hiçbir nesne yoktur

SYSTEM-LEVEL TEXT REQUIREMENTS

R1. Butona basma ile üst sınır güvenlik mesafesi cm cinsinden tanımlanacak

R2. Butona basma ile alt sınır güvenlik mesafesi cm cinsinden belirlenecek

R3. Sensor mesafe okuma işlemini %95 hassaslıkta yapacak

R4. Üst sınır için tanımlanan mesafede herhangi bir cisim var ise mavi led ile uyarı verecek ve motorun dönüş hızı iki katına çıkacak.

R5. Alt sınır için tanımlanan mesafede herhangi bir cisim var ise kırmızı led ile uyarı verecek ve motor duracak.

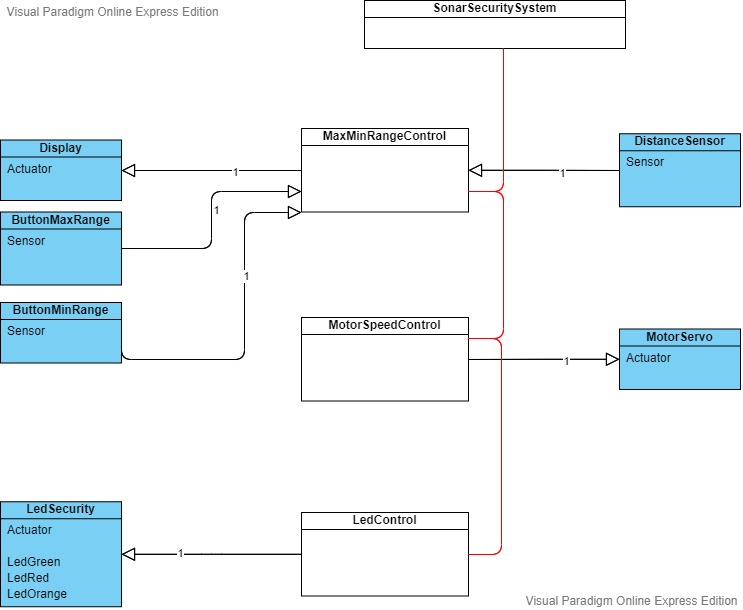
R6. Kullanıcı alt ve üst sınır güvenlik mesafesini sıfır olarak girdiğinde turuncu led uyarı verecek

R7. Ortamda hiçbir nesne yoksa yeşil led ile ortam güvenli uyarısı verilecek.

**Requirements must address off-nominal behaviors that are not apparent in use cases**

U1- Maksimum tarama uzaklığı için sıfır girilmesi

U2- Minimum tarama uzaklığı için sıfır girilmesi



Şekil 2. Class Diagram

**System Sensors**

* ButtonMaxRange[s](v) : Maksimum aralığı seçer.
* V = { counter =1..50} , s =1
* Maksimum güvenlik mesafesi sınırlarını belirlemede kullanılır.
* Varsayılan değeri 0 dır.
* Kullanıcı buton ile maksimum güvenlik sınırını 1 ile 50 cm, arasında ayarlayabilmelidir.
* Buton basıldığında ButtonMaxRange[s](counter) maximum range için counter değeri bir artmalıdır
* ButtonMinRange[s](v) : Minumum güvenlik aralığını seçer.
* V = { counter =1..20} , s =1
* Minumum güvenlik mesafesi sınırlarını belirlemede kullanılır.
* Varsayılan değeri 0 dır.
* Kullanıcı buton ile minumum güvenlik sınırını 1 ile 20 cm, arasında ayarlayabilmelidir.
* Butona basıldığında ButtonMinRange[s](counter) minumum range için counter değeri bir artmalıdır
* DistanceSensor[s](v) : Mesafe ölçümünü gerçekleştiren sensördür.
* V ={range = 0.. 300 cm } s = 1
* Varsayılan değeri 0 dır.
* Ölçüm hassasiyetini %95 oranında gerçekleştirebilmektedir.

**System Actuators**

* Display[s](v) : Mesafe ölçüm sonucunu, motor dönme hızını göstermektedir.
* V ={Measurement, MotorSpeed,LedStatus}
* S =1..2
* Motor dönme hızını, mesafe ölçüm sonucunu ve max-min range ler için tanımlanmış mesafeyi gösterebilmektedir.
* İki adet 2x16 lcd ekran mevcuttur bunlar mcu1 ve mcu3 üzerinde yer almalıdır.
* MotorServo[s](v) : Ultrasonik sensörü belirli bir açıda döndürmede sorumludur.
* V = {True, False}
* S = 1
* Servo motor pwm sinyali ile kontrol edilebilmektedir.
* 180 derece tarama alanına sahiptir.
* LedSecurity[s](c)(v) : Güvenlik mesafesinin aşıldığı farklı renkteki ledler ile ifade edilmektedir.

* V= {True, False} s = 1
* C = {red,blue,orange,green}
* Başlangıçta tüm led durumları False değerini alır.
* Kırmızı led minumum range için alarm işareti anlamına gelmektedir.
* Mavi led maximum range için alarm durumunu ifade etmektedir
* Turuncu led maximum- minumum range için sıfır girildiğini ifade etmektedir.
* Yeşil led ortam güvenli herhangi bir nesne yok durumunu ifade etmektedir.

**Software Control Objects**

* MaxMinRangeControl[s]
* Güvenlik mesafesindeki alanı ifade control eder.
* Maximum ve minimum giriş parametrelerini almaktan sorumludur
* Ölçüm işlemini gerçekleştirir.
* Kullanıcıdan giriş verileri aldıktan sonra led control birimi ile haberleşir.
* MotorSpeedControl[s]
* Servo motorun hızını control eden birimdir.
* PWM sinyali ile servo dönüş açısını ve hızını ayarlar
* Başlangıçta mSpeedMotor[s] =1 değerini alır.
* LedControl biriminden gelen parametreler neticesinde motorun dönme hızını arttrılır veya durdurur.
* mSpeedMotor[s](integer) ile gelen değer ile servo motorun dönüş hızını kontrol eder.
* LedRed=True değerini aldığında MotorSpeedControl birimi motorun çalışma hızını keser.
* LedBlue = True değerini aldığında MotorSpeedControl birimi moturun dönme hızını iki katına çıkarır.
* LedOrange = True değerini aldığında MotorSpeedControl birimi moturu durdurur.
* LedControl[s]

* Kırmızı ,yeşil,mavi ve turuncu ledlerin olay döngüsünde yanmasından sorumludur.
* Kırmızı led: MaxMinRange[s] kontrol biriminden gelecek ölçümde minimum güvenlik sınırında bir cismin olduğunu haber vermektedir.
* Mavi led : MaxMinRange[s] kontrol biriminden gelecek ölçümde maximum güvenlik sınırında bir nesnenin olduğunu haber vermektedir.
* Turunucu led: MaxMinRange[s] kontrol biriminden alınan giriş parametrelerinde güvenlik sınırları için kullanıcının yanlış bir giriş yaptığını göstermektedir. Örneğin maximum veya minumum güvenlik mesafesi için kullanıcı tarafından sıfır girildiğini veya maximum güvenlik mesafesi için daha küçük değer girildiğini belirtmektedir.
* Yeşil led :MaxMinRange[s] kontrol biriminden gelecek ölçümde herhangi bir nesnenin olmadığını ortamın güvenli olduğunu belirtmektedir.

