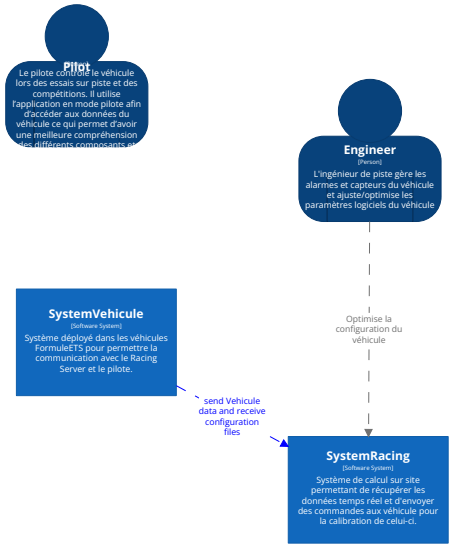
FormuleETS DashView project

L'objectif de ce document d'architecture est principalement de démontrer aux étudiants de l'ÉTS comment créer et formater un document d'architecture logiciels.

Nous nous sommes basé sur le document de spécification suivant pour réaliser ce document.
https://docs.google.com/document/d/14a59GwEGnIW3WzfuF_-8D4op0vDEPHjpkoNcHlEko0w/edit#heading=h.38czs75

Voici une vue architecturale présentant tout les acteurs et systèmes impliqué dans le projet DashView FormuleETS.

Cet ensemble de système est utilisé pour faire l'acquisition de données, configurer et calibre les véhicules du club FormuleETS.

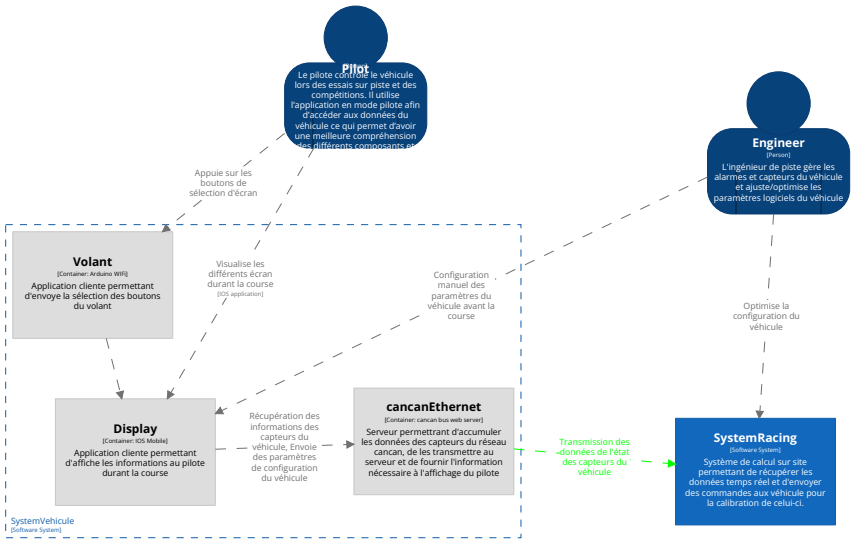


[System Landscape] FormuleETS
Diagramme d'integration de tout les systèmes pour le projet FormuleETS DashView
Tuesday, November 15, 2022 at 1:11 PM Eastern Standard Time

Voici la vue de context du serveur de gestion de la course.

Unable to embed view 'racingSystemView' - there is no view with this key in the workspace.

Voici la vue de contexte du véhicule FormuleETS



[Container] SystemVehicule

Diagramme de décomposition du véhicule

Tuesday, November 15, 2022 at 1:11 PM Eastern Standard Time

0.1 Table des éléments du conteneur véhicule

0.2 Volant

Application cliente permettant d'envoye la sélection des boutons du volant

Tags: Element,Container

Parent	Key	Category	Title	Priority
null	EF10	Mode pilote	Changer l'interface lors de l'appui sur le bouton du volant L'application doit changer l'interface affichée lorsque le pilote appuie sur un bouton du volant. Le bouton envoie un message CAN à l'application pour lui indiquer de changer. L'application en mode pilote affichera quatre interfaces différentes en boucle.	4

0.2.1 Perspectives

EF10 - Changer l'interface lors de l'appui sur le bouton du volant L'application doit changer l'interface affichée lorsque le pilote appuie sur un bouton du volant. Le bouton envoie un message CAN à l'application pour lui indiquer de changer. L'application en mode pilote affichera quatre interfaces différentes en boucle.

0.3 cancanEthernet

Serveur permettrant d'accumuler les données des capteurs du réseau cancan, de les transmettre au serveur et de fournir l'information nécessaire à l'affichage du pilote

Tags: Element,Container

Parent	Key	Category	Title	Priority
null	C03	null	can2Ethernet La librairie Can2Ethernet développée par le club Formule ÉTS doit être utilisée.	9
null	C05	Performance	fi UDP	4

			Les données sont envoyées par Wi-Fi via le protocole UDP et le bus CAN.	
null	EF11	Mode pilote	Contenu de la première interface L'application doit afficher, sur la première interface, les capteurs suivants : la température du moteur, le voltage de la batterie, l'indicateur d'utilisation et d'angle du système de réduction de traînée, les différentes alarmes, la boîte de messages, l'indicateur de vitesse et l'indicateur de révolutions par minute.	4
null	EF12	Mode pilote	Contenu de la deuxième interface Le Dash Display doit afficher, sur la deuxième interface, les capteurs suivants : la pression et la température des pneus, le biais de frein, l'antiroulis, l'odomètre, le voltage de la batterie, la température du moteur et un indicateur d'utilisation et l'angle du système de réduction de traînée.	4
null	EF13	Mode pilote	Contenu de la troisième interface L'application doit afficher, sur la troisième interface, le temps total de course, le temps du tour courant, la différence de temps par rapport au meilleur temps, le meilleur temps de tour de piste et la différence de temps par rapport au dernier de tour de piste.	4
null	EF14	Mode pilote	Contenu de la quatrième interface Dash Display doit afficher, sur la quatrième interface, un schéma de la piste de course et le déplacement de la voiture en temps réel. De plus, les temps suivants doivent être présents : le temps du tour de piste, le meilleur temps et la différence par rapport au meilleur temps.	4

0.3.1 Perspectives

EF11 - Contenu de la première interface L'application doit afficher, sur la première interface, les capteurs suivants : la température du moteur, le voltage de la batterie, l'indicateur d'utilisation et d'angle du système de réduction de traînée, les différentes alarmes, la boîte de messages, l'indicateur de vitesse et l'indicateur de révolutions par minute.

EF12 - Contenu de la deuxième interface Le Dash Display doit afficher, sur la deuxième interface, les capteurs suivants : la pression et la température des pneus, le biais de frein, l'antiroulis, l'odomètre, le voltage de la batterie, la température du moteur et un indicateur d'utilisation et l'angle du système de réduction de traînée.

EF13 - Contenu de la troisième interface L'application doit afficher, sur la troisième interface, le temps total de course, le temps du tour courant, la différence de temps par rapport au meilleur temps, le meilleur temps de tour de piste et la différence de temps par rapport au dernier de tour de piste.

EF14 - Contenu de la quatrième interface Dash Display doit afficher, sur la quatrième interface, un schéma de la piste de course et le déplacement de la voiture en temps réel. De plus, les temps suivants doivent être présents : le temps du tour de piste, le meilleur temps et la différence par rapport au meilleur temps.

C03 - can2Ethernet La librairie Can2Ethernet développée par le club Formule ÉTS doit être utilisée.

C05 - Wifi UDP Les données sont envoyées par Wi-Fi via le protocole UDP et le bus CAN.

0.4 Pilot

Le pilote contrôle le véhicule lors des essais sur piste et des compétitions. Il utilise l'application en mode pilote afin d'accéder aux données du véhicule ce qui permet d'avoir une meilleure compréhension des différents composants et d'améliorer sa conduite.

Tags: Element,Person

Parent	Key	Category	Title	Priority
null	EF02	Générale	Configuration de l'application dans les paramètres d'iOS Le système doit permettre de changer quelques configurations directement dans les paramètres de l'application sur iOS. Les configurations doivent inclure, entre autres, le changement de mode entre pilote et ingénieur de piste ainsi que le changement des couleurs de l'interface de pâle à foncé.	2
null	EF09	Mode pilote	Visualiser les alarmes et les capteurs sur l'interface pilote L'application doit afficher, de manière claire, les informations nécessaires au pilote en provenance des différents capteurs du véhicule. De plus, les messages d'alarmes seront aussi affichés de façon évidente et de façon à ce que le pilote les remarques immédiatement.	6
null	EF10	Mode pilote	Changer l'interface lors de l'appui sur le bouton du volant	4

			L’application doit changer l’interface affichée lorsque le pilote appuie sur un bouton du volant. Le bouton envoie un message CAN à l’application pour lui indiquer de changer. L’application en mode pilote affichera quatre interfaces différentes en boucle.	
null	EF11	Mode pilote	Contenu de la première interface	4
			L’application doit afficher, sur la première interface, les capteurs suivants : la température du moteur, le voltage de la batterie, l’indicateur d’utilisation et d’angle du système de réduction de traînée, les différentes alarmes, la boîte de messages, l’indicateur de vitesse et l’indicateur de révolutions par minute.	
null	EF12	Mode pilote	Contenu de la deuxième interface	4
			Le Dash Display doit afficher, sur la deuxième interface, les capteurs suivants : la pression et la température des pneus, le biais de frein, l’antiroulis, l’odomètre, le voltage de la batterie, la température du moteur et un indicateur d’utilisation et l’angle du système de réduction de traînée.	
null	EF13	Mode pilote	Contenu de la troisième interface	4
			L’application doit afficher, sur la troisième interface, le temps total de course, le temps du tour courant, la différence de temps par rapport au meilleur temps, le meilleur temps de tour de piste et la différence de temps par rapport au dernier de tour de piste.	
null	EF14	Mode pilote	Contenu de la quatrième interface	4
			Dash Display doit afficher, sur la quatrième interface, un schéma de la piste de course et le déplacement de la voiture en temps réel. De plus, les temps suivants doivent être présents : le temps du tour de piste, le meilleur temps et la différence par rapport au meilleur temps.	
null	EF15	Mode pilote	Affichage en mode paysage pour le mode pilote	1
			doit afficher les interfaces en mode pilote sous le format paysage.	
null	EF16	Mode pilote	Mettre en veille l’application après 3 secondes sans données	1
			doit se mettre en veille si une interruption de données survient et si elle dure plus de trois secondes.	

0.4.1 Perspectives

EF10 - Changer l’interface lors de l’appui sur le bouton du volant L’application doit changer l’interface affichée lorsque le pilote appuie sur un bouton du volant. Le bouton envoie un message CAN à l’application pour lui indiquer de changer. L’application en mode pilote affichera quatre interfaces différentes en boucle.

EF11 - Le pilot doit pouvoir activer l'écran 1 a partir d'un bouton sur le volant

EF12 - Le pilot doit pouvoir activer l'écran 2 a partir d'un bouton sur le volant

EF02 - Le système doit permettre de changer quelques configurations directement dans les paramètres de l’application sur iOS. Les configurations doivent inclure, entre autres, le changement de mode entre pilote et ingénieur de piste ainsi que le changement des couleurs de l’interface de pâle à foncé.

EF13 - Le pilot doit pouvoir activer l'écran 3 a partir d'un bouton sur le volant

EF15 - Affichage en mode paysage pour le mode pilote

EF16 - Mettre en veille l’application après 3 secondes sans donnéesÉ

EF09 - Visualiser les alarmes et les capteurs sur l’interface pilote L’application doit afficher, de manière claire, les informations nécessaires au pilote en provenance des différents capteurs du véhicule. De plus, les messages d’alarmes seront aussi affichés de façon évidente et de façon à ce que le pilote les remarques immédiatement.

0.5 Engineer

L'ingénieur de piste gère les alarmes et capteurs du véhicule et ajuste/optimise les paramètres logiciels du véhicule

Tags: Element,Person

Parent	Key	Category	Title	Priority
null	EF01	Général	Configuration de l’application avec un fichier XML	4
			L’application doit utiliser un fichier de configuration, sous le format XML, pour déterminer les alarmes et capteurs disponibles. La liste des alarmes et des capteurs sont définis selon la table CAN fournie par la Formule ÉTS.	
null	EF17	Mode ingénieur de piste	Visualiser les alarmes et les capteurs sur l’interface ingénieur	6
			doit afficher, sous forme de liste, les différents capteurs et alarmes que l’utilisateur décide d’inclure. Les capteurs et les alarmes sont affichés séparément, les alarmes se trouvant en haut de la liste. Lors de la présentation du prototype, le club formule ÉTS a précisé qu’ils souhaiteraient traiter les alarmes de la même façon que les différents capteurs donc les rendre modifiable sur la page principale.	
null	EF18	Mode ingénieur de piste	Ajouter une alarme ou un capteur	2
			doit permettre à l’utilisateur de sélectionner dans une liste une alarme ou un capteur à ajouter à la liste d’affichage. L’application doit aussi permettre de filtrer la liste des alarmes et des capteurs qui peuvent être ajoutés et d’y effectuer une recherche.	
null	EF19	Mode ingénieur de piste	Changer l’ordre des alarmes et des capteurs affichés	4
			doit permettre à l’utilisateur, une fois en mode édition de la liste, de réorganiser respectivement les alarmes et les capteurs entre eux.	
null	EF20	Mode ingénieur de piste	Supprimer une alarme ou un capteur affiché	1
			doit permettre à l’utilisateur, une fois en mode édition de la liste, de supprimer un capteur ou une alarme.	
null	EF21	Mode ingénieur de piste	Afficher les détails de l’alarme ou du capteur	4
			système doit permettre de cliquer sur un capteur ou une alarme affichés afin d’obtenir plus de détails. Lors de la présentation du prototype, cette exigence a été clarifiée. Le client désire avoir la possibilité de modifier l’affichage du widget pour d’autres formats ainsi qu’obtenir un historique des dernières données. Les informations apparaissent sous le widget principal avec la possibilité d’afficher l’historique en plein écran.	
null	EF22	Mode ingénieur de piste	Gérer les cas d’erreurs de l’application	9
			doit, en cas d’erreurs de l’application, afficher les dernières données reçues. Les cas d’erreurs peuvent être, par exemple, une erreur de transmission de données ou un message d’erreur reçu par une chaîne CAN du module Can2Ethernet.	

0.5.1 Perspectives

EF21 - Afficher les détails de l’alarme ou du capteur

EF22 - Gérer les cas d’erreurs de l’application

EF01 - Configuration de l’application avec un fichier XML

C01 - L'ingénieur est responsable de réaliser le fichier de configureation XML

EF17 - Visualiser les alarmes et les capteurs sur l’interface ingénieur

EF18 - Ajouter une alarme ou un capteur doit permettre à l’utilisateur de sélectionner dans une liste une alarme ou un capteur à ajouter à la liste d’affichage. L’application doit aussi permettre de filtrer la liste des alarmes et des capteurs qui peuvent être ajoutés et d’y effectuer une recherche.

EF19 - Changer l’ordre des alarmes et des capteurs affichés

0.6 SystemRacing

Système de calcul sur site permettant de récupérer les données temps réel et d'envoyer des commandes aux véhicule pour la calibration de celui-ci.

Tags: Element,Software System

Parent	Key	Category	Title	Priority
null	C01	null	Usability	2
			Les configurations pour les catégories des alarmes et des capteurs doivent être définies dans un fichier XML.	
null	C04	null	Objective-C	1
			L'application doit être en Objective-C sous la plateforme iOS 7 et est destinée aux iPod Touch de 5e génération avec un écran de 4 pouces	
null	C06	null	P port	1
			Le port UDP est le port par défaut soit 1337.	
null	C08	null	langue	1
			Les textes de l'application doivent être en anglais.	
null	EF01	Général	Configuration de l'application avec un fichier XML	4
			L'application doit utiliser un fichier de configuration, sous le format XML, pour déterminer les alarmes et capteurs disponibles. La liste des alarmes et des capteurs sont définis selon la table CAN fournie par la Formule ÉTS.	
null	EF03	Générale	Gérer les données reçues en temps réel	9
			L'application doit constamment recevoir des données du bus CAN via Wi-Fi à partir du module Can2Ethernet et, en comparant avec la table des messages CAN, associer ces données aux capteurs et alarmes pour les afficher.	
null	EF17	Mode ingénieur de piste	Visualiser les alarmes et les capteurs sur l'interface ingénieur	6
			doit afficher, sous forme de liste, les différents capteurs et alarmes que l'utilisateur décide d'inclure. Les capteurs et les alarmes sont affichés séparément, les alarmes se trouvant en haut de la liste. Lors de la présentation du prototype, le club formule ÉTS a précisé qu'ils souhaiteraient traiter les alarmes de la même façon que les différents capteurs donc les rendre modifiable sur la page principale.	
null	EF18	Mode ingénieur de piste	Ajouter une alarme ou un capteur	2
			doit permettre à l'utilisateur de sélectionner dans une liste une alarme ou un capteur à ajouter à la liste d'affichage. L'application doit aussi permettre de filtrer la liste des alarmes et des capteurs qui peuvent être ajoutés et d'y effectuer une recherche.	
null	EF19	Mode ingénieur de piste	Changer l'ordre des alarmes et des capteurs affichés	4
			doit permettre à l'utilisateur, une fois en mode édition de la liste, de réorganiser respectivement les alarmes et les capteurs entre eux.	
null	EF20	Mode ingénieur de piste	Supprimer une alarme ou un capteur affiché	1
			doit permettre à l'utilisateur, une fois en mode édition de la liste, de supprimer un capteur ou une alarme.	
null	EF21	Mode ingénieur de piste	Afficher les détails de l'alarme ou du capteur	4
			système doit permettre de cliquer sur un capteur ou une alarme affichés afin d'obtenir plus de détails. Lors de la présentation du prototype, cette exigence a été clarifiée. Le client désire avoir la possibilité de modifier l'affichage du widget pour d'autres formats ainsi qu'obtenir un historique des	

			dernières données. Les informations apparaissent sous le widget principal avec la possibilité d'afficher l'historique en plein écran.	
null	EF22	Mode ingénieur de piste	Gérer les cas d'erreurs de l'application	9
			doit, en cas d'erreurs de l'application, afficher les dernières données reçues. Les cas d'erreurs peuvent être, par exemple, une erreur de transmission de données ou un message d'erreur reçu par une chaîne CAN du module Can2Ethernet.	
null	ENF07	Usability	Modification rapide des alarmes et des capteurs	9
			La liste des alarmes et des capteurs peut facilement être modifiée en 5 minutes et moins. Cette modification est effectuée sur le fichier de configuration XML en fournissant le « id » et l'« offset » définis dans la table CAN. Cette exigence n'était pas précisée de la part du client, mais une précision concernant la table CAN qui peut être modifiée a été faite. Il est donc important que tout le logiciel soit facilement maintenable.	

0.6.1 Perspectives

EF21 - Afficher les détails de l'alarme ou du capteur système doit permettre de cliquer sur un capteur ou une alarme affichés afin d'obtenir plus de détails. Lors de la présentation du prototype, cette exigence a été clarifiée. Le client désire avoir la possibilité de modifier l'affichage du widget pour d'autres formats ainsi qu'obtenir un historique des dernières données. Les informations apparaissent sous le widget principal avec la possibilité d'afficher l'historique en plein écran.

EF22 - Gérer les cas d'erreurs de l'application doit, en cas d'erreurs de l'application, afficher les dernières données reçues. Les cas d'erreurs peuvent être, par exemple, une erreur de transmission de données ou un message d'erreur reçu par une chaîne CAN du module Can2Ethernet.

EF01 - Configuration de l'application avec un fichier XML

EF03 - Gérer les données reçues en temps réel L'application doit constamment recevoir des données du bus CAN via Wi-Fi à partir du module Can2Ethernet et, en comparant avec la table des messages CAN, associer ces données aux capteurs et alarmes pour les afficher.

C01 - Les configurations pour les catégories des alarmes et des capteurs doivent être définies dans un fichier XML.

C04 - L'application doit être en Objective-C sous la plateforme iOS 7 et est destinée aux iPod Touch de 5e génération avec un écran de 4 pouces

C03 - Recoit les données du véhicule et archives celles-ci

EF17 - Visualiser les alarmes et les capteurs sur l'interface ingénieur Doit afficher, sous forme de liste, les différents capteurs et alarmes que l'utilisateur décide d'inclure. Les capteurs et les alarmes sont affichés séparément, les alarmes se trouvant en haut de la liste. Lors de la présentation du prototype, le club formule ÉTS a précisé qu'ils souhaiteraient traiter les alarmes de la même façon que les différents capteurs donc les rendre modifiable sur la page principale.

C06 - Le port UDP est le port par défaut soit 1337.

EF19 - Changer l'ordre des alarmes et des capteurs affichés doit permettre à l'utilisateur, une fois en mode édition de la liste, de réorganiser respectivement les alarmes et les capteurs entre eux.

C08 - Les textes de l'application doivent être en anglais.

EF20 - Supprimer une alarme ou un capteur affiché doit permettre à l'utilisateur, une fois en mode édition de la liste, de supprimer un capteur ou une alarme.

0.7 Display

Application cliente permettant d'affiche les informations au pilote durant la course

Tags: Element,Container

Parent	Key	Category	Title	Priority
null	C02	null	Usability	2
			Les configurations pour le mode par défaut et les couleurs de l'interface sont définies directement	

			dans iOS.	
null	C04	null	Objective-C	1
			L'application doit être en Objective-C sous la plateforme iOS 7 et est destinée aux iPod Touch de 5e génération avec un écran de 4 pouces	
null	C07	null	Id et offset des capteurs	2
			Les « id » et les « offset » des capteurs doivent suivre la table CAN fournie.	
null	C08	null	langue	1
			Les textes de l'application doivent être en anglais.	
null	C09	null	deployment	4
			L'application doit être installée et exécutée sur un iPod Touch qui ne nécessite pas un « iOS jailbreaking »	
null	EF02	Générale	Configuration de l'application dans les paramètres d'iOS	2
			Le système doit permettre de changer quelques configurations directement dans les paramètres de l'application sur iOS. Les configurations doivent inclure, entre autres, le changement de mode entre pilote et ingénieur de piste ainsi que le changement des couleurs de l'interface de pâle à foncé.	
null	EF03	Générale	Gérer les données reçues en temps réel	9
			L'application doit constamment recevoir des données du bus CAN via Wi-Fi à partir du module Can2Ethernet et, en comparant avec la table des messages CAN, associer ces données aux capteurs et alarmes pour les afficher.	
null	EF04	Générale	Afficher des couleurs spécifiques pour les RPM	2
			L'application doit afficher l'indicateur de RPM avec un code de couleur précis, soit de jaune à rouge en passant par une zone orange visible. De 3000 RPM à 15 000 RPM, l'indicateur doit être dans le spectre de jaune à rouge. À 15 000 RPM et plus, l'indicateur doit être rouge. De plus, le rouge doit changer selon un paramètre calculé par l'ACL lorsqu'un message correspondant au « id » de la table CAN de ce paramètre est reçu.	
null	EF05	Générale	Afficher des couleurs spécifiques pour la température des pneus	2
			L'application doit afficher la température des trois capteurs de chaque pneu selon des couleurs spécifiques et avec des transitions fluides. Les capteurs sont situés à l'extérieur, au milieu et à l'intérieur de chacun des pneus. Lorsque la température est de 25°C et moins, la couleur est bleue. Entre 25°C et 65°C, la couleur passe de bleue à jaune. Entre 65°C et 95°C, l'indicateur passe de jaune à rouge. Finalement, en haut de 95°C, la couleur est rouge.	
null	EF06	Générale	Afficher des couleurs spécifiques pour la température du moteur	2
			Le Dash Display doit afficher la température du moteur à l'aide de quatre couleurs. Lorsque la température est de 70°C et moins, l'indicateur doit être bleu. Entre 70°C et 90°C, la couleur utilisée est le vert. Entre 90°C et 100°C, l'indicateur doit être jaune et, finalement, il doit être rouge lorsque la température dépasse le 100°C.	
null	EF07	Générale	Afficher des couleurs spécifiques pour la puissance de la batterie	2
			Le système doit afficher la puissance de la batterie à l'aide de deux couleurs. Lorsque la batterie est à la puissance maximale (14V) jusqu'à 11.5V, le fond du capteur est vert. Lorsque la puissance atteint 11.5 V et en dessous, le fond devient rouge.	
null	EF08	Générale	Afficher des couleurs spécifiques pour les alertes	2
			Le Dash Display doit afficher les alertes de deux façons selon leur statut. Dans tous les cas, elles sont affichées en rouge. Lorsqu'elles sont en cours, elles ont une opacité de 100 %. Sinon, l'opacité diminue à 30 %.	
null	EF09	Mode pilote	Visualiser les alarmes et les capteurs sur l'interface pilote	6
			L'application doit afficher, de manière claire, les informations nécessaires au pilote en provenance des différents capteurs du véhicule. De plus, les messages d'alarmes seront aussi affichés de façon évidente et de façon à ce que le pilote les remarques immédiatement.	
null	EF15	Mode pilote	Affichage en mode paysage pour le mode pilote	1
			doit afficher les interfaces en mode pilote sous le format paysage.	
null	EF16	Mode pilote	Mettre en veille l'application après 3 secondes sans données	1
			doit se mettre en veille si une interruption de données survient et si elle dure plus de trois secondes.	
null	ENF01	Usability	Utilisation du visuel de façon intuitive	6
			L'interface doit respecter le fonctionnement natif d'iOS lorsque les diverses actions sont	

		effectuées dans le mode ingénieur de piste. Par exemple, l'ingénieur de piste doit entrer en mode édition pour supprimer ou déplacer une alarme ou un capteur.	
null	ENF02 Usability	Utilisation du mode pilote doit être très simple Les différentes interfaces du mode pilote doivent être simples, claires et précises. Lorsque la voiture est en piste, le pilote ne doit pas avoir à réfléchir pour comprendre et utiliser l'application. Les quatre interfaces disponibles doivent être toutes visibles en trois clics du bouton situé sur le volant puis continuer de cette façon en boucle.	2
null	ENF03 Usability	Démarrage simple et rapide dans le mode configuré Lorsque le Dash Display démarre, il ne doit pas y avoir d'attente ou de commande à effectuer pour que l'application puisse être utilisée. Cette exigence est surtout importante pour le mode pilote qui ne peut pas utiliser l'écran tactile. De plus, le système doit utiliser le mode choisi dans les configurations de l'application directement dans iOS.	4
null	ENF04 Usability	Haut contraste dans les couleurs de l'interface Les couleurs de l'interface doivent avoir de très haut contraste. De plus, deux versions de couleurs doivent être disponibles dans la configuration de l'application sur iOS : un mode foncé et un mode pâle.	1
null	ENF05 Performance	Rafraîchissement de l'écran à une cadence de 10 Hz La fréquence de rafraîchissement des interfaces est de 10 Hz afin que les données affichées soient toujours à jour en temps réel. Avec cette cadence, le véhicule a le temps de transmettre les nouvelles données par Wi-Fi. Cette mesure signifie 10 fois par seconde donc le rafraîchissement est d'une fois à chaque 100 ms.	4
null	ENF06 Disponibility	Période d'utilisation d'au maximum 25 minutes Les courses ont une durée d'au maximum 25 minutes donc l'application doit être optimisée pour une utilisation sans problème pour ce délai de temps.	1
null	ENF08 Disponibility	Aucun redémarrage de l'application en cas d'erreur Lorsqu'une erreur survient, l'application ne doit pas redémarrer seule. Il faut la redémarrer manuellement à chaque fois.	9

0.7.1 Perspectives

EF02 - Configuration de l'application dans les paramètres d'iOS Le système doit permettre de changer quelques configurations directement dans les paramètres de l'application sur iOS. Les configurations doivent inclure, entre autres, le changement de mode entre pilote et ingénieur de piste ainsi que le changement des couleurs de l'interface de pâle à foncé.

C02 - Usability Les configurations pour le mode par défaut et les couleurs de l'interface sont définies directement dans iOS.

EF03 - Gérer les données reçues en temps réel L'application doit constamment recevoir des données du bus CAN via Wi-Fi à partir du module Can2Ethernet et, en comparant avec la table des messages CAN, associer ces données aux capteurs et alarmes pour les afficher.

EF04 - Afficher des couleurs spécifiques pour les RPM L'application doit afficher l'indicateur de RPM avec un code de couleur précis, soit de jaune à rouge en passant par une zone orange visible. De 3000 RPM à 15 000 RPM, l'indicateur doit être dans le spectre de jaune à rouge. À 15 000 RPM et plus, l'indicateur doit être rouge. De plus, le rouge doit changer selon un paramètre calculé par l'ACL lorsqu'un message correspondant au « id » de la table CAN de ce paramètre est reçu.

EF15 - Affichage en mode paysage pour le mode pilote doit afficher les interfaces en mode pilote sous le format paysage.

C04 - Objective-C L'application doit être en Objective-C sous la plateforme iOS 7 et est destinée aux iPod Touch de 5e génération avec un écran de 4 pouces

EF05 - Afficher des couleurs spécifiques pour la température des pneus L'application doit afficher la température des trois capteurs de chaque pneu selon des couleurs spécifiques et avec des transitions fluides. Les capteurs sont situés à l'extérieur, au milieu et à l'intérieur de chacun des pneus. Lorsque la température est de 25°C et moins, la couleur est bleue. Entre 25°C et 65°C, la couleur passe de bleue à jaune. Entre 65°C et 95°C, l'indicateur passe de jaune à rouge. Finalement, en haut de 95°C, la couleur est rouge.

EF16 - Mettre en veille l'application après 3 secondes sans données doit se mettre en veille si une interruption de données survient et si elle dure plus de trois secondes.

EF06 - Afficher des couleurs spécifiques pour la température du moteur Le Dash Display doit afficher la température du moteur à l'aide de quatre couleurs. Lorsque la température est de 70°C et moins, l'indicateur doit être bleu. Entre 70°C et 90°C, la couleur utilisée est le vert. Entre 90°C et 100°C, l'indicateur doit être jaune et, finalement, il doit être rouge lorsque la température dépasse le 100°C.

EF07 - Afficher des couleurs spécifiques pour la puissance de la batterie Le système doit afficher la puissance de la batterie à l'aide de deux couleurs. Lorsque la batterie est à la puissance maximale (14V) jusqu'à 11.5V, le fond du capteur est vert. Lorsque la puissance atteint 11.5 V et en dessous, le fond devient rouge.

EF08 - Afficher des couleurs spécifiques pour les alertes Le Dash Display doit afficher les alertes de deux façons selon leur statut. Dans tous les cas, elles sont affichées en rouge. Lorsqu'elles sont en cours, elles ont une opacité de 100 %. Sinon, l'opacité diminue à 30 %.

C08 - langue Les textes de l'application doivent être en anglais.

EF09 - Visualiser les alarmes et les capteurs sur l'interface pilote L'application doit afficher, de manière claire, les informations nécessaires au pilote en provenance des différents capteurs du véhicule. De plus, les messages d'alarmes seront aussi affichés de façon évidente et de façon à ce que le pilote les remarques immédiatement.

C07 - Id et offset des capteurs Les « id » et les « offset » des capteurs doivent suivre la table CAN fournie.

C09 - deployment L'application doit être installée et exécutée sur un iPod Touch qui ne nécessite pas un « iOS jailbreaking »

ENF01 - Utilisation du visuel de façon intuitive L'interface doit respecter le fonctionnement natif d'iOS lorsque les diverses actions sont effectuées dans le mode ingénieur de piste. Par exemple, l'ingénieur de piste doit entrer en mode édition pour supprimer ou déplacer une alarme ou un capteur.

ENF02 - Utilisation du mode pilote doit être très simple Les différentes interfaces du mode pilote doivent être simples, claires et précises. Lorsque la voiture est en piste, le pilote ne doit pas avoir à réfléchir pour comprendre et utiliser l'application. Les quatre interfaces disponibles doivent être toutes visibles en trois clics du bouton situé sur le volant puis continuer de cette façon en boucle.

ENF03 - Démarrage simple et rapide dans le mode configuré Lorsque le Dash Display démarre, il ne doit pas y avoir d'attente ou de commande à effectuer pour que l'application puisse être utilisée. Cette exigence est surtout importante pour le mode pilote qui ne peut pas utiliser l'écran tactile. De plus, le système doit utiliser le mode choisi dans les configurations de l'application directement dans iOS.

ENF04 - Haut contraste dans les couleurs de l'interface Les couleurs de l'interface doivent avoir de très haut contraste. De plus, deux versions de couleurs doivent être disponibles dans la configuration de l'application sur iOS : un mode foncé et un mode pâle.

ENF05 - Rafraîchissement de l'écran à une cadence de 10 Hz La fréquence de rafraîchissement des interfaces est de 10 Hz afin que les données affichées soient toujours à jour en temps réel. Avec cette cadence, le véhicule a le temps de transmettre les nouvelles données par Wi-Fi. Cette mesure signifie 10 fois par seconde donc le rafraîchissement est d'une fois à chaque 100 ms.

ENF06 - Période d'utilisation d'au maximum 25 minutes Les courses ont une durée d'au maximum 25 minutes donc l'application doit être optimisée pour une utilisation sans problème pour ce délai de temps.

ENF08 - Aucun redémarrage de l'application en cas d'erreur Lorsqu'une erreur survient, l'application ne doit pas redémarrer seule. Il faut la redémarrer manuellement à chaque fois.

0.8 Pilot

Le pilote contrôle le véhicule lors des essais sur piste et des compétitions. Il utilise l'application en mode pilote afin d'accéder aux données du véhicule ce qui permet d'avoir une meilleure compréhension des différents composants et d'améliorer sa conduite.

Tags: Element,Person

Parent	Key	Category	Title	Priority
null	EF02	Générale	Configuration de l'application dans les paramètres d'iOS	2
			Le système doit permettre de changer quelques configurations directement dans les paramètres de l'application sur iOS. Les configurations doivent inclure, entre autres, le changement de mode entre pilote et ingénieur de piste ainsi que le changement des couleurs de l'interface de pâle à foncé.	
null	EF09	Mode pilote	Visualiser les alarmes et les capteurs sur l'interface pilote	6
			L'application doit afficher, de manière claire, les informations nécessaires au pilote en provenance des différents capteurs du véhicule. De plus, les messages d'alarmes seront aussi affichés de façon évidente et de façon à ce que le pilote les remarques immédiatement.	
null	EF10	Mode pilote	Changer l'interface lors de l'appui sur le bouton du volant	4
			L'application doit changer l'interface affichée lorsque le pilote appuie sur un bouton du volant. Le bouton envoie un message CAN à l'application pour lui indiquer de changer. L'application en mode	

			pilote affichera quatre interfaces différentes en boucle.	
null	EF11	Mode pilote	Contenu de la première interface	4
			L'application doit afficher, sur la première interface, les capteurs suivants : la température du moteur, le voltage de la batterie, l'indicateur d'utilisation et d'angle du système de réduction de traînée, les différentes alarmes, la boîte de messages, l'indicateur de vitesse et l'indicateur de révolutions par minute.	
null	EF12	Mode pilote	Contenu de la deuxième interface	4
			Le Dash Display doit afficher, sur la deuxième interface, les capteurs suivants : la pression et la température des pneus, le biais de frein, l'antiroulis, l'odomètre, le voltage de la batterie, la température du moteur et un indicateur d'utilisation et l'angle du système de réduction de traînée.	
null	EF13	Mode pilote	Contenu de la troisième interface	4
			L'application doit afficher, sur la troisième interface, le temps total de course, le temps du tour courant, la différence de temps par rapport au meilleur temps, le meilleur temps de tour de piste et la différence de temps par rapport au dernier de tour de piste.	
null	EF14	Mode pilote	Contenu de la quatrième interface	4
			Dash Display doit afficher, sur la quatrième interface, un schéma de la piste de course et le déplacement de la voiture en temps réel. De plus, les temps suivants doivent être présents : le temps du tour de piste, le meilleur temps et la différence par rapport au meilleur temps.	
null	EF15	Mode pilote	Affichage en mode paysage pour le mode pilote	1
			doit afficher les interfaces en mode pilote sous le format paysage.	
null	EF16	Mode pilote	Mettre en veille l'application après 3 secondes sans données	1
			doit se mettre en veille si une interruption de données survient et si elle dure plus de trois secondes.	

0.8.1 Perspectives

EF10 - Changer l'interface lors de l'appui sur le bouton du volant L'application doit changer l'interface affichée lorsque le pilote appuie sur un bouton du volant. Le bouton envoie un message CAN à l'application pour lui indiquer de changer. L'application en mode pilote affichera quatre interfaces différentes en boucle.

EF11 - Le pilot doit pouvoir activer l'écran 1 a partir d'un bouton sur le volant

EF12 - Le pilot doit pouvoir activer l'écran 2 a partir d'un bouton sur le volant

EF02 - Le système doit permettre de changer quelques configurations directement dans les paramètres de l'application sur iOS. Les configurations doivent inclure, entre autres, le changement de mode entre pilote et ingénieur de piste ainsi que le changement des couleurs de l'interface de pâle à foncé.

EF13 - Le pilot doit pouvoir activer l'écran 3 a partir d'un bouton sur le volant

EF15 - Affichage en mode paysage pour le mode pilote

EF16 - Mettre en veille l'application après 3 secondes sans donnéesÉ

EF09 - Visualiser les alarmes et les capteurs sur l'interface pilote L'application doit afficher, de manière claire, les informations nécessaires au pilote en provenance des différents capteurs du véhicule. De plus, les messages d'alarmes seront aussi affichés de façon évidente et de façon à ce que le pilote les remarques immédiatement.

0.9 Engineer

L'ingénieur de piste gère les alarmes et capteurs du véhicule et ajuste/optimise les paramètres logiciels du véhicule

Tags: Element,Person

Parent	Key	Category	Title	Priority
null	EF01	Général	Configuration de l'application avec un fichier XML	4

			L'application doit utiliser un fichier de configuration, sous le format XML, pour déterminer les alarmes et capteurs disponibles. La liste des alarmes et des capteurs sont définis selon la table CAN fournie par la Formule ÉTS.	
null	EF17	Mode ingénieur de piste	Visualiser les alarmes et les capteurs sur l'interface ingénieur	6
			doit afficher, sous forme de liste, les différents capteurs et alarmes que l'utilisateur décide d'inclure. Les capteurs et les alarmes sont affichés séparément, les alarmes se trouvant en haut de la liste. Lors de la présentation du prototype, le club formule ÉTS a précisé qu'ils souhaiteraient traiter les alarmes de la même façon que les différents capteurs donc les rendre modifiable sur la page principale.	
null	EF18	Mode ingénieur de piste	Ajouter une alarme ou un capteur	2
			doit permettre à l'utilisateur de sélectionner dans une liste une alarme ou un capteur à ajouter à la liste d'affichage. L'application doit aussi permettre de filtrer la liste des alarmes et des capteurs qui peuvent être ajoutés et d'y effectuer une recherche.	
null	EF19	Mode ingénieur de piste	Changer l'ordre des alarmes et des capteurs affichés	4
			doit permettre à l'utilisateur, une fois en mode édition de la liste, de réorganiser respectivement les alarmes et les capteurs entre eux.	
null	EF20	Mode ingénieur de piste	Supprimer une alarme ou un capteur affiché	1
			doit permettre à l'utilisateur, une fois en mode édition de la liste, de supprimer un capteur ou une alarme.	
null	EF21	Mode ingénieur de piste	Afficher les détails de l'alarme ou du capteur	4
			système doit permettre de cliquer sur un capteur ou une alarme affichés afin d'obtenir plus de détails. Lors de la présentation du prototype, cette exigence a été clarifiée. Le client désire avoir la possibilité de modifier l'affichage du widget pour d'autres formats ainsi qu'obtenir un historique des dernières données. Les informations apparaissent sous le widget principal avec la possibilité d'afficher l'historique en plein écran.	
null	EF22	Mode ingénieur de piste	Gérer les cas d'erreurs de l'application	9
			doit, en cas d'erreurs de l'application, afficher les dernières données reçues. Les cas d'erreurs peuvent être, par exemple, une erreur de transmission de données ou un message d'erreur reçu par une chaîne CAN du module Can2Ethernet.	

0.9.1 Perspectives

EF21 - Afficher les détails de l'alarme ou du capteur

EF22 - Gérer les cas d'erreurs de l'application

EF01 - Configuration de l'application avec un fichier XML

C01 - L'ingénieur est responsable de réaliser le fichier de configureation XML

EF17 - Visualiser les alarmes et les capteurs sur l'interface ingénieur

EF18 - Ajouter une alarme ou un capteur doit permettre à l'utilisateur de sélectionner dans une liste une alarme ou un capteur à ajouter à la liste d'affichage. L'application doit aussi permettre de filtrer la liste des alarmes et des capteurs qui peuvent être ajoutés et d'y effectuer une recherche.

EF19 - Changer l'ordre des alarmes et des capteurs affichés

EF20 - Supprimer une alarme ou un capteur affiché

0.10 SystemRacing

Système de calcul sur site permettant de récupérer les données temps réel et d'envoyer des commandes aux véhicule pour la calibration de celui-ci.

Tags: Element,Software System

Parent	Key	Category	Title	Priority
null	C01	null	Usability	2
			Les configurations pour les catégories des alarmes et des capteurs doivent être définies dans un fichier XML.	
null	C04	null	Objective-C	1
			L'application doit être en Objective-C sous la plateforme iOS 7 et est destinée aux iPod Touch de 5e génération avec un écran de 4 pouces	
null	C06	null	P port	1
			Le port UDP est le port par défaut soit 1337.	
null	C08	null	langue	1
			Les textes de l'application doivent être en anglais.	
null	EF01	Général	Configuration de l'application avec un fichier XML	4
			L'application doit utiliser un fichier de configuration, sous le format XML, pour déterminer les alarmes et capteurs disponibles. La liste des alarmes et des capteurs sont définis selon la table CAN fournie par la Formule ÉTS.	
null	EF03	Générale	Gérer les données reçues en temps réel	9
			L'application doit constamment recevoir des données du bus CAN via Wi-Fi à partir du module Can2Ethernet et, en comparant avec la table des messages CAN, associer ces données aux capteurs et alarmes pour les afficher.	
null	EF17	Mode ingénieur de piste	Visualiser les alarmes et les capteurs sur l'interface ingénieur	6
			doit afficher, sous forme de liste, les différents capteurs et alarmes que l'utilisateur décide d'inclure. Les capteurs et les alarmes sont affichés séparément, les alarmes se trouvant en haut de la liste. Lors de la présentation du prototype, le club formule ÉTS a précisé qu'ils souhaiteraient traiter les alarmes de la même façon que les différents capteurs donc les rendre modifiable sur la page principale.	
null	EF18	Mode ingénieur de piste	Ajouter une alarme ou un capteur	2
			doit permettre à l'utilisateur de sélectionner dans une liste une alarme ou un capteur à ajouter à la liste d'affichage. L'application doit aussi permettre de filtrer la liste des alarmes et des capteurs qui peuvent être ajoutés et d'y effectuer une recherche.	
null	EF19	Mode ingénieur de piste	Changer l'ordre des alarmes et des capteurs affichés	4
			doit permettre à l'utilisateur, une fois en mode édition de la liste, de réorganiser respectivement les alarmes et les capteurs entre eux.	
null	EF20	Mode ingénieur de piste	Supprimer une alarme ou un capteur affiché	1
			doit permettre à l'utilisateur, une fois en mode édition de la liste, de supprimer un capteur ou une alarme.	
null	EF21	Mode ingénieur de piste	Afficher les détails de l'alarme ou du capteur	4
			système doit permettre de cliquer sur un capteur ou une alarme affichés afin d'obtenir plus de détails. Lors de la présentation du prototype, cette exigence a été clarifiée. Le client désire avoir la possibilité de modifier l'affichage du widget pour d'autres formats ainsi qu'obtenir un historique des dernières données. Les informations apparaissent sous le widget principal avec la possibilité d'afficher l'historique en plein écran.	

null	EF22	Mode ingénieur de piste	Gérer les cas d'erreurs de l'application	9
			doit, en cas d'erreurs de l'application, afficher les dernières données reçues. Les cas d'erreurs peuvent être, par exemple, une erreur de transmission de données ou un message d'erreur reçu par une chaîne CAN du module Can2Ethernet.	
null	ENF07	Usability	Modification rapide des alarmes et des capteurs	9
			La liste des alarmes et des capteurs peut facilement être modifiée en 5 minutes et moins. Cette modification est effectuée sur le fichier de configuration XML en fournissant le « id » et l'« offset » définis dans la table CAN. Cette exigence n'était pas précisée de la part du client, mais une précision concernant la table CAN qui peut être modifiée a été faite. Il est donc important que tout le logiciel soit facilement maintenable.	

0.10.1 Perspectives

EF21 - Afficher les détails de l'alarme ou du capteur système doit permettre de cliquer sur un capteur ou une alarme affichés afin d'obtenir plus de détails. Lors de la présentation du prototype, cette exigence a été clarifiée. Le client désire avoir la possibilité de modifier l'affichage du widget pour d'autres formats ainsi qu'obtenir un historique des dernières données. Les informations apparaissent sous le widget principal avec la possibilité d'afficher l'historique en plein écran.

EF22 - Gérer les cas d'erreurs de l'application doit, en cas d'erreurs de l'application, afficher les dernières données reçues. Les cas d'erreurs peuvent être, par exemple, une erreur de transmission de données ou un message d'erreur reçu par une chaîne CAN du module Can2Ethernet.