**Controller Messages**

|  |  |  |  |
| --- | --- | --- | --- |
| Message Dictionary | | | |
| MessageID | Sender | Receiver | Definition |
| mMeasurement | MaxMinRangeControl | LedControl | mMeasurement = {0..200}(integer) Ölçüm değerini göstermektedir. |
| mMaxButton | MaxRangeButton | MaxMinRangeControl | mMaxButton= {True, False} Kullanıcının her butona basışında True değerini gönderir. |
| mMinButton | MinRangeButton | MaxMinRangeControl | mMinButton= {True, False} Kullanıcının her butona basışında True değerini gönderir |
| mMaxRange | MaxMinRangeControl | LedControl | mMaxRange = {0..50}(integer) Kullanıcının her butona basışında counter değerinin aldığı değerdir. |
| mMinRange | MaxMinRangeControl | LedControl | mMinRange = {0..20}(integer) Kullanıcının her butona basışında counter değerinin aldığı değerdir. |
| mLedBlue | MaxMinRangeControl | LedControl | mLedBlue = {True, False} mMaxRange in gönderdiği mesajda tanımlanan mesafede nesne varsa True değerini gönderir. |
| mLedRed | LedControl | LedSecurity | mLedRed = {True, False} mMinRange in gönderdiği mesajda tanımlanan mesafede nesne varsa True değerini gönderir. |
| mLedOrange | LedControl | LedSecurity | mLedOrange = {True, False} mMinRange veya mMaxRange mesajlarından birisinin değeri sıfır ise True değerini gönderir. |
| mLedGreen | LedControl | LedSecurity | mLedGreen= {True, False} mMinRange veya mMaxRange aralığının içerisinde herhangibir nesne olmadığını ifade etmektedir. |
| mMotorSpeed | LedControl | MotorSpeedControl | mMotorSpeed ={0..10}(integer) Servo motorun dönme hızının aldığı değerdir. |

Software Requirements

• Structured representation of control objects

• Scenarios

• UML Sequence Diagrams

• A stylized detailed requirements template

Use Case 1: Kullanıcı, buton ile alt, üst sınır güvenlik mesafesini ve belirler

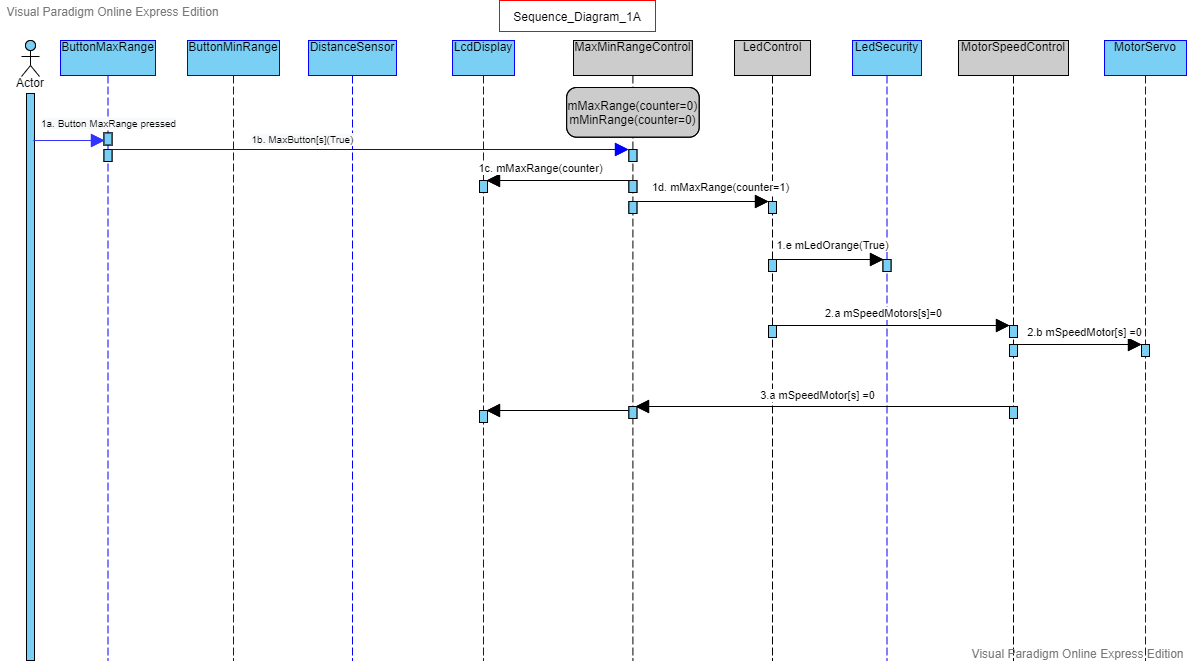
* **Scenario 1A: Kullanıcı alt ve üst güvenlik mesafesini kullanıcı butonu ile sıfır girer.**

Ön Koşullar:

* Kullanıcı sistem parametrelerini belirlerken ortamda hiçbir nesnenin olmadığı varsaymıştır.

Scenario :

1. Kullanıcı güvenlik için gerekli parametreleri sisteme buton aracılığıyla girer.
2. Kullanıcı alt ve üst güvenlik sınırları için sisteme sıfır cm uzaklığını girmiştir.
3. Sistem kullanıcının yanlış parametre girdiğini uyarmak zorundadır.
4. MaxMinRangeControl birimi LedControl birimine mMaxRange ve mMinRange mesajlarını iletir.
5. LedControl birimi LedSecuritye mLedOrange = True değerini gönderir.
6. LedControl birimi mSpeedMotor mesajı ile MotorSpeedControl birimine ileterek servo moturun dönmesini durdurur.
7. MotorSpeedControl birimi mSpeedMotor=0 mesajını MaxMinRangeControl birimine iletip motor hızını ekranda gösterir.
8. Kullanıcının sıfırdan farklı değer girmesi sağlanır.



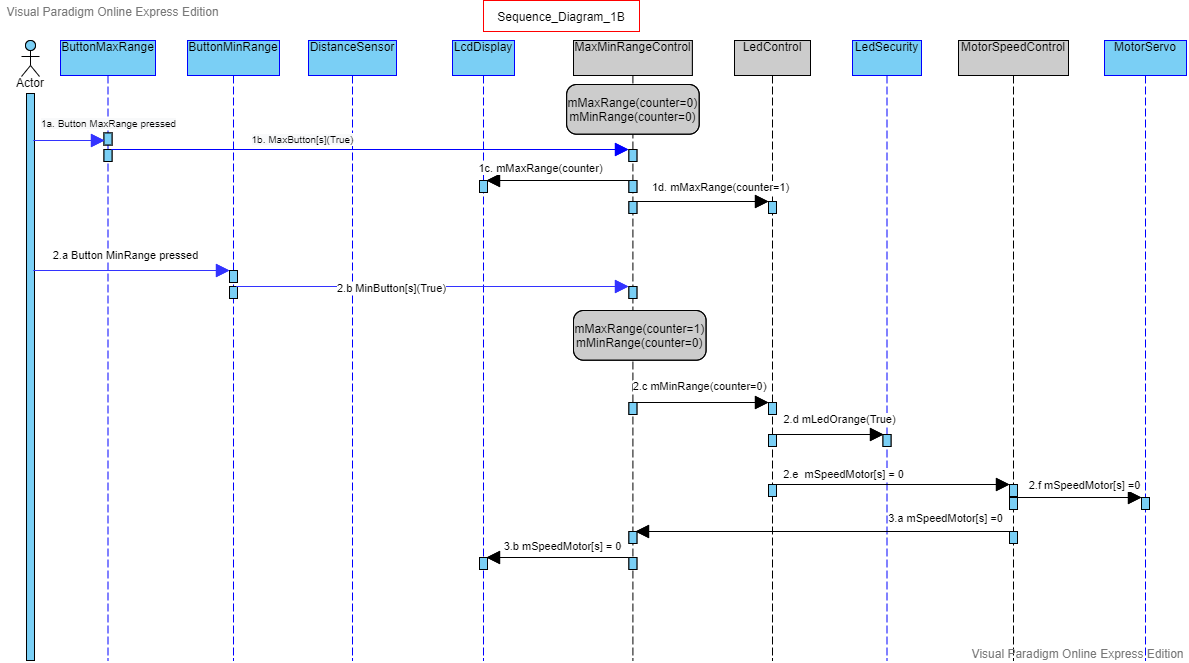
* **Scenario 1B: Kullanıcı üst güvenlik mesafesini sıfırdan farklı bir değer girip alt güvenlik mesafesine sıfır girmesi durumu.**

Ön Koşullar:

* Kullanıcı sistem parametrelerini belirlerken ortamda hiçbir nesnenin olmadığı varsaymıştır.