EF01 - Configuration de l'application avec un fichier XML

EF03 - Gérer les données reçues en temps réel L'application doit constamment recevoir des données du bus CAN via Wi-Fi à partir du module Can2Ethernet et, en comparant avec la table des messages CAN, associer ces données aux capteurs et alarmes pour les afficher.

C01 - Les configurations pour les catégories des alarmes et des capteurs doivent être définies dans un fichier XML.

C04 - L'application doit être en Objective-C sous la plateforme iOS 7 et est destinée aux iPod Touch de 5e génération avec un écran de 4 pouces

C03 - Recoit les données du véhicule et archives celles-ci

EF17 - Visualiser les alarmes et les capteurs sur l'interface ingénieur Doit afficher, sous forme de liste, les différents capteurs et alarmes que l'utilisateur décide d'inclure. Les capteurs et les alarmes sont affichés séparément, les alarmes se trouvant en haut de la liste. Lors de la présentation du prototype, le club formule ÉTS a précisé qu'ils souhaiteraient traiter les alarmes de la même façon que les différents capteurs donc les rendre modifiable sur la page principale.

C06 - Le port UDP est le port par défaut soit 1337.

EF19 - Changer l'ordre des alarmes et des capteurs affichés doit permettre à l'utilisateur, une fois en mode édition de la liste, de réorganiser respectivement les alarmes et les capteurs entre eux.

C08 - Les textes de l'application doivent être en anglais.

EF20 - Supprimer une alarme ou un capteur affiché doit permettre à l'utilisateur, une fois en mode édition de la liste, de supprimer un capteur ou une alarme.

0.11 SystemVehicule

Système déployé dans les véhicules FormuleETS pour permettre la communication avec le Racing Server et le pilote.

Tags: Element,Software System

Parent	Key	Category	Title	Priority
null	C02	null	Usability	2
			Les configurations pour le mode par défaut et les couleurs de l'interface sont définies directement dans iOS.	
null	C03	null	can2Ethernet	9

			La librairie Can2Ethernet développée par le club Formule ÉTS doit être utilisée.	
null	C04	null	Objective-C	1
			L'application doit être en Objective-C sous la plateforme iOS 7 et est destinée aux iPod Touch de 5e génération avec un écran de 4 pouces	
null	C05	Performance	fi UDP	4
			Les données sont envoyées par Wi-Fi via le protocole UDP et le bus CAN.	
null	C07	null	Id et offset des capteurs	2
			Les « id » et les « offset » des capteurs doivent suivre la table CAN fournie.	
null	C08	null	langue	1
			Les textes de l'application doivent être en anglais.	
null	C09	null	deployment	4
			L'application doit être installée et exécutée sur un iPod Touch qui ne nécessite pas un « iOS jailbreaking »	
null	EF02	Générale	Configuration de l'application dans les paramètres d'iOS	2
			Le système doit permettre de changer quelques configurations directement dans les paramètres de l'application sur iOS. Les configurations doivent inclure, entre autres, le changement de mode entre pilote et ingénieur de piste ainsi que le changement des couleurs de l'interface de pâle à foncé.	
null	EF03	Générale	Gérer les données reçues en temps réel	9
			L'application doit constamment recevoir des données du bus CAN via Wi-Fi à partir du module Can2Ethernet et, en comparant avec la table des messages CAN, associer ces données aux capteurs et alarmes pour les afficher.	
null	EF04	Générale	Afficher des couleurs spécifiques pour les RPM	2
			L'application doit afficher l'indicateur de RPM avec un code de couleur précis, soit de jaune à rouge en passant par une zone orange visible. De 3000 RPM à 15 000 RPM, l'indicateur doit être dans le spectre de jaune à rouge. À 15 000 RPM et plus, l'indicateur doit être rouge. De plus, le rouge doit changer selon un paramètre calculé par l'ACL lorsqu'un message correspondant au « id » de la table CAN de ce paramètre est reçu.	
null	EF05	Générale	Afficher des couleurs spécifiques pour la température des pneus	2
			L'application doit afficher la température des trois capteurs de chaque pneu selon des couleurs spécifiques et avec des transitions fluides. Les capteurs sont situés à l'extérieur, au milieu et à l'intérieur de chacun des pneus. Lorsque la température est de 25°C et moins, la couleur est bleue. Entre 25°C et 65°C, la couleur passe de bleue à jaune. Entre 65°C et 95°C, l'indicateur passe de jaune à rouge. Finalement, en haut de 95°C, la couleur est rouge.	
null	EF06	Générale	Afficher des couleurs spécifiques pour la température du moteur	2
			Le Dash Display doit afficher la température du moteur à l'aide de quatre couleurs. Lorsque la température est de 70°C et moins, l'indicateur doit être bleu. Entre 70°C et 90°C, la couleur utilisée est le vert. Entre 90°C et 100°C, l'indicateur doit être jaune et, finalement, il doit être rouge lorsque la température dépasse le 100°C.	
null	EF07	Générale	Afficher des couleurs spécifiques pour la puissance de la batterie	2
			Le système doit afficher la puissance de la batterie à l'aide de deux couleurs. Lorsque la batterie est à la puissance maximale (14V) jusqu'à 11.5V, le fond du capteur est vert. Lorsque la puissance atteint 11.5 V et en dessous, le fond devient rouge.	
null	EF08	Générale	Afficher des couleurs spécifiques pour les alertes	2
			Le Dash Display doit afficher les alertes de deux façons selon leur statut. Dans tous les cas, elles sont affichées en rouge. Lorsqu'elles sont en cours, elles ont une opacité de 100 %. Sinon, l'opacité diminue à 30 %.	
null	EF09	Mode pilote	Visualiser les alarmes et les capteurs sur l'interface pilote	6
			L'application doit afficher, de manière claire, les informations nécessaires au pilote en provenance des différents capteurs du véhicule. De plus, les messages d'alarmes seront aussi affichés de façon évidente et de façon à ce que le pilote les remarques immédiatement.	
null	EF10	Mode pilote	Changer l'interface lors de l'appui sur le bouton du volant	4
			L'application doit changer l'interface affichée lorsque le pilote appuie sur un bouton du volant. Le bouton envoie un message CAN à l'application pour lui indiquer de changer. L'application en mode pilote affichera quatre interfaces différentes en boucle.	
null	EF11	Mode pilote	Contenu de la première interface	4

			L’application doit afficher, sur la première interface, les capteurs suivants : la température du moteur, le voltage de la batterie, l’indicateur d’utilisation et d’angle du système de réduction de traînée, les différentes alarmes, la boîte de messages, l’indicateur de vitesse et l’indicateur de révolutions par minute.	
null	EF12	Mode pilote	Contenu de la deuxième interface Le Dash Display doit afficher, sur la deuxième interface, les capteurs suivants : la pression et la température des pneus, le biais de frein, l’antiroulis, l’odomètre, le voltage de la batterie, la température du moteur et un indicateur d’utilisation et l’angle du système de réduction de traînée.	4
null	EF13	Mode pilote	Contenu de la troisième interface L’application doit afficher, sur la troisième interface, le temps total de course, le temps du tour courant, la différence de temps par rapport au meilleur temps, le meilleur temps de tour de piste et la différence de temps par rapport au dernier de tour de piste.	4
null	EF14	Mode pilote	Contenu de la quatrième interface Dash Display doit afficher, sur la quatrième interface, un schéma de la piste de course et le déplacement de la voiture en temps réel. De plus, les temps suivants doivent être présents : le temps du tour de piste, le meilleur temps et la différence par rapport au meilleur temps.	4
null	EF15	Mode pilote	Affichage en mode paysage pour le mode pilote doit afficher les interfaces en mode pilote sous le format paysage.	1
null	EF16	Mode pilote	Mettre en veille l’application après 3 secondes sans données doit se mettre en veille si une interruption de données survient et si elle dure plus de trois secondes.	1
null	ENF01	Usability	Utilisation du visuel de façon intuitive L’interface doit respecter le fonctionnement natif d’iOS lorsque les diverses actions sont effectuées dans le mode ingénieur de piste. Par exemple, l’ingénieur de piste doit entrer en mode édition pour supprimer ou déplacer une alarme ou un capteur.	6
null	ENF02	Usability	Utilisation du mode pilote doit être très simple Les différentes interfaces du mode pilote doivent être simples, claires et précises. Lorsque la voiture est en piste, le pilote ne doit pas avoir à réfléchir pour comprendre et utiliser l’application. Les quatre interfaces disponibles doivent être toutes visibles en trois clics du bouton situé sur le volant puis continuer de cette façon en boucle.	2
null	ENF03	Usability	Démarrage simple et rapide dans le mode configuré Lorsque le Dash Display démarre, il ne doit pas y avoir d’attente ou de commande à effectuer pour que l’application puisse être utilisée. Cette exigence est surtout importante pour le mode pilote qui ne peut pas utiliser l’écran tactile. De plus, le système doit utiliser le mode choisi dans les configurations de l’application directement dans iOS.	4
null	ENF04	Usability	Haut contraste dans les couleurs de l’interface Les couleurs de l’interface doivent avoir de très haut contraste. De plus, deux versions de couleurs doivent être disponibles dans la configuration de l’application sur iOS : un mode foncé et un mode pâle.	1
null	ENF05	Performance	Rafraîchissement de l’écran à une cadence de 10 Hz La fréquence de rafraîchissement des interfaces est de 10 Hz afin que les données affichées soient toujours à jour en temps réel. Avec cette cadence, le véhicule a le temps de transmettre les nouvelles données par Wi-Fi. Cette mesure signifie 10 fois par seconde donc le rafraîchissement est d’une fois à chaque 100 ms.	4
null	ENF06	Disponibility	Période d’utilisation d’au maximum 25 minutes Les courses ont une durée d’au maximum 25 minutes donc l’application doit être optimisée pour une utilisation sans problème pour ce délai de temps.	1
null	ENF08	Disponibility	Aucun redémarrage de l’application en cas d’erreur Lorsqu’une erreur survient, l’application ne doit pas redémarrer seule. Il faut la redémarrer manuellement à chaque fois.	9

0.11.1 Perspectives

EF02 - Configuration de l’application dans les paramètres d’iOS Le système doit permettre de changer quelques configurations directement dans les paramètres de l’application sur iOS. Les configurations doivent inclure, entre autres, le changement de mode entre pilote et ingénieur de piste ainsi que le changement des couleurs de l’interface de pâle à foncé.

C02 - Usability Les configurations pour le mode par défaut et les couleurs de l'interface sont définies directement dans iOS.

EF03 - Gérer les données reçues en temps réel L'application doit constamment recevoir des données du bus CAN via Wi-Fi à partir du module Can2Ethernet et, en comparant avec la table des messages CAN, associer ces données aux capteurs et alarmes pour les afficher.

EF04 - Afficher des couleurs spécifiques pour les RPM L'application doit afficher l'indicateur de RPM avec un code de couleur précis, soit de jaune à rouge en passant par une zone orange visible. De 3000 RPM à 15 000 RPM, l'indicateur doit être dans le spectre de jaune à rouge. À 15 000 RPM et plus, l'indicateur doit être rouge. De plus, le rouge doit changer selon un paramètre calculé par l'ACL lorsqu'un message correspondant au « id » de la table CAN de ce paramètre est reçu.

C04 - Objective-C L'application doit être en Objective-C sous la plateforme iOS 7 et est destinée aux iPod Touch de 5e génération avec un écran de 4 pouces

EF05 - Afficher des couleurs spécifiques pour la température des pneus L'application doit afficher la température des trois capteurs de chaque pneu selon des couleurs spécifiques et avec des transitions fluides. Les capteurs sont situés à l'extérieur, au milieu et à l'intérieur de chacun des pneus. Lorsque la température est de 25°C et moins, la couleur est bleue. Entre 25°C et 65°C, la couleur passe de bleue à jaune. Entre 65°C et 95°C, l'indicateur passe de jaune à rouge. Finalement, en haut de 95°C, la couleur est rouge.

C03 - can2Ethernet La librairie Can2Ethernet développée par le club Formule ÉTS doit être utilisée.

EF06 - Afficher des couleurs spécifiques pour la température du moteur Le Dash Display doit afficher la température du moteur à l'aide de quatre couleurs. Lorsque la température est de 70°C et moins, l'indicateur doit être bleu. Entre 70°C et 90°C, la couleur utilisée est le vert. Entre 90°C et 100°C, l'indicateur doit être jaune et, finalement, il doit être rouge lorsque la température dépasse le 100°C.

EF07 - Afficher des couleurs spécifiques pour la puissance de la batterie Le système doit afficher la puissance de la batterie à l'aide de deux couleurs. Lorsque la batterie est à la puissance maximale (14V) jusqu'à 11.5V, le fond du capteur est vert. Lorsque la puissance atteint 11.5 V et en dessous, le fond devient rouge.

C05 - Wifi UDP Les données sont envoyées par Wi-Fi via le protocole UDP et le bus CAN.

EF08 - Afficher des couleurs spécifiques pour les alertes Le Dash Display doit afficher les alertes de deux façons selon leur statut. Dans tous les cas, elles sont affichées en rouge. Lorsqu'elles sont en cours, elles ont une opacité de 100 %. Sinon, l'opacité diminue à 30 %.

C08 - langue Les textes de l'application doivent être en anglais.

EF09 - Visualiser les alarmes et les capteurs sur l'interface pilote L'application doit afficher, de manière claire, les informations nécessaires au pilote en provenance des différents capteurs du véhicule. De plus, les messages d'alarmes seront aussi affichés de façon évidente et de façon à ce que le pilote les remarques immédiatement.

C07 - Id et offset des capteurs Les « id » et les « offset » des capteurs doivent suivre la table CAN fournie.

C09 - deployment L'application doit être installée et exécutée sur un iPod Touch qui ne nécessite pas un « iOS jailbreaking »

ENF01 - Utilisation du visuel de façon intuitive L'interface doit respecter le fonctionnement natif d'iOS lorsque les diverses actions sont effectuées dans le mode ingénieur de piste. Par exemple, l'ingénieur de piste doit entrer en mode édition pour supprimer ou déplacer une alarme ou un capteur.

ENF02 - Utilisation du mode pilote doit être très simple Les différentes interfaces du mode pilote doivent être simples, claires et précises. Lorsque la voiture est en piste, le pilote ne doit pas avoir à réfléchir pour comprendre et utiliser l'application. Les quatre interfaces disponibles doivent être toutes visibles en trois clics du bouton situé sur le volant puis continuer de cette façon en boucle.

EF10 - Changer l'interface lors de l'appui sur le bouton du volant L'application doit changer l'interface affichée lorsque le pilote appuie sur un bouton du volant. Le bouton envoie un message CAN à l'application pour lui indiquer de changer. L'application en mode pilote affichera quatre interfaces différentes en boucle.

EF11 - Contenu de la première interface L'application doit afficher, sur la première interface, les capteurs suivants : la température du moteur, le voltage de la batterie, l'indicateur d'utilisation et d'angle du système de réduction de traînée, les différentes alarmes, la boîte de messages, l'indicateur de vitesse et l'indicateur de révolutions par minute.

EF12 - Contenu de la deuxième interface Le Dash Display doit afficher, sur la deuxième interface, les capteurs suivants : la pression et la température des pneus, le biais de frein, l'antiroulis, l'odomètre, le voltage de la batterie, la température du moteur et un indicateur d'utilisation et l'angle du système de réduction de traînée.

EF13 - Contenu de la troisième interface L'application doit afficher, sur la troisième interface, le temps total de course, le temps du tour

courant, la différence de temps par rapport au meilleur temps, le meilleur temps de tour de piste et la différence de temps par rapport au dernier de tour de piste.

EF14 - Contenu de la quatrième interface Dash Display doit afficher, sur la quatrième interface, un schéma de la piste de course et le déplacement de la voiture en temps réel. De plus, les temps suivants doivent être présents : le temps du tour de piste, le meilleur temps et la différence par rapport au meilleur temps.

EF15 - Affichage en mode paysage pour le mode pilote doit afficher les interfaces en mode pilote sous le format paysage.

EF16 - Mettre en veille l'application après 3 secondes sans données doit se mettre en veille si une interruption de données survient et si elle dure plus de trois secondes.

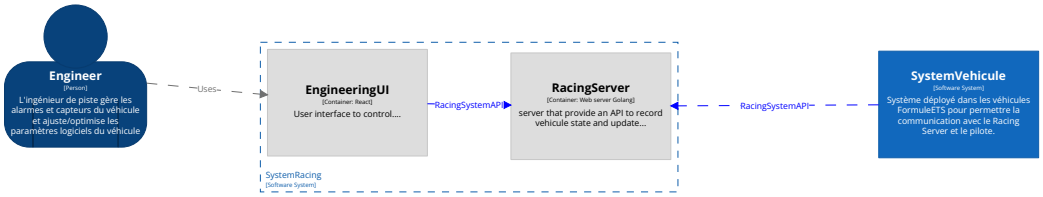
ENF03 - Démarrage simple et rapide dans le mode configuré Lorsque le Dash Display démarre, il ne doit pas y avoir d'attente ou de commande à effectuer pour que l'application puisse être utilisée. Cette exigence est surtout importante pour le mode pilote qui ne peut pas utiliser l'écran tactile. De plus, le système doit utiliser le mode choisi dans les configurations de l'application directement dans iOS.

ENF04 - Haut contraste dans les couleurs de l'interface Les couleurs de l'interface doivent avoir de très haut contraste. De plus, deux versions de couleurs doivent être disponibles dans la configuration de l'application sur iOS : un mode foncé et un mode pâle.

ENF05 - Rafraîchissement de l'écran à une cadence de 10 Hz La fréquence de rafraîchissement des interfaces est de 10 Hz afin que les données affichées soient toujours à jour en temps réel. Avec cette cadence, le véhicule a le temps de transmettre les nouvelles données par Wi-Fi. Cette mesure signifie 10 fois par seconde donc le rafraîchissement est d'une fois à chaque 100 ms.

ENF06 - Période d'utilisation d'au maximum 25 minutes Les courses ont une durée d'au maximum 25 minutes donc l'application doit être optimisée pour une utilisation sans problème pour ce délai de temps.

ENF08 - Aucun redémarrage de l'application en cas d'erreur Lorsqu'une erreur survient, l'application ne doit pas redémarrer seule. Il faut la redémarrer manuellement à chaque fois.



[Container] SystemRacing
Diagramme de décomposition du système Racing
Tuesday, November 15, 2022 at 1:11 PM Eastern Standard Time

0.12 Table des éléments du conteneur véhicule

0.13 Engineer

L'ingénieur de piste gère les alarmes et capteurs du véhicule et ajuste/optimise les paramètres logiciels du véhicule

Tags: Element,Person

Parent	Key	Category	Title	Priority
null	EF01	Général	Configuration de l’application avec un fichier XML	4
			L’application doit utiliser un fichier de configuration, sous le format XML, pour déterminer les alarmes et capteurs disponibles. La liste des alarmes et des capteurs sont définis selon la table CAN fournie par la Formule ÉTS.	
null	EF17	Mode ingénieur de piste	Visualiser les alarmes et les capteurs sur l’interface ingénieur	6
			doit afficher, sous forme de liste, les différents capteurs et alarmes que l’utilisateur décide d’inclure. Les capteurs et les alarmes sont affichés séparément, les alarmes se trouvant en haut de la liste. Lors de la présentation du prototype, le club formule ÉTS a précisé qu’ils souhaiteraient traiter les alarmes de la même façon que les différents capteurs donc les rendre modifiable sur la page principale.	
null	EF18	Mode ingénieur de piste	Ajouter une alarme ou un capteur	2
			doit permettre à l’utilisateur de sélectionner dans une liste une alarme ou un capteur à ajouter à la liste d’affichage. L’application doit aussi permettre de filtrer la liste des alarmes et des capteurs qui peuvent être ajoutés et d’y effectuer une recherche.	
null	EF19	Mode ingénieur de piste	Changer l’ordre des alarmes et des capteurs affichés	4
			doit permettre à l’utilisateur, une fois en mode édition de la liste, de réorganiser respectivement les alarmes et les capteurs entre eux.	
null	EF20	Mode ingénieur de piste	Supprimer une alarme ou un capteur affiché	1
			doit permettre à l’utilisateur, une fois en mode édition de la liste, de supprimer un capteur ou une alarme.	
null	EF21	Mode ingénieur de piste	Afficher les détails de l’alarme ou du capteur	4
			système doit permettre de cliquer sur un capteur ou une alarme affichés afin d’obtenir plus de détails. Lors de la présentation du prototype, cette exigence a été clarifiée. Le client désire avoir la possibilité de modifier l’affichage du widget pour d’autres formats ainsi qu’obtenir un historique des dernières données. Les informations apparaissent sous le widget principal avec la possibilité d’afficher l’historique en plein écran.	
null	EF22	Mode ingénieur de piste	Gérer les cas d’erreurs de l’application	9
			doit, en cas d’erreurs de l’application, afficher les dernières données reçues. Les cas d’erreurs peuvent être, par exemple, une erreur de transmission de données ou un message d’erreur reçu par une chaîne CAN du module Can2Ethernet.	

0.13.1 Perspectives

EF21 - Afficher les détails de l’alarme ou du capteur

EF22 - Gérer les cas d’erreurs de l’application

EF01 - Configuration de l’application avec un fichier XML

C01 - L'ingénieur est responsable de réaliser le fichier de configureation XML

EF17 - Visualiser les alarmes et les capteurs sur l’interface ingénieur

EF18 - Ajouter une alarme ou un capteur doit permettre à l’utilisateur de sélectionner dans une liste une alarme ou un capteur à ajouter à la liste d’affichage. L’application doit aussi permettre de filtrer la liste des alarmes et des capteurs qui peuvent être ajoutés et d’y effectuer une recherche.

EF19 - Changer l’ordre des alarmes et des capteurs affichés

EF20 - Supprimer une alarme ou un capteur affiché

0.14 EngineeringUI

User interface to control....

Tags: Element,Container

Parent	Key	Category	Title	Priority
null	C04	null	Objective-C	1
			L’application doit être en Objective-C sous la plateforme iOS 7 et est destinée aux iPod Touch de 5e génération avec un écran de 4 pouces	
null	C08	null	langue	1
			Les textes de l’application doivent être en anglais.	
null	EF17	Mode ingénieur de piste	Visualiser les alarmes et les capteurs sur l’interface ingénieur	6
			doit afficher, sous forme de liste, les différents capteurs et alarmes que l’utilisateur décide d’inclure. Les capteurs et les alarmes sont affichés séparément, les alarmes se trouvant en haut de la liste. Lors de la présentation du prototype, le club formule ÉTS a précisé qu’ils souhaiteraient traiter les alarmes de la même façon que les différents capteurs donc les rendre modifiable sur la page principale.	
null	EF18	Mode ingénieur de piste	Ajouter une alarme ou un capteur	2
			doit permettre à l’utilisateur de sélectionner dans une liste une alarme ou un capteur à ajouter à la liste d’affichage. L’application doit aussi permettre de filtrer la liste des alarmes et des capteurs qui peuvent être ajoutés et d’y effectuer une recherche.	
null	EF19	Mode ingénieur de piste	Changer l’ordre des alarmes et des capteurs affichés	4
			doit permettre à l’utilisateur, une fois en mode édition de la liste, de réorganiser respectivement les alarmes et les capteurs entre eux.	
null	EF20	Mode ingénieur de piste	Supprimer une alarme ou un capteur affiché	1
			doit permettre à l’utilisateur, une fois en mode édition de la liste, de supprimer un capteur ou une alarme.	
null	EF21	Mode ingénieur de piste	Afficher les détails de l’alarme ou du capteur	4
			système doit permettre de cliquer sur un capteur ou une alarme affichés afin d’obtenir plus de détails. Lors de la présentation du prototype, cette exigence a été clarifiée. Le client désire avoir la possibilité de modifier l’affichage du widget pour d’autres formats ainsi qu’obtenir un historique des dernières données. Les informations apparaissent sous le widget principal avec la possibilité d’afficher l’historique en plein écran.	
null	EF22	Mode ingénieur de piste	Gérer les cas d’erreurs de l’application	9
			doit, en cas d’erreurs de l’application, afficher les dernières données reçues. Les cas d’erreurs peuvent être, par exemple, une erreur de transmission de données ou un message d’erreur reçu par une chaîne CAN du module Can2Ethernet.	

0.14.1 Perspectives

EF21 - Afficher les détails de l’alarme ou du capteur système doit permettre de cliquer sur un capteur ou une alarme affichés afin d’obtenir plus de détails. Lors de la présentation du prototype, cette exigence a été clarifiée. Le client désire avoir la possibilité de modifier l’affichage du widget pour d’autres formats ainsi qu’obtenir un historique des dernières données. Les informations

apparaissent sous le widget principal avec la possibilité d'afficher l'historique en plein écran.

EF22 - Gérer les cas d'erreurs de l'application doit, en cas d'erreurs de l'application, afficher les dernières données reçues. Les cas d'erreurs peuvent être, par exemple, une erreur de transmission de données ou un message d'erreur reçu par une chaîne CAN du module Can2Ethernet.

C04 - Objective-C L'application doit être en Objective-C sous la plateforme iOS 7 et est destinée aux iPod Touch de 5e génération avec un écran de 4 pouces

EF17 - Visualiser les alarmes et les capteurs sur l'interface ingénieur doit afficher, sous forme de liste, les différents capteurs et alarmes que l'utilisateur décide d'inclure. Les capteurs et les alarmes sont affichés séparément, les alarmes se trouvant en haut de la liste. Lors de la présentation du prototype, le club formule ÉTS a précisé qu'ils souhaiteraient traiter les alarmes de la même façon que les différents capteurs donc les rendre modifiable sur la page principale.

EF18 - Ajouter une alarme ou un capteur doit permettre à l'utilisateur de sélectionner dans une liste une alarme ou un capteur à ajouter à la liste d'affichage. L'application doit aussi permettre de filtrer la liste des alarmes et des capteurs qui peuvent être ajoutés et d'y effectuer une recherche.

EF19 - Changer l'ordre des alarmes et des capteurs affichés doit permettre à l'utilisateur, une fois en mode édition de la liste, de réorganiser respectivement les alarmes et les capteurs entre eux.

C08 - langue Les textes de l'application doivent être en anglais.

EF20 - Supprimer une alarme ou un capteur affiché doit permettre à l'utilisateur, une fois en mode édition de la liste, de supprimer un capteur ou une alarme.

0.15 RacingServer

server that provide an API to record vehicule state and update...

Tags: Element,Container

Parent	Key	Category	Title	Priority
null	C01	null	Usability	2
			Les configurations pour les catégories des alarmes et des capteurs doivent être définies dans un fichier XML.	
null	C06	null	P port	1
			Le port UDP est le port par défaut soit 1337.	
null	EF01	Général	Configuration de l'application avec un fichier XML	4
			L'application doit utiliser un fichier de configuration, sous le format XML, pour déterminer les alarmes et capteurs disponibles. La liste des alarmes et des capteurs sont définis selon la table CAN fournie par la Formule ÉTS.	
null	EF03	Générale	Gérer les données reçues en temps réel	9
			L'application doit constamment recevoir des données du bus CAN via Wi-Fi à partir du module Can2Ethernet et, en comparant avec la table des messages CAN, associer ces données aux capteurs et alarmes pour les afficher.	
null	ENF07	Usability	Modification rapide des alarmes et des capteurs	9
			La liste des alarmes et des capteurs peut facilement être modifiée en 5 minutes et moins. Cette modification est effectuée sur le fichier de configuration XML en fournissant le « id » et l'« offset » définis dans la table CAN. Cette exigence n'était pas précisée de la part du client, mais une précision concernant la table CAN qui peut être modifiée a été faite. Il est donc important que tout le logiciel soit facilement maintenable.	

0.15.1 Perspectives

EF01 - Configuration de l'application avec un fichier XML L'application doit utiliser un fichier de configuration, sous le format XML, pour déterminer les alarmes et capteurs disponibles. La liste des alarmes et des capteurs sont définis selon la table CAN fournie par la Formule ÉTS.

EF03 - Gérer les données reçues en temps réel L'application doit constamment recevoir des données du bus CAN via Wi-Fi à partir du module Can2Ethernet et, en comparant avec la table des messages CAN, associer ces données aux capteurs et alarmes pour les

afficher.

C01 - Usability Les configurations pour les catégories des alarmes et des capteurs doivent être définies dans un fichier XML.

C03 - Recoit les données du véhicule

C06 - UDP port Le port UDP est le port par défaut soit 1337.

0.16 SystemVehicule

Système déployé dans les véhicules FormuleETS pour permettre la communication avec le Racing Server et le pilote.

Tags: Element,Software System

Parent	Key	Category	Title	Priority
null	C02	null	Usability	2
			Les configurations pour le mode par défaut et les couleurs de l'interface sont définies directement dans iOS.	
null	C03	null	can2Ethernet	9
			La librairie Can2Ethernet développée par le club Formule ÉTS doit être utilisée.	
null	C04	null	Objective-C	1
			L'application doit être en Objective-C sous la plateforme iOS 7 et est destinée aux iPod Touch de 5e génération avec un écran de 4 pouces	
null	C05	Performance	fi UDP	4
			Les données sont envoyées par Wi-Fi via le protocole UDP et le bus CAN.	
null	C07	null	Id et offset des capteurs	2
			Les « id » et les « offset » des capteurs doivent suivre la table CAN fournie.	
null	C08	null	langue	1
			Les textes de l'application doivent être en anglais.	
null	C09	null	deployment	4
			L'application doit être installée et exécutée sur un iPod Touch qui ne nécessite pas un « iOS jailbreaking »	
null	EF02	Générale	Configuration de l'application dans les paramètres d'iOS	2
			Le système doit permettre de changer quelques configurations directement dans les paramètres de l'application sur iOS. Les configurations doivent inclure, entre autres, le changement de mode entre pilote et ingénieur de piste ainsi que le changement des couleurs de l'interface de pâle à foncé.	
null	EF03	Générale	Gérer les données reçues en temps réel	9
			L'application doit constamment recevoir des données du bus CAN via Wi-Fi à partir du module Can2Ethernet et, en comparant avec la table des messages CAN, associer ces données aux capteurs et alarmes pour les afficher.	
null	EF04	Générale	Afficher des couleurs spécifiques pour les RPM	2
			L'application doit afficher l'indicateur de RPM avec un code de couleur précis, soit de jaune à rouge en passant par une zone orange visible. De 3000 RPM à 15 000 RPM, l'indicateur doit être dans le spectre de jaune à rouge. À 15 000 RPM et plus, l'indicateur doit être rouge. De plus, le rouge doit changer selon un paramètre calculé par l'ACL lorsqu'un message correspondant au « id » de la table CAN de ce paramètre est reçu.	
null	EF05	Générale	Afficher des couleurs spécifiques pour la température des pneus	2
			L'application doit afficher la température des trois capteurs de chaque pneu selon des couleurs spécifiques et avec des transitions fluides. Les capteurs sont situés à l'extérieur, au milieu et à l'intérieur de chacun des pneus. Lorsque la température est de 25°C et moins, la couleur est bleue. Entre 25°C et 65°C, la couleur passe de bleue à jaune. Entre 65°C et 95°C, l'indicateur passe de jaune à rouge. Finalement, en haut de 95°C, la couleur est rouge.	
null	EF06	Générale	Afficher des couleurs spécifiques pour la température du moteur	2
			Le Dash Display doit afficher la température du moteur à l'aide de quatre couleurs. Lorsque la température est de 70°C et moins, l'indicateur doit être bleu. Entre 70°C et 90°C, la couleur utilisée est le vert. Entre 90°C et 100°C, l'indicateur doit être jaune et, finalement, il doit être rouge lorsque la température dépasse le 100°C.	

null	EF07	Générale	Afficher des couleurs spécifiques pour la puissance de la batterie Le système doit afficher la puissance de la batterie à l'aide de deux couleurs. Lorsque la batterie est à la puissance maximale (14V) jusqu'à 11.5V, le fond du capteur est vert. Lorsque la puissance atteint 11.5 V et en dessous, le fond devient rouge.	2
null	EF08	Générale	Afficher des couleurs spécifiques pour les alertes Le Dash Display doit afficher les alertes de deux façons selon leur statut. Dans tous les cas, elles sont affichées en rouge. Lorsqu'elles sont en cours, elles ont une opacité de 100 %. Sinon, l'opacité diminue à 30 %.	2
null	EF09	Mode pilote	Visualiser les alarmes et les capteurs sur l'interface pilote L'application doit afficher, de manière claire, les informations nécessaires au pilote en provenance des différents capteurs du véhicule. De plus, les messages d'alarmes seront aussi affichés de façon évidente et de façon à ce que le pilote les remarques immédiatement.	6
null	EF10	Mode pilote	Changer l'interface lors de l'appui sur le bouton du volant L'application doit changer l'interface affichée lorsque le pilote appuie sur un bouton du volant. Le bouton envoie un message CAN à l'application pour lui indiquer de changer. L'application en mode pilote affichera quatre interfaces différentes en boucle.	4
null	EF11	Mode pilote	Contenu de la première interface L'application doit afficher, sur la première interface, les capteurs suivants : la température du moteur, le voltage de la batterie, l'indicateur d'utilisation et d'angle du système de réduction de traînée, les différentes alarmes, la boîte de messages, l'indicateur de vitesse et l'indicateur de révolutions par minute.	4
null	EF12	Mode pilote	Contenu de la deuxième interface Le Dash Display doit afficher, sur la deuxième interface, les capteurs suivants : la pression et la température des pneus, le biais de frein, l'antiroulis, l'odomètre, le voltage de la batterie, la température du moteur et un indicateur d'utilisation et l'angle du système de réduction de traînée.	4
null	EF13	Mode pilote	Contenu de la troisième interface L'application doit afficher, sur la troisième interface, le temps total de course, le temps du tour courant, la différence de temps par rapport au meilleur temps, le meilleur temps de tour de piste et la différence de temps par rapport au dernier de tour de piste.	4
null	EF14	Mode pilote	Contenu de la quatrième interface Dash Display doit afficher, sur la quatrième interface, un schéma de la piste de course et le déplacement de la voiture en temps réel. De plus, les temps suivants doivent être présents : le temps du tour de piste, le meilleur temps et la différence par rapport au meilleur temps.	4
null	EF15	Mode pilote	Affichage en mode paysage pour le mode pilote doit afficher les interfaces en mode pilote sous le format paysage.	1
null	EF16	Mode pilote	Mettre en veille l'application après 3 secondes sans données doit se mettre en veille si une interruption de données survient et si elle dure plus de trois secondes.	1
null	ENF01	Usability	Utilisation du visuel de façon intuitive L'interface doit respecter le fonctionnement natif d'iOS lorsque les diverses actions sont effectuées dans le mode ingénieur de piste. Par exemple, l'ingénieur de piste doit entrer en mode édition pour supprimer ou déplacer une alarme ou un capteur.	6
null	ENF02	Usability	Utilisation du mode pilote doit être très simple Les différentes interfaces du mode pilote doivent être simples, claires et précises. Lorsque la voiture est en piste, le pilote ne doit pas avoir à réfléchir pour comprendre et utiliser l'application. Les quatre interfaces disponibles doivent être toutes visibles en trois clics du bouton situé sur le volant puis continuer de cette façon en boucle.	2
null	ENF03	Usability	Démarrage simple et rapide dans le mode configuré Lorsque le Dash Display démarre, il ne doit pas y avoir d'attente ou de commande à effectuer pour que l'application puisse être utilisée. Cette exigence est surtout importante pour le mode pilote qui ne peut pas utiliser l'écran tactile. De plus, le système doit utiliser le mode choisi dans les configurations de l'application directement dans iOS.	4
null	ENF04	Usability	Haut contraste dans les couleurs de l'interface Les couleurs de l'interface doivent avoir de très haut contraste. De plus, deux versions de couleurs	1

		doivent être disponibles dans la configuration de l'application sur iOS : un mode foncé et un mode pâle.	
null	ENF05 Performance	Rafraîchissement de l'écran à une cadence de 10 Hz La fréquence de rafraîchissement des interfaces est de 10 Hz afin que les données affichées soient toujours à jour en temps réel. Avec cette cadence, le véhicule a le temps de transmettre les nouvelles données par Wi-Fi. Cette mesure signifie 10 fois par seconde donc le rafraîchissement est d'une fois à chaque 100 ms.	4
null	ENF06 Disponibility	Période d'utilisation d'au maximum 25 minutes Les courses ont une durée d'au maximum 25 minutes donc l'application doit être optimisée pour une utilisation sans problème pour ce délai de temps.	1
null	ENF08 Disponibility	Aucun redémarrage de l'application en cas d'erreur Lorsqu'une erreur survient, l'application ne doit pas redémarrer seule. Il faut la redémarrer manuellement à chaque fois.	9

0.16.1 Perspectives

EF02 - Configuration de l'application dans les paramètres d'iOS Le système doit permettre de changer quelques configurations directement dans les paramètres de l'application sur iOS. Les configurations doivent inclure, entre autres, le changement de mode entre pilote et ingénieur de piste ainsi que le changement des couleurs de l'interface de pâle à foncé.

C02 - Usability Les configurations pour le mode par défaut et les couleurs de l'interface sont définies directement dans iOS.

EF03 - Gérer les données reçues en temps réel L'application doit constamment recevoir des données du bus CAN via Wi-Fi à partir du module Can2Ethernet et, en comparant avec la table des messages CAN, associer ces données aux capteurs et alarmes pour les afficher.

EF04 - Afficher des couleurs spécifiques pour les RPM L'application doit afficher l'indicateur de RPM avec un code de couleur précis, soit de jaune à rouge en passant par une zone orange visible. De 3000 RPM à 15 000 RPM, l'indicateur doit être dans le spectre de jaune à rouge. À 15 000 RPM et plus, l'indicateur doit être rouge. De plus, le rouge doit changer selon un paramètre calculé par l'ACL lorsqu'un message correspondant au « id » de la table CAN de ce paramètre est reçu.

C04 - Objective-C L'application doit être en Objective-C sous la plateforme iOS 7 et est destinée aux iPod Touch de 5e génération avec un écran de 4 pouces

EF05 - Afficher des couleurs spécifiques pour la température des pneus L'application doit afficher la température des trois capteurs de chaque pneu selon des couleurs spécifiques et avec des transitions fluides. Les capteurs sont situés à l'extérieur, au milieu et à l'intérieur de chacun des pneus. Lorsque la température est de 25°C et moins, la couleur est bleue. Entre 25°C et 65°C, la couleur passe de bleue à jaune. Entre 65°C et 95°C, l'indicateur passe de jaune à rouge. Finalement, en haut de 95°C, la couleur est rouge.

C03 - can2Ethernet La librairie Can2Ethernet développée par le club Formule ÉTS doit être utilisée.

EF06 - Afficher des couleurs spécifiques pour la température du moteur Le Dash Display doit afficher la température du moteur à l'aide de quatre couleurs. Lorsque la température est de 70°C et moins, l'indicateur doit être bleu. Entre 70°C et 90°C, la couleur utilisée est le vert. Entre 90°C et 100°C, l'indicateur doit être jaune et, finalement, il doit être rouge lorsque la température dépasse le 100°C.

EF07 - Afficher des couleurs spécifiques pour la puissance de la batterie Le système doit afficher la puissance de la batterie à l'aide de deux couleurs. Lorsque la batterie est à la puissance maximale (14V) jusqu'à 11.5V, le fond du capteur est vert. Lorsque la puissance atteint 11.5 V et en dessous, le fond devient rouge.

C05 - Wifi UDP Les données sont envoyées par Wi-Fi via le protocole UDP et le bus CAN.

EF08 - Afficher des couleurs spécifiques pour les alertes Le Dash Display doit afficher les alertes de deux façons selon leur statut. Dans tous les cas, elles sont affichées en rouge. Lorsqu'elles sont en cours, elles ont une opacité de 100 %. Sinon, l'opacité diminue à 30 %.

C08 - langue Les textes de l'application doivent être en anglais.

EF09 - Visualiser les alarmes et les capteurs sur l'interface pilote L'application doit afficher, de manière claire, les informations nécessaires au pilote en provenance des différents capteurs du véhicule. De plus, les messages d'alarmes seront aussi affichés de façon évidente et de façon à ce que le pilote les remarques immédiatement.

C07 - Id et offset des capteurs Les « id » et les « offset » des capteurs doivent suivre la table CAN fournie.

C09 - deployment L'application doit être installée et exécutée sur un iPod Touch qui ne nécessite pas un « iOS jailbreaking »

ENF01 - Utilisation du visuel de façon intuitive L'interface doit respecter le fonctionnement natif d'iOS lorsque les diverses actions sont effectuées dans le mode ingénieur de piste. Par exemple, l'ingénieur de piste doit entrer en mode édition pour supprimer ou déplacer une alarme ou un capteur.

ENF02 - Utilisation du mode pilote doit être très simple Les différentes interfaces du mode pilote doivent être simples, claires et précises. Lorsque la voiture est en piste, le pilote ne doit pas avoir à réfléchir pour comprendre et utiliser l'application. Les quatre interfaces disponibles doivent être toutes visibles en trois clics du bouton situé sur le volant puis continuer de cette façon en boucle.

EF10 - Changer l'interface lors de l'appui sur le bouton du volant L'application doit changer l'interface affichée lorsque le pilote appuie sur un bouton du volant. Le bouton envoie un message CAN à l'application pour lui indiquer de changer. L'application en mode pilote affichera quatre interfaces différentes en boucle.

EF11 - Contenu de la première interface L'application doit afficher, sur la première interface, les capteurs suivants : la température du moteur, le voltage de la batterie, l'indicateur d'utilisation et d'angle du système de réduction de traînée, les différentes alarmes, la boîte de messages, l'indicateur de vitesse et l'indicateur de révolutions par minute.

EF12 - Contenu de la deuxième interface Le Dash Display doit afficher, sur la deuxième interface, les capteurs suivants : la pression et la température des pneus, le biais de frein, l'antiroulis, l'odomètre, le voltage de la batterie, la température du moteur et un indicateur d'utilisation et l'angle du système de réduction de traînée.

EF13 - Contenu de la troisième interface L'application doit afficher, sur la troisième interface, le temps total de course, le temps du tour courant, la différence de temps par rapport au meilleur temps, le meilleur temps de tour de piste et la différence de temps par rapport au dernier de tour de piste.

EF14 - Contenu de la quatrième interface Dash Display doit afficher, sur la quatrième interface, un schéma de la piste de course et le déplacement de la voiture en temps réel. De plus, les temps suivants doivent être présents : le temps du tour de piste, le meilleur temps et la différence par rapport au meilleur temps.

EF15 - Affichage en mode paysage pour le mode pilote doit afficher les interfaces en mode pilote sous le format paysage.

EF16 - Mettre en veille l'application après 3 secondes sans données doit se mettre en veille si une interruption de données survient et si elle dure plus de trois secondes.

ENF03 - Démarrage simple et rapide dans le mode configuré Lorsque le Dash Display démarre, il ne doit pas y avoir d'attente ou de commande à effectuer pour que l'application puisse être utilisée. Cette exigence est surtout importante pour le mode pilote qui ne peut pas utiliser l'écran tactile. De plus, le système doit utiliser le mode choisi dans les configurations de l'application directement dans iOS.

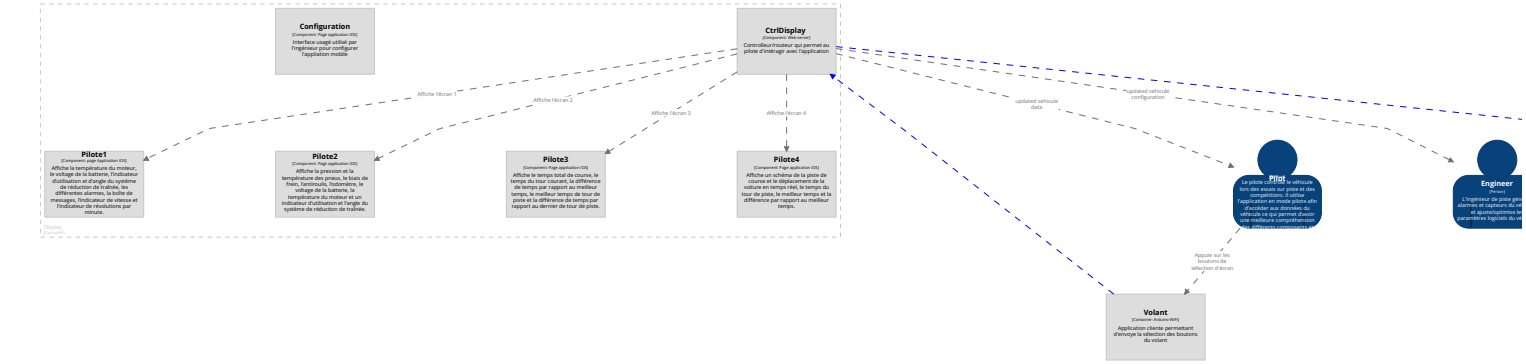
ENF04 - Haut contraste dans les couleurs de l'interface Les couleurs de l'interface doivent avoir de très haut contraste. De plus, deux versions de couleurs doivent être disponibles dans la configuration de l'application sur iOS : un mode foncé et un mode pâle.

ENF05 - Rafraîchissement de l'écran à une cadence de 10 Hz La fréquence de rafraîchissement des interfaces est de 10 Hz afin que les données affichées soient toujours à jour en temps réel. Avec cette cadence, le véhicule a le temps de transmettre les nouvelles données par Wi-Fi. Cette mesure signifie 10 fois par seconde donc le rafraîchissement est d'une fois à chaque 100 ms.