Scenario :

1. Kullanıcı güvenlik için gerekli parametreleri sisteme buton aracılığıyla girer.
2. mMaxRange =1 değerini girmiş fakat mMinRange = 0 değerini girmiştir.
3. MaxMinRangeControl birimi LedControl birimine mMaxRange ve mMinRange mesajlarını iletir.
4. LedControl birimi LedSecuritye mLedOrange = True değerini gönderir.
5. LedControl birimi mSpeedMotor=0 mesajı ile MotorSpeedControl birimine ileterek servo moturun dönmesini durdurur.
6. MotorSpeedControl birimi mSpeedMotor=0 mesajını MaxMinRangeControl birimine iletip motor hızını ekranda gösterir.
7. Kullanıcının sıfırdan farklı değer girmesi sağlanır.



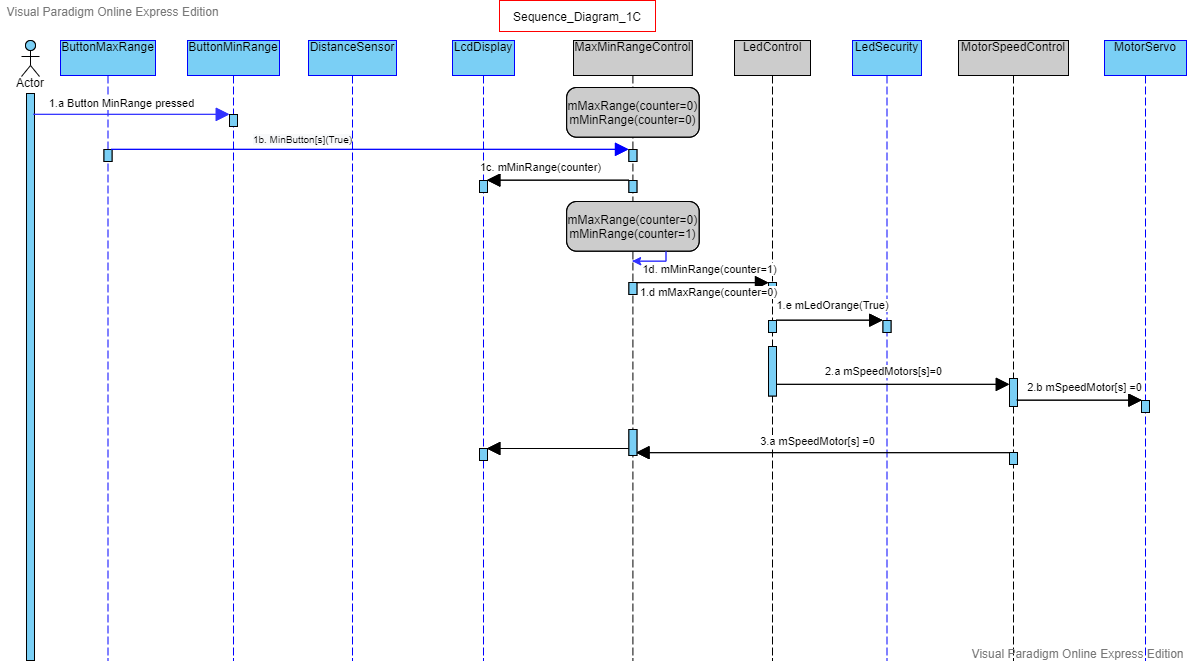
* **Scenario 1C: Kullanıcı sadece alt güvenlik mesafesini kullanıcı butonu ile sıfırdan farklı bir değer girer, üst güvenlik mesafesini sıfır girer.**

Ön Koşullar:

* Kullanıcı sistem parametrelerini belirlerken ortamda hiçbir nesnenin olmadığı varsaymıştır.

Scenario :

1. Kullanıcı güvenlik için gerekli parametreleri sisteme buton aracılığıyla girer.
2. mMinRange =1 değerini girmiş fakat mMaxRange değerini girmemiştir
3. mMinRange ve mMaxRange sayaçları başlangıçta sıfır değerini almaktadır.
4. MaxMinRangeControl birimi LedControl birimine mMaxRange ve mMinRange mesajlarını iletir.
5. LedControl birimi LedSecuritye mLedOrange = True değerini gönderir.
6. LedControl birimi mSpeedMotor=0 mesajı ile MotorSpeedControl birimine ileterek servo moturun dönmesini durdurur.
7. MotorSpeedControl birimi mSpeedMotor=0 mesajını MaxMinRangeControl birimine iletip motor hızını ekranda gösterir.
8. Kullanıcının sıfırdan farklı değer girmesi sağlanır.



Use Case 2: Alt sınıra yakın bir mesafede nesne vardır

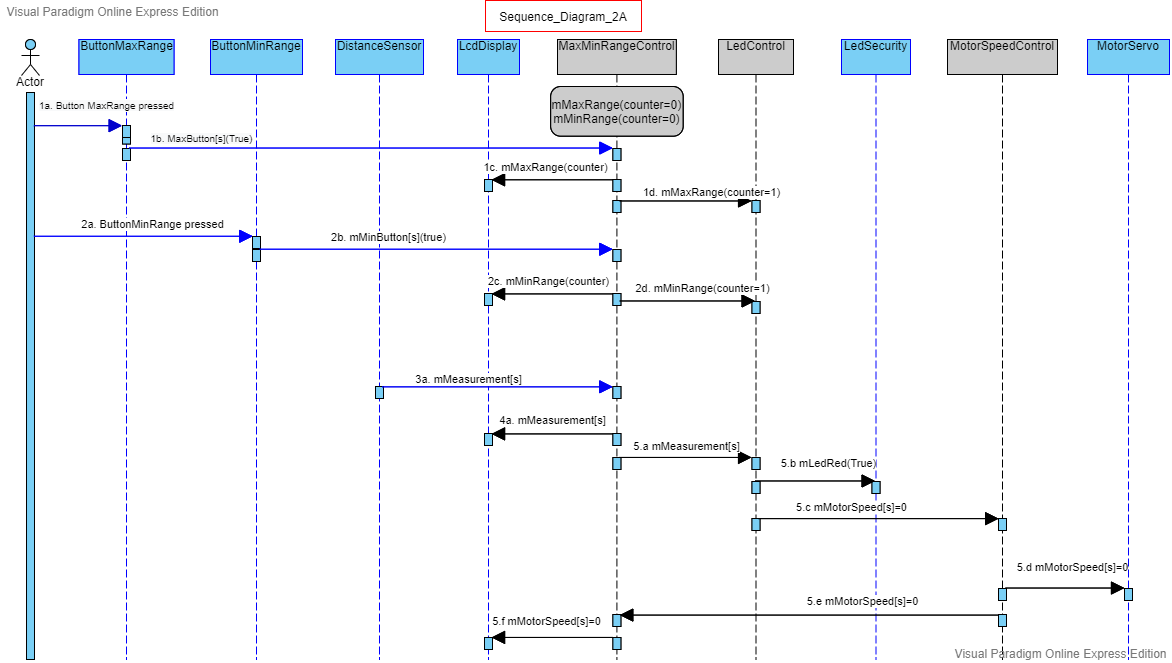
* **Scenario 2A: Kullanıcı alt ve üst güvenlik değerlerini doğru girdikten sonra alt sınıra yakın bir mesafede nesne olması durumu**

Ön Koşullar:

* Kullanıcı sistem parametrelerini doğru olarak girmiştir.