ENF06 - Période d'utilisation d'au maximum 25 minutes Les courses ont une durée d'au maximum 25 minutes donc l'application doit être optimisée pour une utilisation sans problème pour ce délai de temps.

ENF08 - Aucun redémarrage de l'application en cas d'erreur Lorsqu'une erreur survient, l'application ne doit pas redémarrer seule. Il faut la redémarrer manuellement à chaque fois.

Priority	**Requirements**
1	C04,C06,C08,EF15,EF16,EF20,ENF04,ENF06
2	C01,C02,C07,EF02,EF04,EF05,EF06,EF07,EF08,EF18,ENF02
3	
4	C05,C09,EF01,EF10,EF11,EF12,EF13,EF14,EF19,EF21,ENF03,ENF05
5	
6	EF09,EF17,ENF01
7	
8	
9	C03,EF03,EF22,ENF07,ENF08



0.17 Volant

Application cliente permettant d'envoye la sélection des boutons du volant

Tags: Element,Container

Parent	Key	Category	Title	Priority
null	EF10	Mode pilote	Changer l'interface lors de l'appui sur le bouton du volant L'application doit changer l'interface affichée lorsque le pilote appuie sur un bouton du volant. Le bouton envoie un message CAN à l'application pour lui indiquer de changer. L'application en mode pilote affichera quatre interfaces différentes en boucle.	4

0.17.1 Perspectives

EF10 - Changer l'interface lors de l'appui sur le bouton du volant L'application doit changer l'interface affichée lorsque le pilote appuie sur un bouton du volant. Le bouton envoie un message CAN à l'application pour lui indiquer de changer. L'application en mode pilote affichera quatre interfaces différentes en boucle.

0.18 Pilot

Le pilote contrôle le véhicule lors des essais sur piste et des compétitions. Il utilise l'application en mode pilote afin d'accéder aux données du véhicule ce qui permet d'avoir une meilleure compréhension des différents composants et d'améliorer sa conduite.

Tags: Element,Person

Parent	Key	Category	Title	Priority
null	EF02	Générale	Configuration de l'application dans les paramètres d'iOS Le système doit permettre de changer quelques configurations directement dans les paramètres de l'application sur iOS. Les configurations doivent inclure, entre autres, le changement de mode entre pilote et ingénieur de piste ainsi que le changement des couleurs de l'interface de pâle à foncé.	2
null	EF09	Mode pilote	Visualiser les alarmes et les capteurs sur l'interface pilote L'application doit afficher, de manière claire, les informations nécessaires au pilote en provenance des différents capteurs du véhicule. De plus, les messages d'alarmes seront aussi affichés de façon évidente et de façon à ce que le pilote les remarques immédiatement.	6
null	EF10	Mode pilote	Changer l'interface lors de l'appui sur le bouton du volant L'application doit changer l'interface affichée lorsque le pilote appuie sur un bouton du volant. Le bouton envoie un message CAN à l'application pour lui indiquer de changer. L'application en mode pilote affichera quatre interfaces différentes en boucle.	4
null	EF11	Mode pilote	Contenu de la première interface L'application doit afficher, sur la première interface, les capteurs suivants : la température du moteur, le voltage de la batterie, l'indicateur d'utilisation et d'angle du système de réduction de traînée, les	4

			différentes alarmes, la boîte de messages, l'indicateur de vitesse et l'indicateur de révolutions par minute.	
null	EF12	Mode pilote	Contenu de la deuxième interface Le Dash Display doit afficher, sur la deuxième interface, les capteurs suivants : la pression et la température des pneus, le biais de frein, l'antiroulis, l'odomètre, le voltage de la batterie, la température du moteur et un indicateur d'utilisation et l'angle du système de réduction de traînée.	4
null	EF13	Mode pilote	Contenu de la troisième interface L'application doit afficher, sur la troisième interface, le temps total de course, le temps du tour courant, la différence de temps par rapport au meilleur temps, le meilleur temps de tour de piste et la différence de temps par rapport au dernier de tour de piste.	4
null	EF14	Mode pilote	Contenu de la quatrième interface Dash Display doit afficher, sur la quatrième interface, un schéma de la piste de course et le déplacement de la voiture en temps réel. De plus, les temps suivants doivent être présents : le temps du tour de piste, le meilleur temps et la différence par rapport au meilleur temps.	4
null	EF15	Mode pilote	Affichage en mode paysage pour le mode pilote doit afficher les interfaces en mode pilote sous le format paysage.	1
null	EF16	Mode pilote	Mettre en veille l'application après 3 secondes sans données doit se mettre en veille si une interruption de données survient et si elle dure plus de trois secondes.	1

0.18.1 Perspectives

EF10 - Changer l'interface lors de l'appui sur le bouton du volant L'application doit changer l'interface affichée lorsque le pilote appuie sur un bouton du volant. Le bouton envoie un message CAN à l'application pour lui indiquer de changer. L'application en mode pilote affichera quatre interfaces différentes en boucle.

EF11 - Le pilot doit pouvoir activer l'écran 1 a partir d'un bouton sur le volant

EF12 - Le pilot doit pouvoir activer l'écran 2 a partir d'un bouton sur le volant

EF02 - Le système doit permettre de changer quelques configurations directement dans les paramètres de l'application sur iOS. Les configurations doivent inclure, entre autres, le changement de mode entre pilote et ingénieur de piste ainsi que le changement des couleurs de l'interface de pâle à foncé.

EF13 - Le pilot doit pouvoir activer l'écran 3 a partir d'un bouton sur le volant

EF15 - Affichage en mode paysage pour le mode pilote

EF16 - Mettre en veille l'application après 3 secondes sans donnéesÉ

EF09 - Visualiser les alarmes et les capteurs sur l'interface pilote L'application doit afficher, de manière claire, les informations nécessaires au pilote en provenance des différents capteurs du véhicule. De plus, les messages d'alarmes seront aussi affichés de façon évidente et de façon à ce que le pilote les remarques immédiatement.

0.19 CtrlDisplay

https://github.com/yvanross/log430-dashview-architecture/blob/master/src/main/java/dashview/Interfaces/ICtrlDisplay.java
Controlleur/routeur qui permet au pilote d'intéragir avec l'application

Technologie: Web server

Tags: Element,Component

Parent	Key	Category	Title	Priority
null	C02	null	Usability Les configurations pour le mode par défaut et les couleurs de l'interface sont définies directement dans iOS.	2

null	EF03	Générale	Gérer les données reçues en temps réel L’application doit constamment recevoir des données du bus CAN via Wi-Fi à partir du module Can2Ethernet et, en comparant avec la table des messages CAN, associer ces données aux capteurs et alarmes pour les afficher.	9
null	EF10	Mode pilote	Changer l’interface lors de l’appui sur le bouton du volant L’application doit changer l’interface affichée lorsque le pilote appuie sur un bouton du volant. Le bouton envoie un message CAN à l’application pour lui indiquer de changer. L’application en mode pilote affichera quatre interfaces différentes en boucle.	4
null	ENF03	Usability	Démarrage simple et rapide dans le mode configuré Lorsque le Dash Display démarre, il ne doit pas y avoir d’attente ou de commande à effectuer pour que l’application puisse être utilisée. Cette exigence est surtout importante pour le mode pilote qui ne peut pas utiliser l’écran tactile. De plus, le système doit utiliser le mode choisi dans les configurations de l’application directement dans iOS.	4
null	ENF05	Performance	Rafraîchissement de l’écran à une cadence de 10 Hz La fréquence de rafraîchissement des interfaces est de 10 Hz afin que les données affichées soient toujours à jour en temps réel. Avec cette cadence, le véhicule a le temps de transmettre les nouvelles données par Wi-Fi. Cette mesure signifie 10 fois par seconde donc le rafraîchissement est d’une fois à chaque 100 ms.	4

0.19.1 Perspectives

EF10 - Changer l’interface lors de l’appui sur le bouton du volant L’application doit changer l’interface affichée lorsque le pilote appuie sur un bouton du volant. Le bouton envoie un message CAN à l’application pour lui indiquer de changer. L’application en mode pilote affichera quatre interfaces différentes en boucle.

ENF03 - Démarrage simple et rapide dans le mode configuré Lorsque le Dash Display démarre, il ne doit pas y avoir d’attente ou de commande à effectuer pour que l’application puisse être utilisée. Cette exigence est surtout importante pour le mode pilote qui ne peut pas utiliser l’écran tactile. De plus, le système doit utiliser le mode choisi dans les configurations de l’application directement dans iOS.

ENF05 - Rafraîchissement de l’écran à une cadence de 10 Hz La fréquence de rafraîchissement des interfaces est de 10 Hz afin que les données affichées soient toujours à jour en temps réel. Avec cette cadence, le véhicule a le temps de transmettre les nouvelles données par Wi-Fi. Cette mesure signifie 10 fois par seconde donc le rafraîchissement est d’une fois à chaque 100 ms.

C02 - Usability Les configurations pour le mode par défaut et les couleurs de l’interface sont définies directement dans iOS.

EF03 - Gérer les données reçues en temps réel L’application doit constamment recevoir des données du bus CAN via Wi-Fi à partir du module Can2Ethernet et, en comparant avec la table des messages CAN, associer ces données aux capteurs et alarmes pour les afficher.

0.20 Engineer

L'ingénieur de piste gère les alarmes et capteurs du véhicule et ajuste/optimise les paramètres logiciels du véhicule

Tags: Element,Person

Parent	Key	Category	Title	Priority
null	EF01	Général	Configuration de l’application avec un fichier XML L’application doit utiliser un fichier de configuration, sous le format XML, pour déterminer les alarmes et capteurs disponibles. La liste des alarmes et des capteurs sont définis selon la table CAN fournie par la Formule ÉTS.	4
null	EF17	Mode ingénieur de piste	Visualiser les alarmes et les capteurs sur l’interface ingénieur doit afficher, sous forme de liste, les différents capteurs et alarmes que l’utilisateur décide d’inclure. Les capteurs et les alarmes sont affichés séparément, les alarmes se trouvant en haut de la liste. Lors de la présentation du prototype, le club formule ÉTS a précisé qu’ils souhaiteraient traiter les alarmes de la même façon que les différents capteurs donc les rendre modifiable sur la page principale.	6
null	EF18	Mode ingénieur de piste	Ajouter une alarme ou un capteur	2

			doit permettre à l'utilisateur de sélectionner dans une liste une alarme ou un capteur à ajouter à la liste d'affichage. L'application doit aussi permettre de filtrer la liste des alarmes et des capteurs qui peuvent être ajoutés et d'y effectuer une recherche.	
null	EF19	Mode ingénieur de piste	Changer l'ordre des alarmes et des capteurs affichés	4
			doit permettre à l'utilisateur, une fois en mode édition de la liste, de réorganiser respectivement les alarmes et les capteurs entre eux.	
null	EF20	Mode ingénieur de piste	Supprimer une alarme ou un capteur affiché	1
			doit permettre à l'utilisateur, une fois en mode édition de la liste, de supprimer un capteur ou une alarme.	
null	EF21	Mode ingénieur de piste	Afficher les détails de l'alarme ou du capteur	4
			système doit permettre de cliquer sur un capteur ou une alarme affichés afin d'obtenir plus de détails. Lors de la présentation du prototype, cette exigence a été clarifiée. Le client désire avoir la possibilité de modifier l'affichage du widget pour d'autres formats ainsi qu'obtenir un historique des dernières données. Les informations apparaissent sous le widget principal avec la possibilité d'afficher l'historique en plein écran.	
null	EF22	Mode ingénieur de piste	Gérer les cas d'erreurs de l'application	9
			doit, en cas d'erreurs de l'application, afficher les dernières données reçues. Les cas d'erreurs peuvent être, par exemple, une erreur de transmission de données ou un message d'erreur reçu par une chaîne CAN du module Can2Ethernet.	

0.20.1 Perspectives

EF21 - Afficher les détails de l'alarme ou du capteur

EF22 - Gérer les cas d'erreurs de l'application

EF01 - Configuration de l'application avec un fichier XML

C01 - L'ingénieur est responsable de réaliser le fichier de configureation XML

EF17 - Visualiser les alarmes et les capteurs sur l'interface ingénieur

EF18 - Ajouter une alarme ou un capteur doit permettre à l'utilisateur de sélectionner dans une liste une alarme ou un capteur à ajouter à la liste d'affichage. L'application doit aussi permettre de filtrer la liste des alarmes et des capteurs qui peuvent être ajoutés et d'y effectuer une recherche.

EF19 - Changer l'ordre des alarmes et des capteurs affichés

EF20 - Supprimer une alarme ou un capteur affiché

0.21 Pilote1

Affiche la température du moteur, le voltage de la batterie, l'indicateur d'utilisation et d'angle du système de réduction de traînée, les différentes alarmes, la boîte de messages, l'indicateur de vitesse et l'indicateur de révolutions par minute.

Technologie: page Application IOS

Tags: Element,Component

Parent	Key	Category	Title	Priority
null	EF04	Générale	Afficher des couleurs spécifiques pour les RPM	2
			L'application doit afficher l'indicateur de RPM avec un code de couleur précis, soit de jaune à rouge en passant par une zone orange visible. De 3000 RPM à 15 000 RPM, l'indicateur doit être dans le spectre	

			de jaune à rouge. À 15 000 RPM et plus, l'indicateur doit être rouge. De plus, le rouge doit changer selon un paramètre calculé par l'ACL lorsqu'un message correspondant au « id » de la table CAN de ce paramètre est reçu.	
null	EF06	Générale	Afficher des couleurs spécifiques pour la température du moteur Le Dash Display doit afficher la température du moteur à l'aide de quatre couleurs. Lorsque la température est de 70°C et moins, l'indicateur doit être bleu. Entre 70°C et 90°C, la couleur utilisée est le vert. Entre 90°C et 100°C, l'indicateur doit être jaune et, finalement, il doit être rouge lorsque la température dépasse le 100°C.	2
null	EF07	Générale	Afficher des couleurs spécifiques pour la puissance de la batterie Le système doit afficher la puissance de la batterie à l'aide de deux couleurs. Lorsque la batterie est à la puissance maximale (14V) jusqu'à 11.5V, le fond du capteur est vert. Lorsque la puissance atteint 11.5 V et en dessous, le fond devient rouge.	2
null	EF09	Mode pilote	Visualiser les alarmes et les capteurs sur l'interface pilote L'application doit afficher, de manière claire, les informations nécessaires au pilote en provenance des différents capteurs du véhicule. De plus, les messages d'alarmes seront aussi affichés de façon évidente et de façon à ce que le pilote les remarques immédiatement.	6
null	EF11	Mode pilote	Contenu de la première interface L'application doit afficher, sur la première interface, les capteurs suivants : la température du moteur, le voltage de la batterie, l'indicateur d'utilisation et d'angle du système de réduction de traînée, les différentes alarmes, la boîte de messages, l'indicateur de vitesse et l'indicateur de révolutions par minute.	4
null	ENF04	Usability	Haut contraste dans les couleurs de l'interface Les couleurs de l'interface doivent avoir de très haut contraste. De plus, deux versions de couleurs doivent être disponibles dans la configuration de l'application sur iOS : un mode foncé et un mode pâle.	1

0.21.1 Perspectives

EF11 - Contenu de la première interface L'application doit afficher, sur la première interface, les capteurs suivants : la température du moteur, le voltage de la batterie, l'indicateur d'utilisation et d'angle du système de réduction de traînée, les différentes alarmes, la boîte de messages, l'indicateur de vitesse et l'indicateur de révolutions par minute.

ENF04 - Haut contraste dans les couleurs de l'interface Les couleurs de l'interface doivent avoir de très haut contraste. De plus, deux versions de couleurs doivent être disponibles dans la configuration de l'application sur iOS : un mode foncé et un mode pâle.

EF04 - Afficher des couleurs spécifiques pour les RPM L'application doit afficher l'indicateur de RPM avec un code de couleur précis, soit de jaune à rouge en passant par une zone orange visible. De 3000 RPM à 15 000 RPM, l'indicateur doit être dans le spectre de jaune à rouge. À 15 000 RPM et plus, l'indicateur doit être rouge. De plus, le rouge doit changer selon un paramètre calculé par l'ACL lorsqu'un message correspondant au « id » de la table CAN de ce paramètre est reçu.

EF06 - Afficher des couleurs spécifiques pour la température du moteur Le Dash Display doit afficher la température du moteur à l'aide de quatre couleurs. Lorsque la température est de 70°C et moins, l'indicateur doit être bleu. Entre 70°C et 90°C, la couleur utilisée est le vert. Entre 90°C et 100°C, l'indicateur doit être jaune et, finalement, il doit être rouge lorsque la température dépasse le 100°C.

EF07 - Afficher des couleurs spécifiques pour la puissance de la batterie Le système doit afficher la puissance de la batterie à l'aide de deux couleurs. Lorsque la batterie est à la puissance maximale (14V) jusqu'à 11.5V, le fond du capteur est vert. Lorsque la puissance atteint 11.5 V et en dessous, le fond devient rouge.

EF09 - Visualiser les alarmes et les capteurs sur l'interface pilote L'application doit afficher, de manière claire, les informations nécessaires au pilote en provenance des différents capteurs du véhicule. De plus, les messages d'alarmes seront aussi affichés de façon évidente et de façon à ce que le pilote les remarques immédiatement.

0.22 Pilote2

Affiche la pression et la température des pneus, le biais de frein, l'antiroulis, l'odomètre, le voltage de la batterie, la température du moteur et un indicateur d'utilisation et l'angle du système de réduction de traînée.

Technologie: Page application IOS

Tags: Element,Component

Parent	Key	Category	Title	Priority
null	EF05	Générale	Afficher des couleurs spécifiques pour la température des pneus L'application doit afficher la température des trois capteurs de chaque pneu selon des couleurs spécifiques et avec des transitions fluides. Les capteurs sont situés à l'extérieur, au milieu et à l'intérieur de chacun des pneus. Lorsque la température est de 25°C et moins, la couleur est bleue. Entre 25°C et 65°C, la couleur passe de bleue à jaune. Entre 65°C et 95°C, l'indicateur passe de jaune à rouge. Finalement, en haut de 95°C, la couleur est rouge.	2
null	EF06	Générale	Afficher des couleurs spécifiques pour la température du moteur Le Dash Display doit afficher la température du moteur à l'aide de quatre couleurs. Lorsque la température est de 70°C et moins, l'indicateur doit être bleu. Entre 70°C et 90°C, la couleur utilisée est le vert. Entre 90°C et 100°C, l'indicateur doit être jaune et, finalement, il doit être rouge lorsque la température dépasse le 100°C.	2
null	EF07	Générale	Afficher des couleurs spécifiques pour la puissance de la batterie Le système doit afficher la puissance de la batterie à l'aide de deux couleurs. Lorsque la batterie est à la puissance maximale (14V) jusqu'à 11.5V, le fond du capteur est vert. Lorsque la puissance atteint 11.5 V et en dessous, le fond devient rouge.	2
null	EF09	Mode pilote	Visualiser les alarmes et les capteurs sur l'interface pilote L'application doit afficher, de manière claire, les informations nécessaires au pilote en provenance des différents capteurs du véhicule. De plus, les messages d'alarmes seront aussi affichés de façon évidente et de façon à ce que le pilote les remarques immédiatement.	6
null	EF12	Mode pilote	Contenu de la deuxième interface Le Dash Display doit afficher, sur la deuxième interface, les capteurs suivants : la pression et la température des pneus, le biais de frein, l'antiroulis, l'odomètre, le voltage de la batterie, la température du moteur et un indicateur d'utilisation et l'angle du système de réduction de traînée.	4
null	ENF04	Usability	Haut contraste dans les couleurs de l'interface Les couleurs de l'interface doivent avoir de très haut contraste. De plus, deux versions de couleurs doivent être disponibles dans la configuration de l'application sur iOS : un mode foncé et un mode pâle.	1

0.22.1 Perspectives

ENF04 - Haut contraste dans les couleurs de l'interface Les couleurs de l'interface doivent avoir de très haut contraste. De plus, deux versions de couleurs doivent être disponibles dans la configuration de l'application sur iOS : un mode foncé et un mode pâle.

EF12 - Contenu de la deuxième interface Le Dash Display doit afficher, sur la deuxième interface, les capteurs suivants : la pression et la température des pneus, le biais de frein, l'antiroulis, l'odomètre, le voltage de la batterie, la température du moteur et un indicateur d'utilisation et l'angle du système de réduction de traînée.

EF05 - Afficher des couleurs spécifiques pour la température des pneus L'application doit afficher la température des trois capteurs de chaque pneu selon des couleurs spécifiques et avec des transitions fluides. Les capteurs sont situés à l'extérieur, au milieu et à l'intérieur de chacun des pneus. Lorsque la température est de 25°C et moins, la couleur est bleue. Entre 25°C et 65°C, la couleur passe de bleue à jaune. Entre 65°C et 95°C, l'indicateur passe de jaune à rouge. Finalement, en haut de 95°C, la couleur est rouge.

EF06 - Afficher des couleurs spécifiques pour la température du moteur Le Dash Display doit afficher la température du moteur à l'aide de quatre couleurs. Lorsque la température est de 70°C et moins, l'indicateur doit être bleu. Entre 70°C et 90°C, la couleur utilisée est le vert. Entre 90°C et 100°C, l'indicateur doit être jaune et, finalement, il doit être rouge lorsque la température dépasse le 100°C.

EF07 - Afficher des couleurs spécifiques pour la puissance de la batterie Le système doit afficher la puissance de la batterie à l'aide de deux couleurs. Lorsque la batterie est à la puissance maximale (14V) jusqu'à 11.5V, le fond du capteur est vert. Lorsque la puissance atteint 11.5 V et en dessous, le fond devient rouge.

EF08 - Afficher des couleurs spécifiques pour les alertes Le Dash Display doit afficher les alertes de deux façons selon leur statut. Dans tous les cas, elles sont affichées en rouge. Lorsqu'elles sont en cours, elles ont une opacité de 100 %. Sinon, l'opacité diminue à 30 %.

EF09 - Visualiser les alarmes et les capteurs sur l'interface pilote L'application doit afficher, de manière claire, les informations nécessaires au pilote en provenance des différents capteurs du véhicule. De plus, les messages d'alarmes seront aussi affichés de façon

évidente et de façon à ce que le pilote les remarques immédiatement.

0.23 Pilote3

Affiche le temps total de course, le temps du tour courant, la différence de temps par rapport au meilleur temps, le meilleur temps de tour de piste et la différence de temps par rapport au dernier de tour de piste.

Technologie: Page application IOS

Tags: Element,Component

Parent	Key	Category	Title	Priority
null	EF09	Mode pilote	Visualiser les alarmes et les capteurs sur l'interface pilote	6
			L'application doit afficher, de manière claire, les informations nécessaires au pilote en provenance des différents capteurs du véhicule. De plus, les messages d'alarmes seront aussi affichés de façon évidente et de façon à ce que le pilote les remarques immédiatement.	
null	EF13	Mode pilote	Contenu de la troisième interface	4
			L'application doit afficher, sur la troisième interface, le temps total de course, le temps du tour courant, la différence de temps par rapport au meilleur temps, le meilleur temps de tour de piste et la différence de temps par rapport au dernier de tour de piste.	
null	ENF04	Usability	Haut contraste dans les couleurs de l'interface	1
			Les couleurs de l'interface doivent avoir de très haut contraste. De plus, deux versions de couleurs doivent être disponibles dans la configuration de l'application sur iOS : un mode foncé et un mode pâle.	

0.23.1 Perspectives

ENF04 - Haut contraste dans les couleurs de l'interface Les couleurs de l'interface doivent avoir de très haut contraste. De plus, deux versions de couleurs doivent être disponibles dans la configuration de l'application sur iOS : un mode foncé et un mode pâle.

EF13 - Contenu de la troisième interface L'application doit afficher, sur la troisième interface, le temps total de course, le temps du tour courant, la différence de temps par rapport au meilleur temps, le meilleur temps de tour de piste et la différence de temps par rapport au dernier de tour de piste.

EF09 - Visualiser les alarmes et les capteurs sur l'interface pilote L'application doit afficher, de manière claire, les informations nécessaires au pilote en provenance des différents capteurs du véhicule. De plus, les messages d'alarmes seront aussi affichés de façon évidente et de façon à ce que le pilote les remarques immédiatement.

0.24 Pilote4

Affiche un schéma de la piste de course et le déplacement de la voiture en temps réel, le temps du tour de piste, le meilleur temps et la différence par rapport au meilleur temps.

Technologie: Page application IOS

Tags: Element,Component

Parent	Key	Category	Title	Priority
null	EF09	Mode pilote	Visualiser les alarmes et les capteurs sur l'interface pilote	6
			L'application doit afficher, de manière claire, les informations nécessaires au pilote en provenance des différents capteurs du véhicule. De plus, les messages d'alarmes seront aussi affichés de façon évidente et de façon à ce que le pilote les remarques immédiatement.	
null	EF14	Mode pilote	Contenu de la quatrième interface	4
			Dash Display doit afficher, sur la quatrième interface, un schéma de la piste de course et le déplacement de la voiture en temps réel. De plus, les temps suivants doivent être présents : le temps du tour de piste, le meilleur temps et la différence par rapport au meilleur temps.	
null	ENF04	Usability	Haut contraste dans les couleurs de l'interface	1

			Les couleurs de l’interface doivent avoir de très haut contraste. De plus, deux versions de couleurs doivent être disponibles dans la configuration de l’application sur iOS : un mode foncé et un mode pâle.	

0.24.1 Perspectives

ENF04 - Haut contraste dans les couleurs de l’interface Les couleurs de l’interface doivent avoir de très haut contraste. De plus, deux versions de couleurs doivent être disponibles dans la configuration de l’application sur iOS : un mode foncé et un mode pâle.

EF14 - Contenu de la quatrième interface Dash Display doit afficher, sur la quatrième interface, un schéma de la piste de course et le déplacement de la voiture en temps réel. De plus, les temps suivants doivent être présents : le temps du tour de piste, le meilleur temps et la différence par rapport au meilleur temps.

EF09 - Visualiser les alarmes et les capteurs sur l’interface pilote L’application doit afficher, de manière claire, les informations nécessaires au pilote en provenance des différents capteurs du véhicule. De plus, les messages d’alarmes seront aussi affichés de façon évidente et de façon à ce que le pilote les remarques immédiatement.

0.25 Configuration

Interface usagé utilisé par l'ingénieur pour configurer l'appliation mobile

Technologie: Page application IOS

Tags: Element,Component

Parent	Key	Category	Title	Priority
null	EF02	Générale	Configuration de l’application dans les paramètres d’iOS	2
			Le système doit permettre de changer quelques configurations directement dans les paramètres de l’application sur iOS. Les configurations doivent inclure, entre autres, le changement de mode entre pilote et ingénieur de piste ainsi que le changement des couleurs de l’interface de pâle à foncé.	

0.25.1 Perspectives

EF02 - Configuration de l’application dans les paramètres d’iOS Le système doit permettre de changer quelques configurations directement dans les paramètres de l’application sur iOS. Les configurations doivent inclure, entre autres, le changement de mode entre pilote et ingénieur de piste ainsi que le changement des couleurs de l’interface de pâle à foncé.

0.26 cancanEthernet

https://github.com/yvanross/log430-dashview-architecture/blob/master/src/main/java/dashview/Interfaces/ICancanRouter.java
Routeur du Server web wifi controlant le bus CanCan pour l'acquisition/diffusion des données des capteurs

Technologie: CanCan bus Router

Tags: Element,Component

Parent	Key	Category	Title	Priority
null	C03	null	can2Ethernet	9
			La librairie Can2Ethernet développée par le club Formule ÉTS doit être utilisée.	
null	C05	Performance	fi UDP	4
			Les données sont envoyées par Wi-Fi via le protocole UDP et le bus CAN.	

0.26.1 Perspectives

C03 - can2Ethernet La librairie Can2Ethernet développée par le club Formule ÉTS doit être utilisée.

C05 - Wifi UDP Les données sont envoyées par Wi-Fi via le protocole UDP et le bus CAN.

public interface ICancanRouter

this is a test for the cancanRouter

- **Author:** Yvan Ross

  void setMaximumOperationTemperature(double maximumOperationTemperature)

Temperature d'opération

- **Parameters:** `maximumOperationTemperature` — parametre

  void setMaximumRPM(double maxRPM)

Nombre de tours minues

- **Parameters:** `maxRPM` — en tours par minutes

  void setMaximumTireTemperature(double maxTireTemperature)

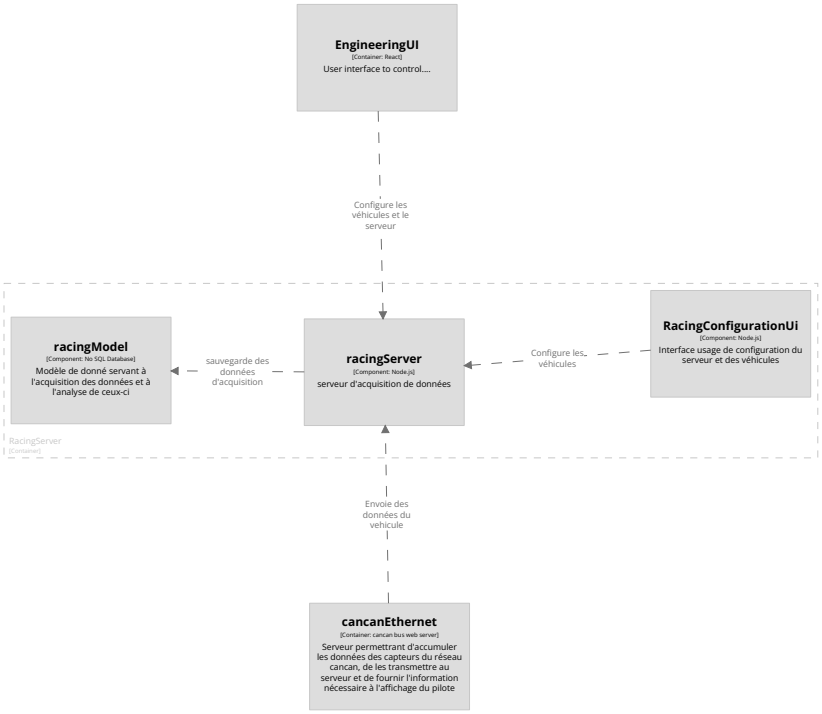
temperature des pneu

- **Parameters:** `maxTireTemperature` — parametre

  void setMaximumMotorTemperature(double maxTemp)

Temperature moteur

- **Parameters:** `maxTemp` — Temperature maximum du moteur en degré celcius



[Component] SystemRacing - RacingServer
Vue de décomposition du cont RacingServer
Tuesday, November 15, 2022 at 1:11 PM Eastern Standard Time

0.27 Table des éléments du conteneur racingServer

0.28 cancanEthernet

Serveur permettrant d'accumuler les données des capteurs du réseau cancan, de les transmettre au serveur et de fournir l'information nécessaire à l'affichage du pilote

Tags: Element,Container

Parent	Key	Category	Title	Priority

null	C03	null	can2Ethernet La librairie Can2Ethernet développée par le club Formule ÉTS doit être utilisée.	9
null	C05	Performance	fi UDP Les données sont envoyées par Wi-Fi via le protocole UDP et le bus CAN.	4
null	EF11	Mode pilote	Contenu de la première interface L'application doit afficher, sur la première interface, les capteurs suivants : la température du moteur, le voltage de la batterie, l'indicateur d'utilisation et d'angle du système de réduction de traînée, les différentes alarmes, la boîte de messages, l'indicateur de vitesse et l'indicateur de révolutions par minute.	4
null	EF12	Mode pilote	Contenu de la deuxième interface Le Dash Display doit afficher, sur la deuxième interface, les capteurs suivants : la pression et la température des pneus, le biais de frein, l'antiroulis, l'odomètre, le voltage de la batterie, la température du moteur et un indicateur d'utilisation et l'angle du système de réduction de traînée.	4
null	EF13	Mode pilote	Contenu de la troisième interface L'application doit afficher, sur la troisième interface, le temps total de course, le temps du tour courant, la différence de temps par rapport au meilleur temps, le meilleur temps de tour de piste et la différence de temps par rapport au dernier de tour de piste.	4
null	EF14	Mode pilote	Contenu de la quatrième interface Dash Display doit afficher, sur la quatrième interface, un schéma de la piste de course et le déplacement de la voiture en temps réel. De plus, les temps suivants doivent être présents : le temps du tour de piste, le meilleur temps et la différence par rapport au meilleur temps.	4

0.28.1 Perspectives

EF11 - Contenu de la première interface L'application doit afficher, sur la première interface, les capteurs suivants : la température du moteur, le voltage de la batterie, l'indicateur d'utilisation et d'angle du système de réduction de traînée, les différentes alarmes, la boîte de messages, l'indicateur de vitesse et l'indicateur de révolutions par minute.

EF12 - Contenu de la deuxième interface Le Dash Display doit afficher, sur la deuxième interface, les capteurs suivants : la pression et la température des pneus, le biais de frein, l'antiroulis, l'odomètre, le voltage de la batterie, la température du moteur et un indicateur d'utilisation et l'angle du système de réduction de traînée.

EF13 - Contenu de la troisième interface L'application doit afficher, sur la troisième interface, le temps total de course, le temps du tour courant, la différence de temps par rapport au meilleur temps, le meilleur temps de tour de piste et la différence de temps par rapport au dernier de tour de piste.

EF14 - Contenu de la quatrième interface Dash Display doit afficher, sur la quatrième interface, un schéma de la piste de course et le déplacement de la voiture en temps réel. De plus, les temps suivants doivent être présents : le temps du tour de piste, le meilleur temps et la différence par rapport au meilleur temps.

C03 - can2Ethernet La librairie Can2Ethernet développée par le club Formule ÉTS doit être utilisée.

C05 - Wifi UDP Les données sont envoyées par Wi-Fi via le protocole UDP et le bus CAN.

0.29 EngineeringUI

User interface to control....

Tags: Element,Container

Parent	Key	Category	Title	Priority
null	C04	null	Objective-C L'application doit être en Objective-C sous la plateforme iOS 7 et est destinée aux iPod Touch de 5e génération avec un écran de 4 pouces	1
null	C08	null	langue Les textes de l'application doivent être en anglais.	1
null	EF17	Mode ingénieur de piste	Visualiser les alarmes et les capteurs sur l'interface ingénieur	6

			doit afficher, sous forme de liste, les différents capteurs et alarmes que l'utilisateur décide d'inclure. Les capteurs et les alarmes sont affichés séparément, les alarmes se trouvant en haut de la liste. Lors de la présentation du prototype, le club formule ÉTS a précisé qu'ils souhaiteraient traiter les alarmes de la même façon que les différents capteurs donc les rendre modifiable sur la page principale.	
null	EF18	Mode ingénieur de piste	Ajouter une alarme ou un capteur doit permettre à l'utilisateur de sélectionner dans une liste une alarme ou un capteur à ajouter à la liste d'affichage. L'application doit aussi permettre de filtrer la liste des alarmes et des capteurs qui peuvent être ajoutés et d'y effectuer une recherche.	2
null	EF19	Mode ingénieur de piste	Changer l'ordre des alarmes et des capteurs affichés doit permettre à l'utilisateur, une fois en mode édition de la liste, de réorganiser respectivement les alarmes et les capteurs entre eux.	4
null	EF20	Mode ingénieur de piste	Supprimer une alarme ou un capteur affiché doit permettre à l'utilisateur, une fois en mode édition de la liste, de supprimer un capteur ou une alarme.	1
null	EF21	Mode ingénieur de piste	Afficher les détails de l'alarme ou du capteur système doit permettre de cliquer sur un capteur ou une alarme affichés afin d'obtenir plus de détails. Lors de la présentation du prototype, cette exigence a été clarifiée. Le client désire avoir la possibilité de modifier l'affichage du widget pour d'autres formats ainsi qu'obtenir un historique des dernières données. Les informations apparaissent sous le widget principal avec la possibilité d'afficher l'historique en plein écran.	4
null	EF22	Mode ingénieur de piste	Gérer les cas d'erreurs de l'application doit, en cas d'erreurs de l'application, afficher les dernières données reçues. Les cas d'erreurs peuvent être, par exemple, une erreur de transmission de données ou un message d'erreur reçu par une chaîne CAN du module Can2Ethernet.	9

0.29.1 Perspectives

EF21 - Afficher les détails de l'alarme ou du capteur système doit permettre de cliquer sur un capteur ou une alarme affichés afin d'obtenir plus de détails. Lors de la présentation du prototype, cette exigence a été clarifiée. Le client désire avoir la possibilité de modifier l'affichage du widget pour d'autres formats ainsi qu'obtenir un historique des dernières données. Les informations apparaissent sous le widget principal avec la possibilité d'afficher l'historique en plein écran.

EF22 - Gérer les cas d'erreurs de l'application doit, en cas d'erreurs de l'application, afficher les dernières données reçues. Les cas d'erreurs peuvent être, par exemple, une erreur de transmission de données ou un message d'erreur reçu par une chaîne CAN du module Can2Ethernet.

C04 - Objective-C L'application doit être en Objective-C sous la plateforme iOS 7 et est destinée aux iPod Touch de 5e génération avec un écran de 4 pouces

EF17 - Visualiser les alarmes et les capteurs sur l'interface ingénieur doit afficher, sous forme de liste, les différents capteurs et alarmes que l'utilisateur décide d'inclure. Les capteurs et les alarmes sont affichés séparément, les alarmes se trouvant en haut de la liste. Lors de la présentation du prototype, le club formule ÉTS a précisé qu'ils souhaiteraient traiter les alarmes de la même façon que les différents capteurs donc les rendre modifiable sur la page principale.

EF18 - Ajouter une alarme ou un capteur doit permettre à l'utilisateur de sélectionner dans une liste une alarme ou un capteur à ajouter à la liste d'affichage. L'application doit aussi permettre de filtrer la liste des alarmes et des capteurs qui peuvent être ajoutés et d'y effectuer une recherche.

EF19 - Changer l'ordre des alarmes et des capteurs affichés doit permettre à l'utilisateur, une fois en mode édition de la liste, de réorganiser respectivement les alarmes et les capteurs entre eux.

C08 - langue Les textes de l’application doivent être en anglais.

EF20 - Supprimer une alarme ou un capteur affiché doit permettre à l’utilisateur, une fois en mode édition de la liste, de supprimer un capteur ou une alarme.

0.30 RacingConfigurationUi

https://github.com/yvanross/log430-dashview-architecture/blob/master/src/main/java/dashview/Interfaces/IRacingConfigurationUI.java Interface usage de configuration du serveur et des véhicules

Technologie: Node.js

Tags: Element,Component

Parent Key Category Title Priority

0.30.1 Perspectives

public class IRacingConfigurationUI

User interface to configure vehicule for race

0.31 racingServer

https://github.com/yvanross/log430-dashview-architecture/blob/master/src/main/java/dashview/Interfaces/IRacingServer.java serveur d'acquisition de données

Technologie: Node.js

Tags: Element,Component

Parent Key Category Title Priority

0.31.1 Perspectives

public interface IRacingServer

interface du server d'acquisition de données provenant du véhicule

`public void startRacing()`

initialise le système en mode de course

`public void startTraining()`

Initialise le système en mode d'analyse de performance pour la préparation à la course.

`public void recordCanCanData(String cancanDataInJson)`

record cancan data from vehicule

- **Parameters:** `cancanData` — json format

  public void saveVehiculeConfiguration(String configurationInJson)

save vehicule configuration

0.32 racingModel

https://github.com/yvanross/log430-dashview-architecture/blob/master/src/main/java/dashview/Interfaces/IRacingModel.java Modèle de donné servant à l'acquisition des données et à l'analyse de ceux-ci

Technologie: No SQL Database

Tags: Element,Component

Parent	Key	Category	Title	Priority

0.32.1 Perspectives

public interface IRacingModel

interface de manipulation des données des véhicules

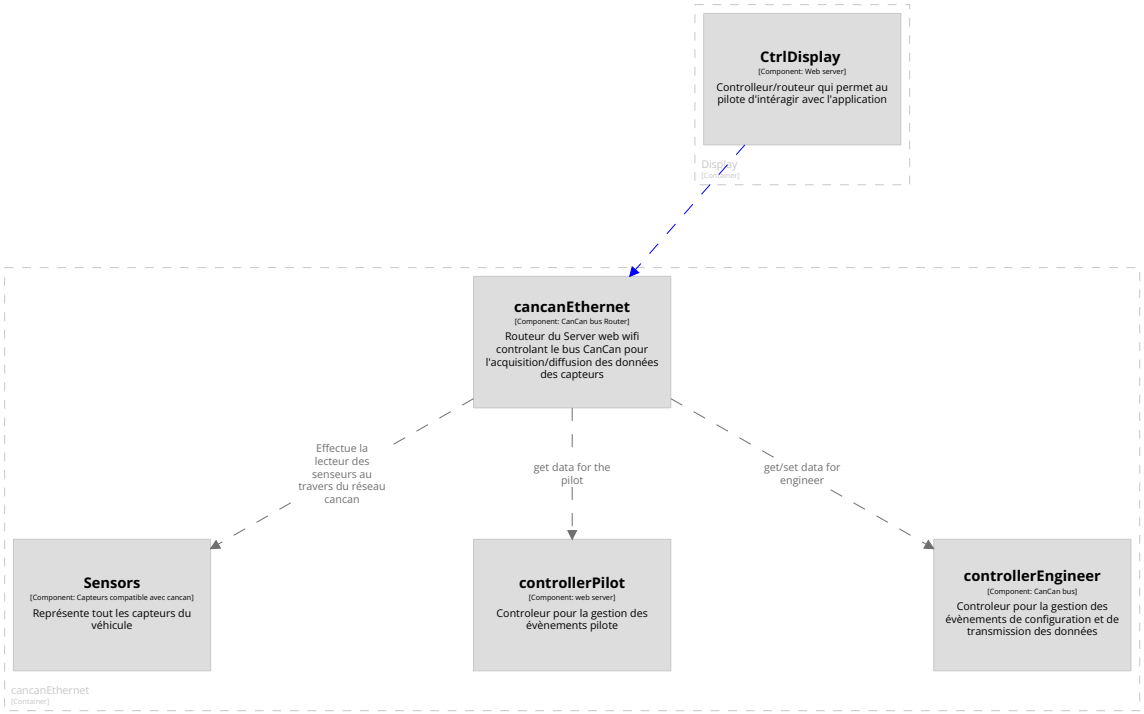
`public ArrayList getCanCanTable(Long vehiculeId)`

get cancan table

- **Parameters:** `vehiculeId` — identification du véhicule
- **Returns:** ArrayList , IsensorValue -> Timestamp,SensorId, valeur

  }

add cancanData



[Component] SystemVehicule - cancanEthernet
Component of the cancan Eternet cont
Tuesday, November 15, 2022 at 1:11 PM Eastern Standard Time

0.33 controllerPilot

https://github.com/yvanross/log430-dashview-architecture/blob/master/src/main/java/dashview/Interfaces/ICtrlPilot.java Controleur pour la gestion des évènements pilote

Technologie: web server

Tags: Element,Component

Parent	Key	Category	Title	Priority
null	EF11	Mode pilote	Contenu de la première interface L'application doit afficher, sur la première interface, les capteurs suivants : la température du moteur, le voltage de la batterie, l'indicateur d'utilisation et d'angle du système de réduction de traînée, les	4

			différentes alarmes, la boîte de messages, l'indicateur de vitesse et l'indicateur de révolutions par minute.	
null	EF12	Mode pilote	Contenu de la deuxième interface Le Dash Display doit afficher, sur la deuxième interface, les capteurs suivants : la pression et la température des pneus, le biais de frein, l'antiroulis, l'odomètre, le voltage de la batterie, la température du moteur et un indicateur d'utilisation et l'angle du système de réduction de traînée.	4
null	EF13	Mode pilote	Contenu de la troisième interface L'application doit afficher, sur la troisième interface, le temps total de course, le temps du tour courant, la différence de temps par rapport au meilleur temps, le meilleur temps de tour de piste et la différence de temps par rapport au dernier de tour de piste.	4
null	EF14	Mode pilote	Contenu de la quatrième interface Dash Display doit afficher, sur la quatrième interface, un schéma de la piste de course et le déplacement de la voiture en temps réel. De plus, les temps suivants doivent être présents : le temps du tour de piste, le meilleur temps et la différence par rapport au meilleur temps.	4

0.33.1 Perspectives

EF11 - Contenu de la première interface L'application doit afficher, sur la première interface, les capteurs suivants : la température du moteur, le voltage de la batterie, l'indicateur d'utilisation et d'angle du système de réduction de traînée, les différentes alarmes, la boîte de messages, l'indicateur de vitesse et l'indicateur de révolutions par minute.

EF12 - Contenu de la deuxième interface Le Dash Display doit afficher, sur la deuxième interface, les capteurs suivants : la pression et la température des pneus, le biais de frein, l'antiroulis, l'odomètre, le voltage de la batterie, la température du moteur et un indicateur d'utilisation et l'angle du système de réduction de traînée.

EF13 - Contenu de la troisième interface L'application doit afficher, sur la troisième interface, le temps total de course, le temps du tour courant, la différence de temps par rapport au meilleur temps, le meilleur temps de tour de piste et la différence de temps par rapport au dernier de tour de piste.