Scenario :

1. Kullanıcı güvenlik için gerekli parametreleri sisteme buton aracılığıyla girer.
2. Kullanıcının alt sınır için girdiği mesafede herhangi bir nesne vardır.
3. MaxMinRangeControl birimi ölçüm mesajını mMinRange ve mMaxRange değerlerini LedControl birimine iletir.
4. mMeasurement ile ölçüm alma işlemi MaxMinRangeControl tarafından başlatılır.
5. MaxMinRangeControl mMeasurement değerini LedControl birimine iletir.
6. LedControl birimi LedSecuritye mLedRed = True değerini gönderir.
7. LedControl birimi MotorSpeedControl birimine mMotorSpeed[s] mesajını gönderir
8. MotorSpeedControl birimi MotorServo birimine mMotorSpeed[s](0) mesajını göndererek motorun dönmesini durdurur.
9. MotorSpeedControl birimi hız değerini lcd ekranda göstermek için MaxMinRangeControl birimine mMotorSpeed değerini gönderir.



Use Case 3: Üst sınıra yakın bir mesafede nesne vardır

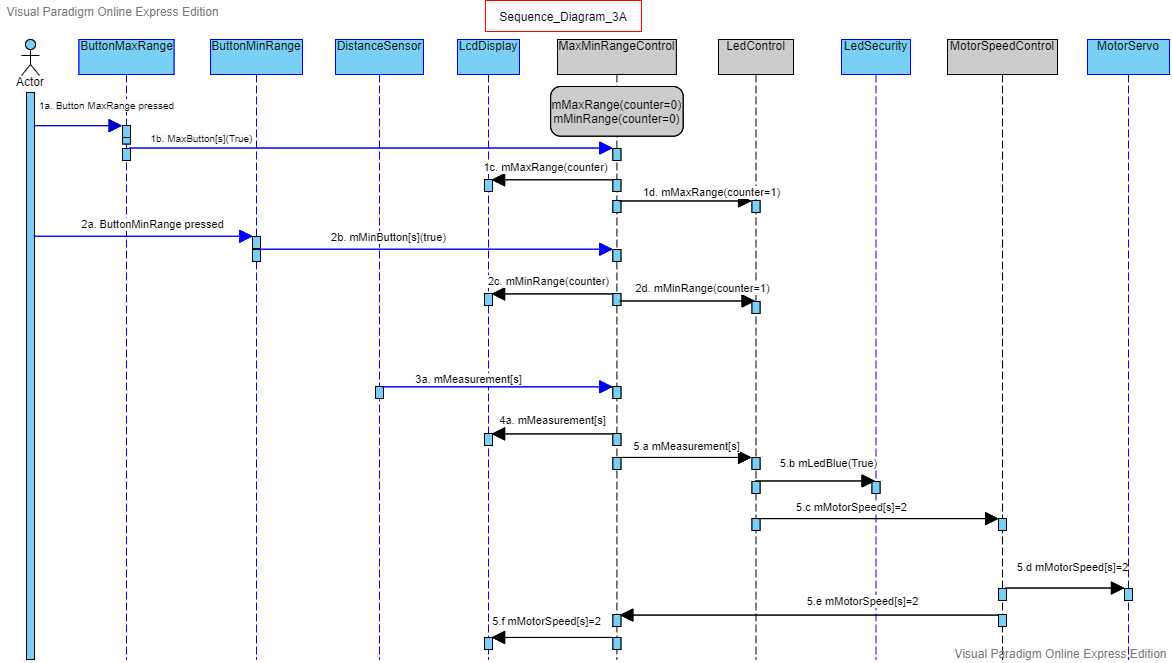
* **Scenario 3A: Kullanıcı alt ve üst güvenlik değerlerini doğru girdikten sonra üst sınıra yakın bir mesafede nesne olması durumu**

Ön Koşullar:

* Kullanıcı sistem parametrelerini doğru olarak girmiştir.

Scenario :

1. Kullanıcı güvenlik için gerekli parametreleri sisteme buton aracılığıyla girer.
2. Kullanıcının üst sınır için girdiği mesafede herhangi bir nesne vardır.
3. MaxMinRangeControl birimi ölçüm mesajını ve mMinRange ve mMaxRange değerlerini LedControl birimine iletir.
4. mMeasurement ile ölçüm alma işlemi MaxMinRangeControl tarafından başlatılır.
5. MaxMinRangeControl mMeasurement değerini LedControl birimine iletir.
6. LedControl birimi LedSecuritye mLedBlue = True değerini gönderir.
7. LedControl birimi MotorSpeedControl birimine mMotorSpeed[s]=2 değerini göndererek daha hızlı dönmesini sağlar.
8. MotorSpeedControl birimi hız değerini lcd ekranda göstermek için MaxMinRangeControl birimine mMotorSpeed değerini gönderir.



Use Case 4: Ortamda hiçbir nesne yoktur.

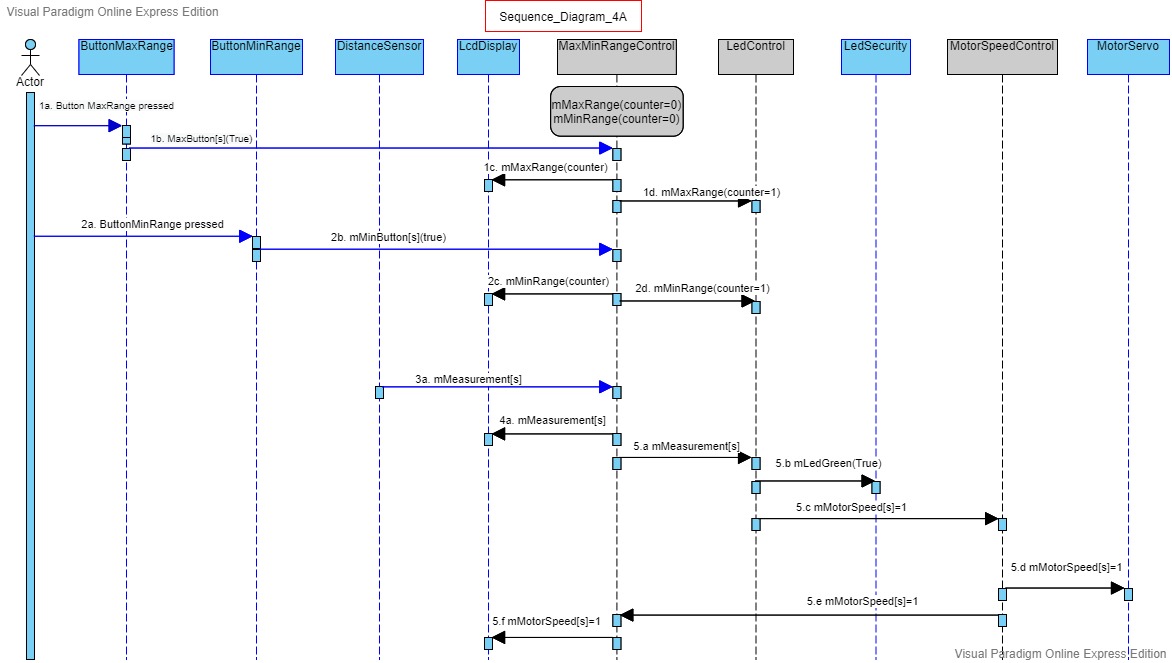
* **Scenario 4A: Kullanıcı alt ve üst güvenlik değerlerini doğru girdikten sonra ortamda hiçbir nesne olmaması durumu**

Ön Koşullar:

* Kullanıcı sistem parametrelerini doğru olarak girmiştir.