EF14 - Contenu de la quatrième interface Dash Display doit afficher, sur la quatrième interface, un schéma de la piste de course et le déplacement de la voiture en temps réel. De plus, les temps suivants doivent être présents : le temps du tour de piste, le meilleur temps et la différence par rapport au meilleur temps.

public class ICtrlPilot

controller pilot class

`void getDataInterface(int userInterfaceNumber)`

get user interfave by number

- **Parameters:** `userInterfaceNumber` — between 1..4

0.34 controllerEngineer

https://github.com/yvanross/log430-dashview-architecture/blob/master/src/main/java/dashview/Interfaces/ICtrlEngineer.java
Controleur pour la gestion des évènements de configuration et de transmission des données

Technologie: CanCan bus

Tags: Element,Component

Parent	Key	Category	Title	Priority
null	ENF07	Usability	Modification rapide des alarmes et des capteurs La liste des alarmes et des capteurs peut facilement être modifiée en 5 minutes et moins. Cette modification est effectuée sur le fichier de configuration XML en fournissant le « id » et l'« offset » définis dans la table CAN. Cette exigence n'était pas précisée de la part du client, mais une précision concernant la table CAN qui peut être modifiée a été faite. Il est donc important que tout le logiciel soit facilement maintenable.	9

0.34.1 Perspectives

ENF07 - Modification rapide des alarmes et des capteurs La liste des alarmes et des capteurs peut facilement être modifiée en 5 minutes et moins. Cette modification est effectuée sur le fichier de configuration XML en fournissant le « id » et l'« offset » définis dans la table CAN. Cette exigence n'était pas précisée de la part du client, mais une précision concernant la table CAN qui peut être modifiée a été faite. Il est donc important que tout le logiciel soit facilement maintenable.

public interface ICtrlEngineer

control engineer functions

`void setMaximumOperationTemperature(double maximumOperationTemperature)`

maximum operation temperature of the vehicule

- **Parameters:** `maximumOperationTemperature` — vehicule temp

  void setMaximumRPM(double maxRPM)

maximum rpm of the vehicule

- **Parameters:** `maxRPM` — max rpm in km/h

  void setMaximumTireTemperature(double maxTireTemperature)

max tire temperature

- **Parameters:** `maxTireTemperature` — in degre celcius

  void setMaximumBatteryPower(double maxBatteryPower)

maximum battery power

- **Parameters:** `maxBatteryPower` — in watts

  void setMaximumMotorTemperature(double maxMotorTemperature)

maximum motor temperature

- **Parameters:** `maxMotorTemperature` — in celcius

0.35 CtrlDisplay

https://github.com/yvanross/log430-dashview-architecture/blob/master/src/main/java/dashview/Interfaces/ICtrlDisplay.java
Controlleur/routeur qui permet au pilote d'intéragir avec l'application

Technologie: Web server

Tags: Element,Component

Parent	Key	Category	Title	Priority
null	C02	null	Usability Les configurations pour le mode par défaut et les couleurs de l'interface sont définies directement dans iOS.	2
null	EF03	Générale	Gérer les données reçues en temps réel L'application doit constamment recevoir des données du bus CAN via Wi-Fi à partir du module Can2Ethernet et, en comparant avec la table des messages CAN, associer ces données aux capteurs et alarmes pour les afficher.	9
null	EF10	Mode pilote	Changer l'interface lors de l'appui sur le bouton du volant L'application doit changer l'interface affichée lorsque le pilote appuie sur un bouton du volant. Le bouton envoie un message CAN à l'application pour lui indiquer de changer. L'application en mode pilote affichera quatre interfaces différentes en boucle.	4
null	ENF03	Usability	Démarrage simple et rapide dans le mode configuré Lorsque le Dash Display démarre, il ne doit pas y avoir d'attente ou de commande à effectuer	4

			pour que l'application puisse être utilisée. Cette exigence est surtout importante pour le mode pilote qui ne peut pas utiliser l'écran tactile. De plus, le système doit utiliser le mode choisi dans les configurations de l'application directement dans iOS.	
null	ENF05	Performance	Rafraîchissement de l'écran à une cadence de 10 Hz La fréquence de rafraîchissement des interfaces est de 10 Hz afin que les données affichées soient toujours à jour en temps réel. Avec cette cadence, le véhicule a le temps de transmettre les nouvelles données par Wi-Fi. Cette mesure signifie 10 fois par seconde donc le rafraîchissement est d'une fois à chaque 100 ms.	4

0.35.1 Perspectives

EF10 - Changer l'interface lors de l'appui sur le bouton du volant L'application doit changer l'interface affichée lorsque le pilote appuie sur un bouton du volant. Le bouton envoie un message CAN à l'application pour lui indiquer de changer. L'application en mode pilote affichera quatre interfaces différentes en boucle.

ENF03 - Démarrage simple et rapide dans le mode configuré Lorsque le Dash Display démarre, il ne doit pas y avoir d'attente ou de commande à effectuer pour que l'application puisse être utilisée. Cette exigence est surtout importante pour le mode pilote qui ne peut pas utiliser l'écran tactile. De plus, le système doit utiliser le mode choisi dans les configurations de l'application directement dans iOS.

ENF05 - Rafraîchissement de l'écran à une cadence de 10 Hz La fréquence de rafraîchissement des interfaces est de 10 Hz afin que les données affichées soient toujours à jour en temps réel. Avec cette cadence, le véhicule a le temps de transmettre les nouvelles données par Wi-Fi. Cette mesure signifie 10 fois par seconde donc le rafraîchissement est d'une fois à chaque 100 ms.

C02 - Usability Les configurations pour le mode par défaut et les couleurs de l'interface sont définies directement dans iOS.

EF03 - Gérer les données reçues en temps réel L'application doit constamment recevoir des données du bus CAN via Wi-Fi à partir du module Can2Ethernet et, en comparant avec la table des messages CAN, associer ces données aux capteurs et alarmes pour les afficher.

0.36 cancanEthernet

https://github.com/yvanross/log430-dashview-architecture/blob/master/src/main/java/dashview/Interfaces/ICancanRouter.java
Routeur du Server web wifi controlant le bus CanCan pour l'acquisition/diffusion des données des capteurs

Technologie: CanCan bus Router

Tags: Element,Component

Parent	Key	Category	Title	Priority
null	C03	null	can2Ethernet La librairie Can2Ethernet développée par le club Formule ÉTS doit être utilisée.	9
null	C05	Performance	fi UDP Les données sont envoyées par Wi-Fi via le protocole UDP et le bus CAN.	4

0.36.1 Perspectives

C03 - can2Ethernet La librairie Can2Ethernet développée par le club Formule ÉTS doit être utilisée.

C05 - Wifi UDP Les données sont envoyées par Wi-Fi via le protocole UDP et le bus CAN.

public interface ICancanRouter

this is a test for the cancanRouter

- **Author:** Yvan Ross

  void setMaximumOperationTemperature(double maximumOperationTemperature)

Temperature d'opération

- **Parameters:** `maximumOperationTemperature` — parametre

  void setMaximumRPM(double maxRPM)

Nombre de tours minues

- **Parameters:** `maxRPM` — en tours par minutes

  void setMaximumTireTemperature(double maxTireTemperature)

temperature des pneu

- **Parameters:** `maxTireTemperature` — parametre

  void setMaximumMotorTemperature(double maxTemp)

Temperature moteur

- **Parameters:** `maxTemp` — Temperature maximum du moteur en degré celcius

0.37 Display

Application cliente permettant d'affiche les informations au pilote durant la course

Tags: Element,Container

Parent	Key	Category	Title	Priority
null	C02	null	Usability Les configurations pour le mode par défaut et les couleurs de l'interface sont définies directement dans iOS.	2
null	C04	null	Objective-C L'application doit être en Objective-C sous la plateforme iOS 7 et est destinée aux iPod Touch de 5e génération avec un écran de 4 pouces	1
null	C07	null	Id et offset des capteurs Les « id » et les « offset » des capteurs doivent suivre la table CAN fournie.	2
null	C08	null	langue Les textes de l'application doivent être en anglais.	1
null	C09	null	deployment L'application doit être installée et exécutée sur un iPod Touch qui ne nécessite pas un « iOS jailbreaking »	4
null	EF02	Générale	Configuration de l'application dans les paramètres d'iOS Le système doit permettre de changer quelques configurations directement dans les paramètres de l'application sur iOS. Les configurations doivent inclure, entre autres, le changement de mode entre pilote et ingénieur de piste ainsi que le changement des couleurs de l'interface de pâle à foncé.	2
null	EF03	Générale	Gérer les données reçues en temps réel L'application doit constamment recevoir des données du bus CAN via Wi-Fi à partir du module Can2Ethernet et, en comparant avec la table des messages CAN, associer ces données aux capteurs et alarmes pour les afficher.	9
null	EF04	Générale	Afficher des couleurs spécifiques pour les RPM L'application doit afficher l'indicateur de RPM avec un code de couleur précis, soit de jaune à rouge en passant par une zone orange visible. De 3000 RPM à 15 000 RPM, l'indicateur doit être dans le spectre de jaune à rouge. À 15 000 RPM et plus, l'indicateur doit être rouge. De plus, le rouge doit changer selon un paramètre calculé par l'ACL lorsqu'un message correspondant au « id » de la table CAN de ce paramètre est reçu.	2
null	EF05	Générale	Afficher des couleurs spécifiques pour la température des pneus L'application doit afficher la température des trois capteurs de chaque pneu selon des couleurs spécifiques et avec des transitions fluides. Les capteurs sont situés à l'extérieur, au milieu et à l'intérieur de chacun des pneus. Lorsque la température est de 25°C et moins, la couleur est bleue. Entre 25°C et 65°C, la couleur passe de bleue à jaune. Entre 65°C et 95°C, l'indicateur passe de jaune à rouge. Finalement, en haut de 95°C, la couleur est rouge.	2
null	EF06	Générale	Afficher des couleurs spécifiques pour la température du moteur Le Dash Display doit afficher la température du moteur à l'aide de quatre couleurs. Lorsque la	2

			température est de 70°C et moins, l'indicateur doit être bleu. Entre 70°C et 90°C, la couleur utilisée est le vert. Entre 90°C et 100°C, l'indicateur doit être jaune et, finalement, il doit être rouge lorsque la température dépasse le 100°C.	
null	EF07	Générale	Afficher des couleurs spécifiques pour la puissance de la batterie Le système doit afficher la puissance de la batterie à l'aide de deux couleurs. Lorsque la batterie est à la puissance maximale (14V) jusqu'à 11.5V, le fond du capteur est vert. Lorsque la puissance atteint 11.5 V et en dessous, le fond devient rouge.	2
null	EF08	Générale	Afficher des couleurs spécifiques pour les alertes Le Dash Display doit afficher les alertes de deux façons selon leur statut. Dans tous les cas, elles sont affichées en rouge. Lorsqu'elles sont en cours, elles ont une opacité de 100 %. Sinon, l'opacité diminue à 30 %.	2
null	EF09	Mode pilote	Visualiser les alarmes et les capteurs sur l'interface pilote L'application doit afficher, de manière claire, les informations nécessaires au pilote en provenance des différents capteurs du véhicule. De plus, les messages d'alarmes seront aussi affichés de façon évidente et de façon à ce que le pilote les remarques immédiatement.	6
null	EF15	Mode pilote	Affichage en mode paysage pour le mode pilote doit afficher les interfaces en mode pilote sous le format paysage.	1
null	EF16	Mode pilote	Mettre en veille l'application après 3 secondes sans données doit se mettre en veille si une interruption de données survient et si elle dure plus de trois secondes.	1
null	ENF01	Usability	Utilisation du visuel de façon intuitive L'interface doit respecter le fonctionnement natif d'iOS lorsque les diverses actions sont effectuées dans le mode ingénieur de piste. Par exemple, l'ingénieur de piste doit entrer en mode édition pour supprimer ou déplacer une alarme ou un capteur.	6
null	ENF02	Usability	Utilisation du mode pilote doit être très simple Les différentes interfaces du mode pilote doivent être simples, claires et précises. Lorsque la voiture est en piste, le pilote ne doit pas avoir à réfléchir pour comprendre et utiliser l'application. Les quatre interfaces disponibles doivent être toutes visibles en trois clics du bouton situé sur le volant puis continuer de cette façon en boucle.	2
null	ENF03	Usability	Démarrage simple et rapide dans le mode configuré Lorsque le Dash Display démarre, il ne doit pas y avoir d'attente ou de commande à effectuer pour que l'application puisse être utilisée. Cette exigence est surtout importante pour le mode pilote qui ne peut pas utiliser l'écran tactile. De plus, le système doit utiliser le mode choisi dans les configurations de l'application directement dans iOS.	4
null	ENF04	Usability	Haut contraste dans les couleurs de l'interface Les couleurs de l'interface doivent avoir de très haut contraste. De plus, deux versions de couleurs doivent être disponibles dans la configuration de l'application sur iOS : un mode foncé et un mode pâle.	1
null	ENF05	Performance	Rafraîchissement de l'écran à une cadence de 10 Hz La fréquence de rafraîchissement des interfaces est de 10 Hz afin que les données affichées soient toujours à jour en temps réel. Avec cette cadence, le véhicule a le temps de transmettre les nouvelles données par Wi-Fi. Cette mesure signifie 10 fois par seconde donc le rafraîchissement est d'une fois à chaque 100 ms.	4
null	ENF06	Disponibility	Période d'utilisation d'au maximum 25 minutes Les courses ont une durée d'au maximum 25 minutes donc l'application doit être optimisée pour une utilisation sans problème pour ce délai de temps.	1
null	ENF08	Disponibility	Aucun redémarrage de l'application en cas d'erreur Lorsqu'une erreur survient, l'application ne doit pas redémarrer seule. Il faut la redémarrer manuellement à chaque fois.	9

0.37.1 Perspectives

EF02 - Configuration de l'application dans les paramètres d'iOS Le système doit permettre de changer quelques configurations directement dans les paramètres de l'application sur iOS. Les configurations doivent inclure, entre autres, le changement de mode entre pilote et ingénieur de piste ainsi que le changement des couleurs de l'interface de pâle à foncé.

C02 - Usability Les configurations pour le mode par défaut et les couleurs de l'interface sont définies directement dans iOS.

EF03 - Gérer les données reçues en temps réel L'application doit constamment recevoir des données du bus CAN via Wi-Fi à partir du module Can2Ethernet et, en comparant avec la table des messages CAN, associer ces données aux capteurs et alarmes pour les afficher.

EF04 - Afficher des couleurs spécifiques pour les RPM L'application doit afficher l'indicateur de RPM avec un code de couleur précis, soit de jaune à rouge en passant par une zone orange visible. De 3000 RPM à 15 000 RPM, l'indicateur doit être dans le spectre de jaune à rouge. À 15 000 RPM et plus, l'indicateur doit être rouge. De plus, le rouge doit changer selon un paramètre calculé par l'ACL lorsqu'un message correspondant au « id » de la table CAN de ce paramètre est reçu.

EF15 - Affichage en mode paysage pour le mode pilote doit afficher les interfaces en mode pilote sous le format paysage.

C04 - Objective-C L'application doit être en Objective-C sous la plateforme iOS 7 et est destinée aux iPod Touch de 5e génération avec un écran de 4 pouces

EF05 - Afficher des couleurs spécifiques pour la température des pneus L'application doit afficher la température des trois capteurs de chaque pneu selon des couleurs spécifiques et avec des transitions fluides. Les capteurs sont situés à l'extérieur, au milieu et à l'intérieur de chacun des pneus. Lorsque la température est de 25°C et moins, la couleur est bleue. Entre 25°C et 65°C, la couleur passe de bleue à jaune. Entre 65°C et 95°C, l'indicateur passe de jaune à rouge. Finalement, en haut de 95°C, la couleur est rouge.

EF16 - Mettre en veille l'application après 3 secondes sans données doit se mettre en veille si une interruption de données survient et si elle dure plus de trois secondes.

EF06 - Afficher des couleurs spécifiques pour la température du moteur Le Dash Display doit afficher la température du moteur à l'aide de quatre couleurs. Lorsque la température est de 70°C et moins, l'indicateur doit être bleu. Entre 70°C et 90°C, la couleur utilisée est le vert. Entre 90°C et 100°C, l'indicateur doit être jaune et, finalement, il doit être rouge lorsque la température dépasse le 100°C.

EF07 - Afficher des couleurs spécifiques pour la puissance de la batterie Le système doit afficher la puissance de la batterie à l'aide de deux couleurs. Lorsque la batterie est à la puissance maximale (14V) jusqu'à 11.5V, le fond du capteur est vert. Lorsque la puissance atteint 11.5 V et en dessous, le fond devient rouge.

EF08 - Afficher des couleurs spécifiques pour les alertes Le Dash Display doit afficher les alertes de deux façons selon leur statut. Dans tous les cas, elles sont affichées en rouge. Lorsqu'elles sont en cours, elles ont une opacité de 100 %. Sinon, l'opacité diminue à 30 %.

C08 - langue Les textes de l'application doivent être en anglais.

EF09 - Visualiser les alarmes et les capteurs sur l'interface pilote L'application doit afficher, de manière claire, les informations nécessaires au pilote en provenance des différents capteurs du véhicule. De plus, les messages d'alarmes seront aussi affichés de façon évidente et de façon à ce que le pilote les remarques immédiatement.

C07 - Id et offset des capteurs Les « id » et les « offset » des capteurs doivent suivre la table CAN fournie.

C09 - deployment L'application doit être installée et exécutée sur un iPod Touch qui ne nécessite pas un « iOS jailbreaking »

ENF01 - Utilisation du visuel de façon intuitive L'interface doit respecter le fonctionnement natif d'iOS lorsque les diverses actions sont effectuées dans le mode ingénieur de piste. Par exemple, l'ingénieur de piste doit entrer en mode édition pour supprimer ou déplacer une alarme ou un capteur.

ENF02 - Utilisation du mode pilote doit être très simple Les différentes interfaces du mode pilote doivent être simples, claires et précises. Lorsque la voiture est en piste, le pilote ne doit pas avoir à réfléchir pour comprendre et utiliser l'application. Les quatre interfaces disponibles doivent être toutes visibles en trois clics du bouton situé sur le volant puis continuer de cette façon en boucle.

ENF03 - Démarrage simple et rapide dans le mode configuré Lorsque le Dash Display démarre, il ne doit pas y avoir d'attente ou de commande à effectuer pour que l'application puisse être utilisée. Cette exigence est surtout importante pour le mode pilote qui ne peut pas utiliser l'écran tactile. De plus, le système doit utiliser le mode choisi dans les configurations de l'application directement dans iOS.

ENF04 - Haut contraste dans les couleurs de l'interface Les couleurs de l'interface doivent avoir de très haut contraste. De plus, deux versions de couleurs doivent être disponibles dans la configuration de l'application sur iOS : un mode foncé et un mode pâle.

ENF05 - Rafraîchissement de l'écran à une cadence de 10 Hz La fréquence de rafraîchissement des interfaces est de 10 Hz afin que les données affichées soient toujours à jour en temps réel. Avec cette cadence, le véhicule a le temps de transmettre les nouvelles données par Wi-Fi. Cette mesure signifie 10 fois par seconde donc le rafraîchissement est d'une fois à chaque 100 ms.

ENF06 - Période d'utilisation d'au maximum 25 minutes Les courses ont une durée d'au maximum 25 minutes donc l'application doit

être optimisée pour une utilisation sans problème pour ce délai de temps.

ENF08 - Aucun redémarrage de l'application en cas d'erreur Lorsqu'une erreur survient, l'application ne doit pas redémarrer seule. Il faut la redémarrer manuellement à chaque fois.

0.38 Sensors

Représente tout les capteurs du véhicule

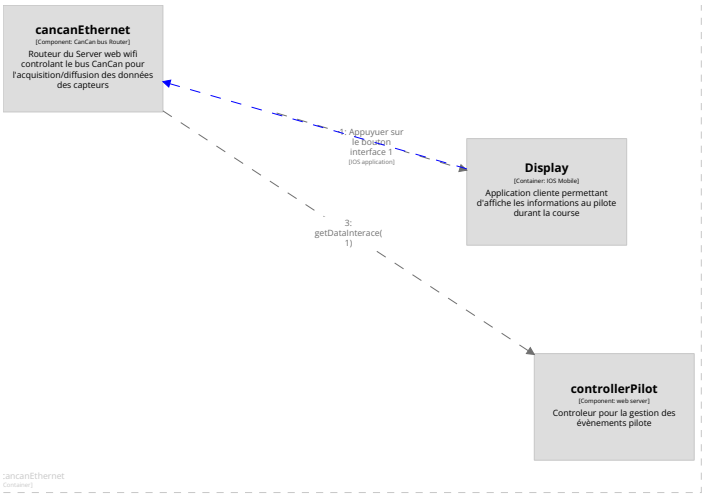
Technologie: Capteurs compatible avec cancan

Tags: Element,Component

Parent Key Category Title Priority

0.38.1 Perspectives

C03 - Capteurs permettant l'acquision de l'état du véhicule



[Dynamic] SystemVehicule - cancanEthernet
Diagramme pour démontrer comment l'appliation du pilot récupérer les données
Tuesday, November 15, 2022 at 1:11 PM Eastern Standard Time

0.39 controllerPilot

https://github.com/yvanross/log430-dashview-architecture/blob/master/src/main/java/dashview/Interfaces/ICtrlPilot.java Controleur pour la gestion des évènements pilote

Technologie: web server

Tags: Element,Component

Parent	Key	Category	Title	Priority
null	EF11	Mode pilote	Contenu de la première interface L'application doit afficher, sur la première interface, les capteurs suivants : la température du moteur, le voltage de la batterie, l'indicateur d'utilisation et d'angle du système de réduction de traînée, les	4

			différentes alarmes, la boîte de messages, l'indicateur de vitesse et l'indicateur de révolutions par minute.	
null	EF12	Mode pilote	Contenu de la deuxième interface Le Dash Display doit afficher, sur la deuxième interface, les capteurs suivants : la pression et la température des pneus, le biais de frein, l'antiroulis, l'odomètre, le voltage de la batterie, la température du moteur et un indicateur d'utilisation et l'angle du système de réduction de traînée.	4
null	EF13	Mode pilote	Contenu de la troisième interface L'application doit afficher, sur la troisième interface, le temps total de course, le temps du tour courant, la différence de temps par rapport au meilleur temps, le meilleur temps de tour de piste et la différence de temps par rapport au dernier de tour de piste.	4
null	EF14	Mode pilote	Contenu de la quatrième interface Dash Display doit afficher, sur la quatrième interface, un schéma de la piste de course et le déplacement de la voiture en temps réel. De plus, les temps suivants doivent être présents : le temps du tour de piste, le meilleur temps et la différence par rapport au meilleur temps.	4

0.39.1 Perspectives

EF11 - Contenu de la première interface L'application doit afficher, sur la première interface, les capteurs suivants : la température du moteur, le voltage de la batterie, l'indicateur d'utilisation et d'angle du système de réduction de traînée, les différentes alarmes, la boîte de messages, l'indicateur de vitesse et l'indicateur de révolutions par minute.

EF12 - Contenu de la deuxième interface Le Dash Display doit afficher, sur la deuxième interface, les capteurs suivants : la pression et la température des pneus, le biais de frein, l'antiroulis, l'odomètre, le voltage de la batterie, la température du moteur et un indicateur d'utilisation et l'angle du système de réduction de traînée.

EF13 - Contenu de la troisième interface L'application doit afficher, sur la troisième interface, le temps total de course, le temps du tour courant, la différence de temps par rapport au meilleur temps, le meilleur temps de tour de piste et la différence de temps par rapport au dernier de tour de piste.

EF14 - Contenu de la quatrième interface Dash Display doit afficher, sur la quatrième interface, un schéma de la piste de course et le déplacement de la voiture en temps réel. De plus, les temps suivants doivent être présents : le temps du tour de piste, le meilleur temps et la différence par rapport au meilleur temps.

public class ICtrlPilot

controller pilot class

```
void getDataInterface(int userInterfaceNumber)
```

get user interfave by number

- **Parameters:** `userInterfaceNumber` — between 1..4

0.40 Pilot

Le pilote contrôle le véhicule lors des essais sur piste et des compétitions. Il utilise l'application en mode pilote afin d'accéder aux données du véhicule ce qui permet d'avoir une meilleure compréhension des différents composants et d'améliorer sa conduite.

Tags: Element,Person

Parent	Key	Category	Title	Priority
null	EF02	Générale	Configuration de l'application dans les paramètres d'iOS Le système doit permettre de changer quelques configurations directement dans les paramètres de l'application sur iOS. Les configurations doivent inclure, entre autres, le changement de mode entre pilote et ingénieur de piste ainsi que le changement des couleurs de l'interface de pâle à foncé.	2
null	EF09	Mode pilote	Visualiser les alarmes et les capteurs sur l'interface pilote L'application doit afficher, de manière claire, les informations nécessaires au pilote en provenance des	6

			différents capteurs du véhicule. De plus, les messages d'alarmes seront aussi affichés de façon évidente et de façon à ce que le pilote les remarques immédiatement.	
null	EF10	Mode pilote	Changer l'interface lors de l'appui sur le bouton du volant	4
			L'application doit changer l'interface affichée lorsque le pilote appuie sur un bouton du volant. Le bouton envoie un message CAN à l'application pour lui indiquer de changer. L'application en mode pilote affichera quatre interfaces différentes en boucle.	
null	EF11	Mode pilote	Contenu de la première interface	4
			L'application doit afficher, sur la première interface, les capteurs suivants : la température du moteur, le voltage de la batterie, l'indicateur d'utilisation et d'angle du système de réduction de traînée, les différentes alarmes, la boîte de messages, l'indicateur de vitesse et l'indicateur de révolutions par minute.	
null	EF12	Mode pilote	Contenu de la deuxième interface	4
			Le Dash Display doit afficher, sur la deuxième interface, les capteurs suivants : la pression et la température des pneus, le biais de frein, l'antiroulis, l'odomètre, le voltage de la batterie, la température du moteur et un indicateur d'utilisation et l'angle du système de réduction de traînée.	
null	EF13	Mode pilote	Contenu de la troisième interface	4
			L'application doit afficher, sur la troisième interface, le temps total de course, le temps du tour courant, la différence de temps par rapport au meilleur temps, le meilleur temps de tour de piste et la différence de temps par rapport au dernier de tour de piste.	
null	EF14	Mode pilote	Contenu de la quatrième interface	4
			Dash Display doit afficher, sur la quatrième interface, un schéma de la piste de course et le déplacement de la voiture en temps réel. De plus, les temps suivants doivent être présents : le temps du tour de piste, le meilleur temps et la différence par rapport au meilleur temps.	
null	EF15	Mode pilote	Affichage en mode paysage pour le mode pilote	1
			doit afficher les interfaces en mode pilote sous le format paysage.	
null	EF16	Mode pilote	Mettre en veille l'application après 3 secondes sans données	1
			doit se mettre en veille si une interruption de données survient et si elle dure plus de trois secondes.	

0.40.1 Perspectives

EF10 - Changer l'interface lors de l'appui sur le bouton du volant L'application doit changer l'interface affichée lorsque le pilote appuie sur un bouton du volant. Le bouton envoie un message CAN à l'application pour lui indiquer de changer. L'application en mode pilote affichera quatre interfaces différentes en boucle.

EF11 - Le pilot doit pouvoir activer l'écran 1 a partir d'un bouton sur le volant

EF12 - Le pilot doit pouvoir activer l'écran 2 a partir d'un bouton sur le volant

EF02 - Le système doit permettre de changer quelques configurations directement dans les paramètres de l'application sur iOS. Les configurations doivent inclure, entre autres, le changement de mode entre pilote et ingénieur de piste ainsi que le changement des couleurs de l'interface de pâle à foncé.

EF13 - Le pilot doit pouvoir activer l'écran 3 a partir d'un bouton sur le volant

EF15 - Affichage en mode paysage pour le mode pilote

EF16 - Mettre en veille l'application après 3 secondes sans donnéesÉ

EF09 - Visualiser les alarmes et les capteurs sur l'interface pilote L'application doit afficher, de manière claire, les informations nécessaires au pilote en provenance des différents capteurs du véhicule. De plus, les messages d'alarmes seront aussi affichés de façon évidente et de façon à ce que le pilote les remarques immédiatement.

0.41 cancanEthernet

https://github.com/yvanross/log430-dashview-architecture/blob/master/src/main/java/dashview/Interfaces/ICancanRouter.java
Routeur du Server web wifi controlant le bus CanCan pour l'acquisition/diffusion des données des capteurs

Technologie: CanCan bus Router

Tags: Element,Component

Parent	Key	Category	Title	Priority
null	C03	null	can2Ethernet La librairie Can2Ethernet développée par le club Formule ÉTS doit être utilisée.	9
null	C05	Performance	fi UDP Les données sont envoyées par Wi-Fi via le protocole UDP et le bus CAN.	4

0.41.1 Perspectives

C03 - can2Ethernet La librairie Can2Ethernet développée par le club Formule ÉTS doit être utilisée.

C05 - Wifi UDP Les données sont envoyées par Wi-Fi via le protocole UDP et le bus CAN.

public interface ICancanRouter

this is a test for the cancanRouter

- **Author:** Yvan Ross

  void setMaximumOperationTemperature(double maximumOperationTemperature)

Temperature d'opération

- **Parameters:** `maximumOperationTemperature` — parametre

  void setMaximumRPM(double maxRPM)

Nombre de tours minues

- **Parameters:** `maxRPM` — en tours par minutes

  void setMaximumTireTemperature(double maxTireTemperature)

temperature des pneu

- **Parameters:** `maxTireTemperature` — parametre

  void setMaximumMotorTemperature(double maxTemp)

Temperature moteur

- **Parameters:** `maxTemp` — Temperature maximum du moteur en degré celcius

0.42 Display

Application cliente permettant d'affiche les informations au pilote durant la course

Tags: Element,Container

Parent	Key	Category	Title	Priority
null	C02	null	Usability Les configurations pour le mode par défaut et les couleurs de l'interface sont définies directement dans iOS.	2
null	C04	null	Objective-C L'application doit être en Objective-C sous la plateforme iOS 7 et est destinée aux iPod Touch de 5e génération avec un écran de 4 pouces	1
null	C07	null	Id et offset des capteurs Les « id » et les « offset » des capteurs doivent suivre la table CAN fournie.	2

null	C08	null	langue Les textes de l'application doivent être en anglais.	1
null	C09	null	deployment L'application doit être installée et exécutée sur un iPod Touch qui ne nécessite pas un « iOS jailbreaking »	4
null	EF02	Générale	Configuration de l'application dans les paramètres d'iOS Le système doit permettre de changer quelques configurations directement dans les paramètres de l'application sur iOS. Les configurations doivent inclure, entre autres, le changement de mode entre pilote et ingénieur de piste ainsi que le changement des couleurs de l'interface de pâle à foncé.	2
null	EF03	Générale	Gérer les données reçues en temps réel L'application doit constamment recevoir des données du bus CAN via Wi-Fi à partir du module Can2Ethernet et, en comparant avec la table des messages CAN, associer ces données aux capteurs et alarmes pour les afficher.	9
null	EF04	Générale	Afficher des couleurs spécifiques pour les RPM L'application doit afficher l'indicateur de RPM avec un code de couleur précis, soit de jaune à rouge en passant par une zone orange visible. De 3000 RPM à 15 000 RPM, l'indicateur doit être dans le spectre de jaune à rouge. À 15 000 RPM et plus, l'indicateur doit être rouge. De plus, le rouge doit changer selon un paramètre calculé par l'ACL lorsqu'un message correspondant au « id » de la table CAN de ce paramètre est reçu.	2
null	EF05	Générale	Afficher des couleurs spécifiques pour la température des pneus L'application doit afficher la température des trois capteurs de chaque pneu selon des couleurs spécifiques et avec des transitions fluides. Les capteurs sont situés à l'extérieur, au milieu et à l'intérieur de chacun des pneus. Lorsque la température est de 25°C et moins, la couleur est bleue. Entre 25°C et 65°C, la couleur passe de bleue à jaune. Entre 65°C et 95°C, l'indicateur passe de jaune à rouge. Finalement, en haut de 95°C, la couleur est rouge.	2
null	EF06	Générale	Afficher des couleurs spécifiques pour la température du moteur Le Dash Display doit afficher la température du moteur à l'aide de quatre couleurs. Lorsque la température est de 70°C et moins, l'indicateur doit être bleu. Entre 70°C et 90°C, la couleur utilisée est le vert. Entre 90°C et 100°C, l'indicateur doit être jaune et, finalement, il doit être rouge lorsque la température dépasse le 100°C.	2
null	EF07	Générale	Afficher des couleurs spécifiques pour la puissance de la batterie Le système doit afficher la puissance de la batterie à l'aide de deux couleurs. Lorsque la batterie est à la puissance maximale (14V) jusqu'à 11.5V, le fond du capteur est vert. Lorsque la puissance atteint 11.5 V et en dessous, le fond devient rouge.	2
null	EF08	Générale	Afficher des couleurs spécifiques pour les alertes Le Dash Display doit afficher les alertes de deux façons selon leur statut. Dans tous les cas, elles sont affichées en rouge. Lorsqu'elles sont en cours, elles ont une opacité de 100 %. Sinon, l'opacité diminue à 30 %.	2
null	EF09	Mode pilote	Visualiser les alarmes et les capteurs sur l'interface pilote L'application doit afficher, de manière claire, les informations nécessaires au pilote en provenance des différents capteurs du véhicule. De plus, les messages d'alarmes seront aussi affichés de façon évidente et de façon à ce que le pilote les remarques immédiatement.	6
null	EF15	Mode pilote	Affichage en mode paysage pour le mode pilote doit afficher les interfaces en mode pilote sous le format paysage.	1
null	EF16	Mode pilote	Mettre en veille l'application après 3 secondes sans données doit se mettre en veille si une interruption de données survient et si elle dure plus de trois secondes.	1
null	ENF01	Usability	Utilisation du visuel de façon intuitive L'interface doit respecter le fonctionnement natif d'iOS lorsque les diverses actions sont effectuées dans le mode ingénieur de piste. Par exemple, l'ingénieur de piste doit entrer en mode édition pour supprimer ou déplacer une alarme ou un capteur.	6
null	ENF02	Usability	Utilisation du mode pilote doit être très simple Les différentes interfaces du mode pilote doivent être simples, claires et précises. Lorsque la voiture est en piste, le pilote ne doit pas avoir à réfléchir pour comprendre et utiliser l'application. Les quatre interfaces disponibles doivent être toutes visibles en trois clics du bouton situé sur le	2

		volant puis continuer de cette façon en boucle.	
null	ENF03 Usability	Démarrage simple et rapide dans le mode configuré Lorsque le Dash Display démarre, il ne doit pas y avoir d'attente ou de commande à effectuer pour que l'application puisse être utilisée. Cette exigence est surtout importante pour le mode pilote qui ne peut pas utiliser l'écran tactile. De plus, le système doit utiliser le mode choisi dans les configurations de l'application directement dans iOS.	4
null	ENF04 Usability	Haut contraste dans les couleurs de l'interface Les couleurs de l'interface doivent avoir de très haut contraste. De plus, deux versions de couleurs doivent être disponibles dans la configuration de l'application sur iOS : un mode foncé et un mode pâle.	1
null	ENF05 Performance	Rafraîchissement de l'écran à une cadence de 10 Hz La fréquence de rafraîchissement des interfaces est de 10 Hz afin que les données affichées soient toujours à jour en temps réel. Avec cette cadence, le véhicule a le temps de transmettre les nouvelles données par Wi-Fi. Cette mesure signifie 10 fois par seconde donc le rafraîchissement est d'une fois à chaque 100 ms.	4
null	ENF06 Disponibility	Période d'utilisation d'au maximum 25 minutes Les courses ont une durée d'au maximum 25 minutes donc l'application doit être optimisée pour une utilisation sans problème pour ce délai de temps.	1
null	ENF08 Disponibility	Aucun redémarrage de l'application en cas d'erreur Lorsqu'une erreur survient, l'application ne doit pas redémarrer seule. Il faut la redémarrer manuellement à chaque fois.	9

0.42.1 Perspectives

EF02 - Configuration de l'application dans les paramètres d'iOS Le système doit permettre de changer quelques configurations directement dans les paramètres de l'application sur iOS. Les configurations doivent inclure, entre autres, le changement de mode entre pilote et ingénieur de piste ainsi que le changement des couleurs de l'interface de pâle à foncé.

C02 - Usability Les configurations pour le mode par défaut et les couleurs de l'interface sont définies directement dans iOS.

EF03 - Gérer les données reçues en temps réel L'application doit constamment recevoir des données du bus CAN via Wi-Fi à partir du module Can2Ethernet et, en comparant avec la table des messages CAN, associer ces données aux capteurs et alarmes pour les afficher.

EF04 - Afficher des couleurs spécifiques pour les RPM L'application doit afficher l'indicateur de RPM avec un code de couleur précis, soit de jaune à rouge en passant par une zone orange visible. De 3000 RPM à 15 000 RPM, l'indicateur doit être dans le spectre de jaune à rouge. À 15 000 RPM et plus, l'indicateur doit être rouge. De plus, le rouge doit changer selon un paramètre calculé par l'ACL lorsqu'un message correspondant au « id » de la table CAN de ce paramètre est reçu.

EF15 - Affichage en mode paysage pour le mode pilote doit afficher les interfaces en mode pilote sous le format paysage.

C04 - Objective-C L'application doit être en Objective-C sous la plateforme iOS 7 et est destinée aux iPod Touch de 5e génération avec un écran de 4 pouces

EF05 - Afficher des couleurs spécifiques pour la température des pneus L'application doit afficher la température des trois capteurs de chaque pneu selon des couleurs spécifiques et avec des transitions fluides. Les capteurs sont situés à l'extérieur, au milieu et à l'intérieur de chacun des pneus. Lorsque la température est de 25°C et moins, la couleur est bleue. Entre 25°C et 65°C, la couleur passe de bleue à jaune. Entre 65°C et 95°C, l'indicateur passe de jaune à rouge. Finalement, en haut de 95°C, la couleur est rouge.

EF16 - Mettre en veille l'application après 3 secondes sans données doit se mettre en veille si une interruption de données survient et si elle dure plus de trois secondes.

EF06 - Afficher des couleurs spécifiques pour la température du moteur Le Dash Display doit afficher la température du moteur à l'aide de quatre couleurs. Lorsque la température est de 70°C et moins, l'indicateur doit être bleu. Entre 70°C et 90°C, la couleur utilisée est le vert. Entre 90°C et 100°C, l'indicateur doit être jaune et, finalement, il doit être rouge lorsque la température dépasse le 100°C.

EF07 - Afficher des couleurs spécifiques pour la puissance de la batterie Le système doit afficher la puissance de la batterie à l'aide de deux couleurs. Lorsque la batterie est à la puissance maximale (14V) jusqu'à 11.5V, le fond du capteur est vert. Lorsque la puissance atteint 11.5 V et en dessous, le fond devient rouge.

EF08 - Afficher des couleurs spécifiques pour les alertes Le Dash Display doit afficher les alertes de deux façons selon leur statut. Dans

tous les cas, elles sont affichées en rouge. Lorsqu'elles sont en cours, elles ont une opacité de 100 %. Sinon, l'opacité diminue à 30 %.

C08 - langue Les textes de l'application doivent être en anglais.

EF09 - Visualiser les alarmes et les capteurs sur l'interface pilote L'application doit afficher, de manière claire, les informations nécessaires au pilote en provenance des différents capteurs du véhicule. De plus, les messages d'alarmes seront aussi affichés de façon évidente et de façon à ce que le pilote les remarques immédiatement.

C07 - Id et offset des capteurs Les « id » et les « offset » des capteurs doivent suivre la table CAN fournie.

C09 - deployment L'application doit être installée et exécutée sur un iPod Touch qui ne nécessite pas un « iOS jailbreaking »

ENF01 - Utilisation du visuel de façon intuitive L'interface doit respecter le fonctionnement natif d'iOS lorsque les diverses actions sont effectuées dans le mode ingénieur de piste. Par exemple, l'ingénieur de piste doit entrer en mode édition pour supprimer ou déplacer une alarme ou un capteur.

ENF02 - Utilisation du mode pilote doit être très simple Les différentes interfaces du mode pilote doivent être simples, claires et précises. Lorsque la voiture est en piste, le pilote ne doit pas avoir à réfléchir pour comprendre et utiliser l'application. Les quatre interfaces disponibles doivent être toutes visibles en trois clics du bouton situé sur le volant puis continuer de cette façon en boucle.

ENF03 - Démarrage simple et rapide dans le mode configuré Lorsque le Dash Display démarre, il ne doit pas y avoir d'attente ou de commande à effectuer pour que l'application puisse être utilisée. Cette exigence est surtout importante pour le mode pilote qui ne peut pas utiliser l'écran tactile. De plus, le système doit utiliser le mode choisi dans les configurations de l'application directement dans iOS.

ENF04 - Haut contraste dans les couleurs de l'interface Les couleurs de l'interface doivent avoir de très haut contraste. De plus, deux versions de couleurs doivent être disponibles dans la configuration de l'application sur iOS : un mode foncé et un mode pâle.

ENF05 - Rafraîchissement de l'écran à une cadence de 10 Hz La fréquence de rafraîchissement des interfaces est de 10 Hz afin que les données affichées soient toujours à jour en temps réel. Avec cette cadence, le véhicule a le temps de transmettre les nouvelles données par Wi-Fi. Cette mesure signifie 10 fois par seconde donc le rafraîchissement est d'une fois à chaque 100 ms.

ENF06 - Période d'utilisation d'au maximum 25 minutes Les courses ont une durée d'au maximum 25 minutes donc l'application doit être optimisée pour une utilisation sans problème pour ce délai de temps.

ENF08 - Aucun redémarrage de l'application en cas d'erreur Lorsqu'une erreur survient, l'application ne doit pas redémarrer seule. Il faut la redémarrer manuellement à chaque fois.

public interface ICancanRouter

this is a test for the cancanRouter

- **Author:** Yvan Ross

  void setMaximumOperationTemperature(double maximumOperationTemperature)

Temperature d'opération

- **Parameters:** `maximumOperationTemperature` — parametre

  void setMaximumRPM(double maxRPM)

Nombre de tours minues

- **Parameters:** `maxRPM` — en tours par minutes

  void setMaximumTireTemperature(double maxTireTemperature)

temperature des pneu

- **Parameters:** `maxTireTemperature` — parametre

  void setMaximumMotorTemperature(double maxTemp)

Temperature moteur

- **Parameters:** `maxTemp` — Temperature maximum du moteur en degré celcius

public interface ICtrlEngineer

control engineer functions

`void setMaximumOperationTemperature(double maximumOperationTemperature)`

maximum operation temperature of the vehicule

- **Parameters:** `maximumOperationTemperature` — vehicule temp

  void setMaximumRPM(double maxRPM)

maximum rpm of the vehicule

- **Parameters:** `maxRPM` — max rpm in km/h

  void setMaximumTireTemperature(double maxTireTemperature)

max tire temperature

- **Parameters:** `maxTireTemperature` — in degre celcius

  void setMaximumBatteryPower(double maxBatteryPower)

maximum battery power

- **Parameters:** `maxBatteryPower` — in watts

  void setMaximumMotorTemperature(double maxMotorTemperature)

maximum motor temperature

- **Parameters:** `maxMotorTemperature` — in celcius

  public class ICtrlPilot

controller pilot class

`void getDataInterface(int userInterfaceNumber)`

get user interfave by number

- **Parameters:** `userInterfaceNumber` — between 1..4

  public interface IExample

Iexample pour la documentation détaillé d'une interface que je veux lier à un lien dans Structurizr

`public static final double CONSTANT_PI = 3.1416`

definition d'une constante

`public String getPilotName()`

getPilotName

- **Returns:** Get the name of the pilot

  public String getPilotWeight()

getPilotWeight

- **Returns:** weight of the pilot

  public void setPilotWeightLB(float weightInLbs) throws ExceptionPilotOverWeight

setPilotWeightLB in livre

- **Parameters:** `weightInLbs` — of the pilot [EF01] @EF01 related to requirement

- **Exceptions:** `ExceptionPilotOverWeight` — pilot in excess of weight

public void setExigence(Requirement requirement)

set related exigenceexigence

- **Parameters:** `requirement` — is an instance of dashview.Requirement

public interface IExampleV2

Iexample pour la documentation détaillé d'une interface que je veux lier à un lien dans Structurizr

```
public static final double CONSTANT_PI2 = 6.28
```

definition d'une constante

```
public String getPilotFirstName()
```

- **Returns:** Get the name of the pilot

public String getPilotWeightV2()

- **Returns:** weight of the pilot in kilogram

public void setPilotWeightV2(float weightInLbs) throws ExceptionPilotOverWeight

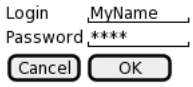
- **Parameters:** `weightInLbs` — of the pilot
- **Exceptions:** `dashview.Interfaces.ExceptionPilotOverWeight` — pilote dépasse le poids reglemenntaire

public void setExigenceV2(Requirement requirement)

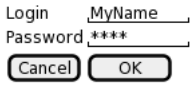
- **Parameters:** `requirement` — is an instance of dashview.Requirement

0.43 Login Page

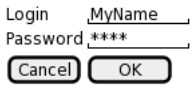
0.44 Pilot Interface #1



0.45 Pilot Interface #2



0.46 Pilot Interface #3



0.47 Pilot Interface #4