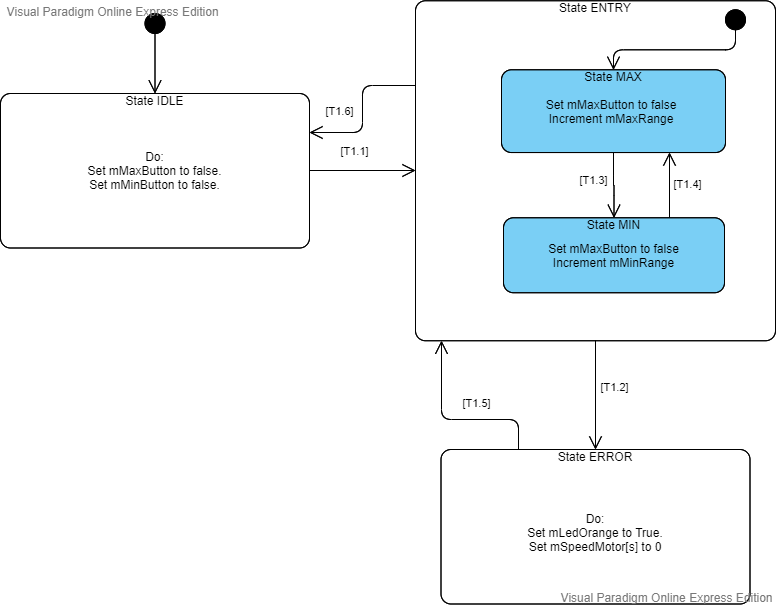
Scenario :

1. Kullanıcı güvenlik için gerekli parametreleri sisteme buton aracılığıyla girer.
2. Kullanıcının üst sınır için girdiği mesafede herhangi bir nesne vardır.
3. MaxMinRangeControl birimi ölçüm mesajını ve mMinRange ve mMaxRange değerlerini LedControl birimine iletir.
4. mMeasurement ile ölçüm alma işlemi MaxMinRangeControl tarafından başlatılır.
5. MaxMinRangeControl mMeasurement değerini LedControl birimine iletir.
6. LedControl birimi LedSecuritye mLedGreen = True değerini gönderir.
7. LedControl birimi MotorSpeedControl birimine mMotorSpeed[s]=1 değerini göndererek başlangıç hızında dönmesini sağlar.
8. MotorSpeedControl birimi hız değerini lcd ekranda göstermek için MaxMinRangeControl birimine mMotorSpeed değerini gönderir.



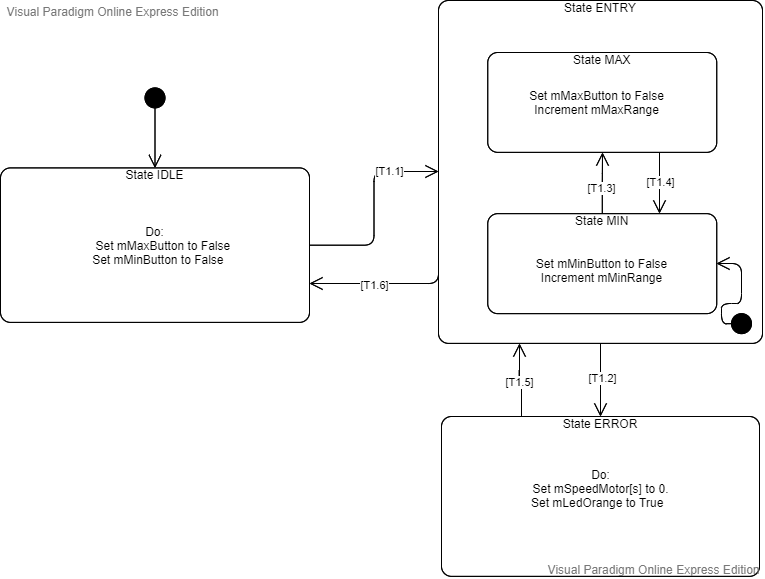
Use case 1A :MaxMinControl Statechart

|  |  |
| --- | --- |
| Transion | Guard |
| T1.1 | mMaxButton[s] = True and mMinButton[s] = True |
| T1.2 | mMinRange[s](counter=0) AND mMaxRange[s](counter=0) |
| T1.3 | mMinButton = True |
| T1.4 | mMaxButton = True |
| T1.5 | mMaxButton = True OR mMinButton =True |
| T1.6 | mMaxButton = False AND mMinButton =False |



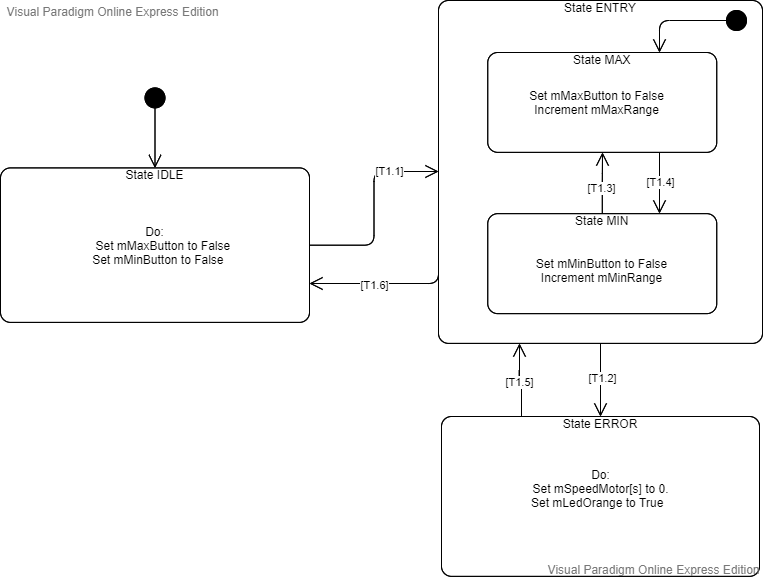
Use case 1B : MaxMinControl Statechart

|  |  |
| --- | --- |
| Transion | Guard |
| T1.1 | mMaxButton[s] = False AND mMinButton[s] = True |
| T1.2 | mMinRange[s](counter=0) AND mMaxRange[s](counter !=0) |
| T1.3 | mMaxButton = True |
| T1.4 | mMinButton = True |
| T1.5 | mMaxButton = False AND mMinButton =True |
| T1.6 | mMaxButton = False AND mMinButton =False |



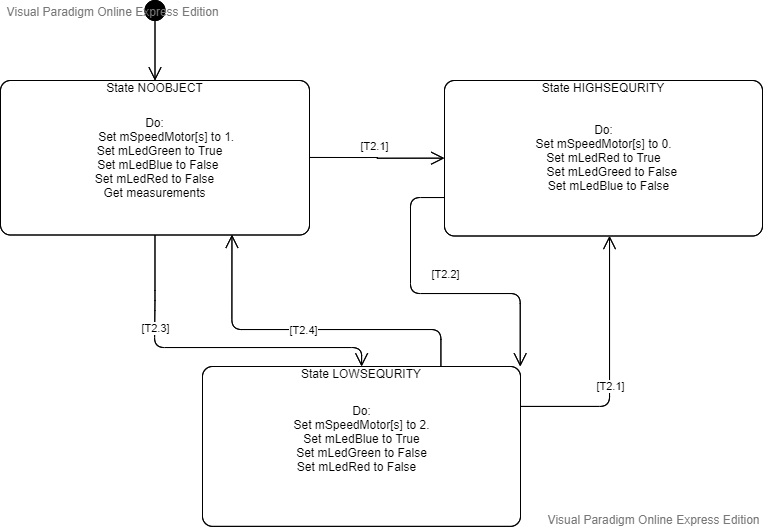
Use case 1C : MaxMinControl Statechart

|  |  |
| --- | --- |
| Transion | Guard |
| T1.1 | mMaxButton[s] = True and mMinButton[s] = False |
| T1.2 | mMinRange[s](counter !=0) and mMaxRange[s](counter =0) |
| T1.3 | mMaxButton = True |
| T1.4 | mMinButton = True |
| T1.5 | mMaxButton = True AND mMinButton =False |
| T1.6 | mMaxButton = False AND mMinButton =False |



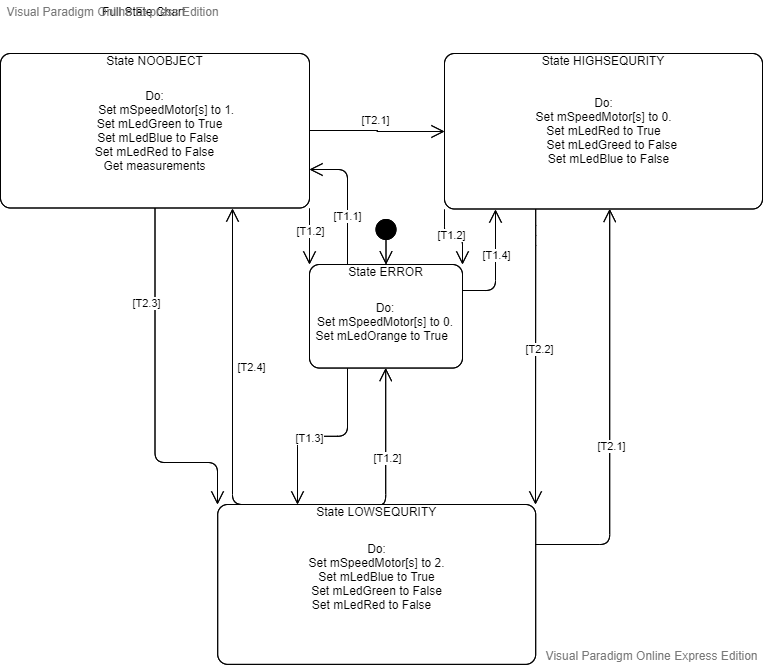
Use case 2-3-4: Ortamda nesne olup olmamasına göre statechart

|  |  |
| --- | --- |
| Transion | Guard |
| T2.1 | mMeasurement <= mMinRange |
| T2.2 | mMeasurement > mMinRange[s] |
| T2.3 | mMeasurement >= mMinRange AND mMeasurement <= mMaxRange+1 |
| T2.4 | mMeasurement > mMaxRange |



Use case All : Full Statechart

|  |  |
| --- | --- |
| Transion | Guard |
| T1.1 | mMinRange[s](counter!=0) AND mMaxRange[s](counter!=0) AND (mMeasurement > mMaxRange) (T2.2) |
| T1.2 | mMinRange[s](counter=0) OR mMaxRange[s](counter=0) |
| T1.3 | mMinRange[s](counter!=0) AND mMaxRange[s](counter!=0) AND (T2.3) |
| T1.4 | mMinRange[s](counter!=0) AND mMaxRange[s](counter!=0) AND (mMeasurement <= mMinRange) (T2.1) |
| T2.1 | mMeasurement <= mMinRange |
| T2.2 | mMeasurement >= mMinRange[s] AND mMeasurement <= mMaxRange |
| T2.3 | mMeasurement >= mMinRange AND mMeasurement <= mMaxRange |
| T2.4 | mMeasurement > mMaxRange |



Requirements to Statecharts Traceability

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Requirements** | | | |
| **States** | **R1.1** | **R2.1** | **R2.2** | **R2.3** |
| NO\_OBJECT |  |  |  | X |
| HIGH\_SEQURITY |  | X |  |  |
| LOW\_SEQURITY |  |  | X |  |
| ERROR | X |  |  |  |
| **Transitions** |  | | | |
| T2.1 |  | X |  |  |
| T2.2 |  |  |  |  |
| T2.3 |  |  | X |  |
| T2.4 |  |  |  | X |
| T1.1 |  |  |  | X |
| T1.2 | X | X | X | X |
| T1.3 |  |  | X |  |
| T1.4 |  | X |  |  |

**LedControl[s] Event Triggered Requirements**

* **ER1.1 If mMinRange[s](counter=0) OR mMaxRange[s](counter=0), then**
* ER1.1a mSpeedMotor[s] shall be set to 0.
* ER1.1b mLedOrange shall be set to False.

* **ER2.1 If mMinRange[s](counter !=0) AND mMaxRange[s](counter !=0) AND (mMeasurement <= mMinRange)**
* ER2.1a mSpeedMotor[s] shall be set to 0.
* ER2.1b mLedRed shall be set to True.
* ER2.1c mLedBlue shall be set to False.
* ER2.1d mLedGreen shall be set to False.
* ER2.1e mLedOrange shall be set to False.
* **ER2.2 mMinRange[s](counter !=0) AND mMaxRange[s](counter !=0) AND mMeasurement > mMinRange AND mMeasurement <= mMaxRange**
* ER2.2a mSpeedMotor[s] shall be set to 2.
* ER2.2b mLedBlue shall be set to True.
* ER2.2c mLedRed shall be set to False.
* ER2.2d mLedGreen shall be set to False.
* ER2.2e mLedOrange shall be set to False.
* **ER2.3 mMinRange[s](counter !=0) AND mMaxRange[s](counter !=0) AND (mMeasurement > mMaxRange)**
* ER2.3a mSpeedMotor[s] shall be set to 1.
* ER2.3b mLedGreen shall be set to True.
* ER2.3c mLedBlue shall be set to False.
* ER2.3d mLedRed shall be set to False.
* ER2.3e mLedOrange shall be set to False.

############################LEDCONTROL\_UNIT ##############################

mMeasurement\_ptr->id = CanReceiveData[0];

**if**(mMeasurement\_ptr->id != 70)

**return** -1;

mMeasurement\_ptr->data = CanReceiveData[1];

mMaxRange\_ptr->id = CanReceiveData[2];

**if**(mMaxRange\_ptr->id != 80)

**return** -1;

mMaxRange\_ptr->data = CanReceiveData[3];

mMinRange\_ptr->id = CanReceiveData[4];

**if**(mMinRange\_ptr->id != 90)

**return** -1;

mMinRange\_ptr->data = CanReceiveData[5];

**if**(mMaxRange\_ptr->data == 0 || mMinRange\_ptr->data == 0) #[T1.2]

state = *ERROR\_S*; // State ERROR

**else** **if**(mMeasurement\_ptr->data <= mMinRange\_ptr->data) #[T2.1]

state = *HIGHSEQURITY*; // State HIGHSEQURITY

**else** **if**(mMeasurement\_ptr->data > mMinRange\_ptr->data && mMeasurement\_ptr->data <= mMaxRange\_ptr->data) #[T2.3]

state = *LOWSEQURITY*; // State LOWSEQURITY

**else** **if**(mMeasurement\_ptr->data >= mMaxRange\_ptr->data) #[T2.4]

state = *NO\_OBJECT*; // State NO\_OBJECT

**switch**(state)

{

**case** *ERROR\_S*:

{

HAL\_GPIO\_WritePin(GPIOD, mLedOrange, *GPIO\_PIN\_SET*);

HAL\_GPIO\_WritePin(GPIOD, mLedBlue, *GPIO\_PIN\_RESET*);

HAL\_GPIO\_WritePin(GPIOD, mLedRed, *GPIO\_PIN\_RESET*);

HAL\_GPIO\_WritePin(GPIOD, mLedGreen, *GPIO\_PIN\_RESET*);

mMotorSpeed.data = 0;

mMotorSpeed.id = 50;

CanTransData[1]= mMotorSpeed.data; // mMotorSpeed =0

CanTransData[0]= mMotorSpeed.id; // mMotorSpeed =0

HAL\_CAN\_AddTxMessage(&hcan1, &pTxHeader, CanTransData, &mTxMailbox); / //Send mMotorSpeed to MotorServoControl unit

**if**((mMeasurement\_ptr->data <= mMinRange\_ptr->data) && (mMaxRange\_ptr->data != 0 && mMinRange\_ptr->data != 0))//# [T1.4]

state = *HIGHSEQURITY*;

**else** **if**((mMeasurement\_ptr->data > mMinRange\_ptr->data) && (mMeasurement\_ptr->data <= mMaxRange\_ptr->data) && (mMaxRange\_ptr->data != 0 && mMinRange\_ptr->data != 0)) //#[T1.3]

state = *LOWSEQURITY*;

**else** **if**((mMeasurement\_ptr->data >= mMaxRange\_ptr->data) && (mMaxRange\_ptr->data != 0 && mMinRange\_ptr->data != 0)) //#[T1.1]

state = *NO\_OBJECT*;

**else**

state =*ERROR\_S*;

**break**;

}

**case** *HIGHSEQURITY*:

{

HAL\_GPIO\_WritePin(GPIOD, mLedOrange, *GPIO\_PIN\_RESET*);

HAL\_GPIO\_WritePin(GPIOD, mLedBlue, *GPIO\_PIN\_RESET*);

HAL\_GPIO\_WritePin(GPIOD, mLedGreen, *GPIO\_PIN\_RESET*);

HAL\_GPIO\_WritePin(GPIOD, mLedRed, *GPIO\_PIN\_SET*);

mMotorSpeed.data = 0;

mMotorSpeed.id = 50;

CanTransData[0]= mMotorSpeed.id; // mMotorSpeed = 0

CanTransData[1]= mMotorSpeed.data;

HAL\_CAN\_AddTxMessage(&hcan1, &pTxHeader, CanTransData, &mTxMailbox);// Send mMotorSpeed to MotorServoControl unit

**if**((mMeasurement\_ptr->data > mMinRange\_ptr->data) && mMeasurement\_ptr->data <= mMaxRange\_ptr->data)) //#[T2.2]

state =*LOWSEQURITY*;

**break**;

}

**case** *LOWSEQURITY*:

{

HAL\_GPIO\_WritePin(GPIOD, mLedOrange, *GPIO\_PIN\_RESET*);

HAL\_GPIO\_WritePin(GPIOD, mLedGreen, *GPIO\_PIN\_RESET*);

HAL\_GPIO\_WritePin(GPIOD, mLedRed, *GPIO\_PIN\_RESET*);

HAL\_GPIO\_WritePin(GPIOD, mLedBlue, *GPIO\_PIN\_SET*);

mMotorSpeed.data = 2;

mMotorSpeed.id = 50;

CanTransData[0]= mMotorSpeed.id; // mMotorSpeed id

CanTransData[1]= mMotorSpeed.data;// mMotorSpeed =2

HAL\_CAN\_AddTxMessage(&hcan1, &pTxHeader, CanTransData, &mTxMailbox);// Send mMotorSpeed to MotorServoControl unit

**break**;

}

**case** *NO\_OBJECT*:

{

HAL\_GPIO\_WritePin(GPIOD, mLedOrange, *GPIO\_PIN\_RESET*);

HAL\_GPIO\_WritePin(GPIOD, mLedBlue, *GPIO\_PIN\_RESET*);

HAL\_GPIO\_WritePin(GPIOD, mLedRed, *GPIO\_PIN\_RESET*);

HAL\_GPIO\_WritePin(GPIOD, mLedGreen, *GPIO\_PIN\_SET*);

mMotorSpeed.data = 1;

mMotorSpeed.id = 50;

CanTransData[0]= mMotorSpeed.id;

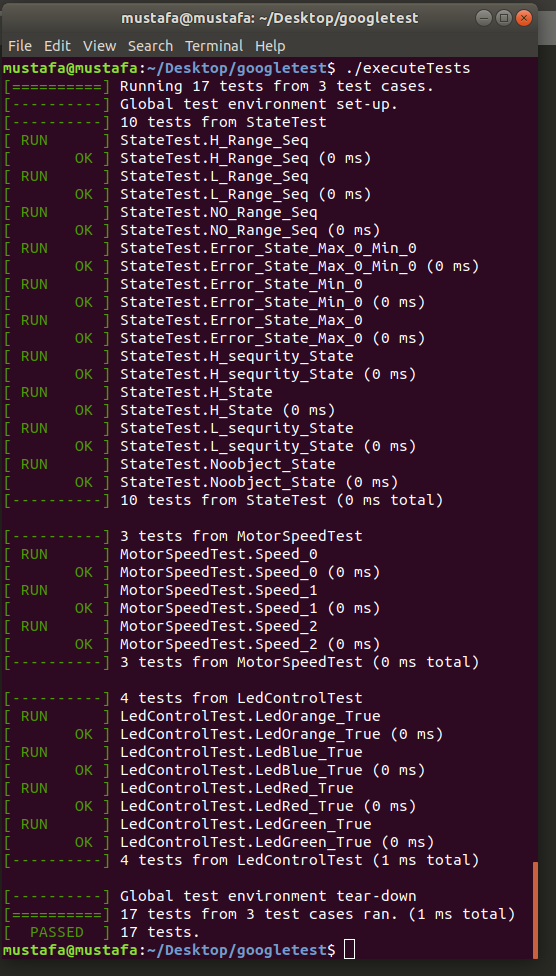
CanTransData[1]= mMotorSpeed.data; // mMotorSpeed =1

HAL\_CAN\_AddTxMessage(&hcan1, &pTxHeader, CanTransData, &mTxMailbox);// Send mMotorSpeed to MotorServoControl unit

**break**;

}

}



UNIT TEST

#include <gtest/gtest.h>

#include "CAN\_Tx\_Rx.c"

#include "StateLed.h"

#include "MotorServoControl.c"

#include "ledcontrol.c"

uint8\_t CanReceiveData[8];

using namespace std;

TEST(StateTest, H\_Range\_Seq)

{

uint8\_t Measurement=0;

while(Measurement <=5)

{

CanReceiveData[0]=70; // Measurement id

CanReceiveData[1]= Measurement; // Measurement

CanReceiveData[2]=80; // MaxRange id

CanReceiveData[3]=10; // MaxRange data

CanReceiveData[4]=90; // MinRange id

CanReceiveData[5]=5; // MinRange data

ASSERT\_EQ(HIGHSEQURITY, CanReceiveData\_From\_User(CanReceiveData));

ASSERT\_NE(ERROR\_S, CanReceiveData\_From\_User(CanReceiveData));

ASSERT\_NE(NO\_OBJECT, CanReceiveData\_From\_User(CanReceiveData));

ASSERT\_NE(LOWSEQURITY, CanReceiveData\_From\_User(CanReceiveData));

Measurement++;

}

}

TEST(StateTest, L\_Range\_Seq)

{

uint8\_t Measurement=6;

while(Measurement > 5 && Measurement <=10)

{

CanReceiveData[0]=70; // Measurement id

CanReceiveData[1]= Measurement; // Measurement

CanReceiveData[2]=80; // MaxRange id

CanReceiveData[3]=10; // MaxRange data

CanReceiveData[4]=90; // MinRange id

CanReceiveData[5]=5; // MinRange data

ASSERT\_EQ(LOWSEQURITY, CanReceiveData\_From\_User(CanReceiveData));

ASSERT\_NE(ERROR\_S, CanReceiveData\_From\_User(CanReceiveData));

ASSERT\_NE(NO\_OBJECT, CanReceiveData\_From\_User(CanReceiveData));

ASSERT\_NE(HIGHSEQURITY, CanReceiveData\_From\_User(CanReceiveData));

Measurement++;

}

}

TEST(StateTest, NO\_Range\_Seq)

{

uint8\_t Measurement=11;

while(Measurement >10)

{

CanReceiveData[0]=70; // Measurement id

CanReceiveData[1]= Measurement; // Measurement

CanReceiveData[2]=80; // MaxRange id

CanReceiveData[3]=10; // MaxRange data

CanReceiveData[4]=90; // MinRange id

CanReceiveData[5]=5; // MinRange data

ASSERT\_EQ(NO\_OBJECT, CanReceiveData\_From\_User(CanReceiveData));

ASSERT\_NE(ERROR\_S, CanReceiveData\_From\_User(CanReceiveData));

ASSERT\_NE(LOWSEQURITY, CanReceiveData\_From\_User(CanReceiveData));

ASSERT\_NE(HIGHSEQURITY, CanReceiveData\_From\_User(CanReceiveData));

Measurement++;

}

}

TEST(StateTest, Error\_State\_Max\_0\_Min\_0)

{

CanReceiveData[0]=70; // Measurement id

CanReceiveData[1]=43; // Measurement

CanReceiveData[2]=80; // MaxRange id

CanReceiveData[3]=0; // MaxRange data

CanReceiveData[4]=90; // MinRange id

CanReceiveData[5]=0; // MinRange data

ASSERT\_EQ(ERROR\_S, CanReceiveData\_From\_User(CanReceiveData));

ASSERT\_NE(NO\_OBJECT, CanReceiveData\_From\_User(CanReceiveData));

ASSERT\_NE(LOWSEQURITY, CanReceiveData\_From\_User(CanReceiveData));

ASSERT\_NE(HIGHSEQURITY, CanReceiveData\_From\_User(CanReceiveData));

}

TEST(StateTest, Error\_State\_Min\_0)

{

CanReceiveData[0]=70; // Measurement id

CanReceiveData[1]=43; // Measurement

CanReceiveData[2]=80; // MaxRange id

CanReceiveData[3]=10; // MaxRange data

CanReceiveData[4]=90; // MinRange id

CanReceiveData[5]=0; // MinRange data

ASSERT\_EQ(ERROR\_S, CanReceiveData\_From\_User(CanReceiveData));

ASSERT\_NE(NO\_OBJECT, CanReceiveData\_From\_User(CanReceiveData));

ASSERT\_NE(LOWSEQURITY, CanReceiveData\_From\_User(CanReceiveData));

ASSERT\_NE(HIGHSEQURITY, CanReceiveData\_From\_User(CanReceiveData));

}

TEST(StateTest, Error\_State\_Max\_0)

{

CanReceiveData[0]=70; // Measurement id

CanReceiveData[1]=43; // Measurement

CanReceiveData[2]=80; // MaxRange id

CanReceiveData[3]=0; // MaxRange data

CanReceiveData[4]=90; // MinRange id

CanReceiveData[5]=35; // MinRange data

ASSERT\_EQ(ERROR\_S, CanReceiveData\_From\_User(CanReceiveData));

ASSERT\_NE(NO\_OBJECT, CanReceiveData\_From\_User(CanReceiveData));

ASSERT\_NE(LOWSEQURITY, CanReceiveData\_From\_User(CanReceiveData));

ASSERT\_NE(HIGHSEQURITY, CanReceiveData\_From\_User(CanReceiveData));

}

TEST(StateTest, H\_sequrity\_State)

{

CanReceiveData[0]=70; // Measurement id

CanReceiveData[1]=5; // Measurement

CanReceiveData[2]=80; // MaxRange id

CanReceiveData[3]=10; // MaxRange data

CanReceiveData[4]=90; // MinRange id

CanReceiveData[5]=5; // MinRange data

ASSERT\_EQ(HIGHSEQURITY, CanReceiveData\_From\_User(CanReceiveData));

ASSERT\_NE(ERROR\_S, CanReceiveData\_From\_User(CanReceiveData));

ASSERT\_NE(NO\_OBJECT, CanReceiveData\_From\_User(CanReceiveData));

ASSERT\_NE(LOWSEQURITY, CanReceiveData\_From\_User(CanReceiveData));

}

TEST(StateTest, H\_State)

{

CanReceiveData[0]=70; // Measurement id

CanReceiveData[1]=5; // Measurement

CanReceiveData[2]=80; // MaxRange id

CanReceiveData[3]=50; // MaxRange data

CanReceiveData[4]=90; // MinRange id

CanReceiveData[5]=40; // MinRange data

ASSERT\_EQ(HIGHSEQURITY, CanReceiveData\_From\_User(CanReceiveData));

ASSERT\_NE(ERROR\_S, CanReceiveData\_From\_User(CanReceiveData));

ASSERT\_NE(NO\_OBJECT, CanReceiveData\_From\_User(CanReceiveData));

ASSERT\_NE(LOWSEQURITY, CanReceiveData\_From\_User(CanReceiveData));

}

TEST(StateTest, L\_sequrity\_State)

{

CanReceiveData[0]=70; // Measurement id

CanReceiveData[1]=10; // Measurement

CanReceiveData[2]=80; // MaxRange id

CanReceiveData[3]=10; // MaxRange data

CanReceiveData[4]=90; // MinRange id

CanReceiveData[5]=5; // MinRange data

ASSERT\_EQ(LOWSEQURITY, CanReceiveData\_From\_User(CanReceiveData));

ASSERT\_NE(HIGHSEQURITY, CanReceiveData\_From\_User(CanReceiveData));

ASSERT\_NE(ERROR\_S, CanReceiveData\_From\_User(CanReceiveData));

ASSERT\_NE(NO\_OBJECT, CanReceiveData\_From\_User(CanReceiveData));

}

TEST(StateTest, Noobject\_State)

{

CanReceiveData[0]=70; // Measurement id

CanReceiveData[1]=15; // Measurement

CanReceiveData[2]=80; // MaxRange id

CanReceiveData[3]=10; // MaxRange data

CanReceiveData[4]=90; // MinRange id

CanReceiveData[5]=5; // MinRange data

ASSERT\_EQ(NO\_OBJECT, CanReceiveData\_From\_User(CanReceiveData));

ASSERT\_NE(LOWSEQURITY, CanReceiveData\_From\_User(CanReceiveData));

ASSERT\_NE(HIGHSEQURITY, CanReceiveData\_From\_User(CanReceiveData));

ASSERT\_NE(ERROR\_S, CanReceiveData\_From\_User(CanReceiveData));

}

TEST(MotorSpeedTest, Speed\_0)

{

CanReceiveData[0]=50; // MotorSpeed\_id

CanReceiveData[1]=0; // Speed\_Value

ASSERT\_EQ(0, MotorServoControl(CanReceiveData));

}

TEST(MotorSpeedTest, Speed\_1)

{

CanReceiveData[0]=50; // MotorSpeed\_id

CanReceiveData[1]=1; // Speed\_Value

ASSERT\_EQ(1, MotorServoControl(CanReceiveData));

}

TEST(MotorSpeedTest, Speed\_2)

{

CanReceiveData[0]=50; // MotorSpeed\_id

CanReceiveData[1]=2; // Speed\_Value

ASSERT\_EQ(2, MotorServoControl(CanReceiveData));

ASSERT\_NE(0, MotorServoControl(CanReceiveData));

ASSERT\_NE(1, MotorServoControl(CanReceiveData));

}

TEST(LedControlTest, LedOrange\_True)

{

CanReceiveData[0]=70; // Measurement id

CanReceiveData[1]=10; // Measurement

CanReceiveData[2]=80; // MaxRange id

CanReceiveData[3]=10; // MaxRange data

CanReceiveData[4]=90; // MinRange id

CanReceiveData[5]=0; // MinRange data

ASSERT\_TRUE(ledControlUnit(CanReceiveData));

}

TEST(LedControlTest, LedBlue\_True)

{

CanReceiveData[0]=70; // Measurement id

CanReceiveData[1]=10; // Measurement

CanReceiveData[2]=80; // MaxRange id

CanReceiveData[3]=10; // MaxRange data

CanReceiveData[4]=90; // MinRange id

CanReceiveData[5]=5; // MinRange data

ASSERT\_TRUE(ledControlUnit(CanReceiveData));

}

TEST(LedControlTest, LedRed\_True)

{

CanReceiveData[0]=70; // Measurement id

CanReceiveData[1]=5; // Measurement

CanReceiveData[2]=80; // MaxRange id

CanReceiveData[3]=10; // MaxRange data

CanReceiveData[4]=90; // MinRange id

CanReceiveData[5]=5; // MinRange data

ASSERT\_TRUE(ledControlUnit(CanReceiveData));

}

TEST(LedControlTest, LedGreen\_True)

{

CanReceiveData[0]=70; // Measurement id

CanReceiveData[1]=15; // Measurement

CanReceiveData[2]=80; // MaxRange id

CanReceiveData[3]=10; // MaxRange data

CanReceiveData[4]=90; // MinRange id

CanReceiveData[5]=5; // MinRange data

ASSERT\_TRUE(ledControlUnit(CanReceiveData));

}

int main(int argc, char \*\*argv)

{

testing::InitGoogleTest(&argc, argv);

return RUN\_ALL\_TESTS();

}q1