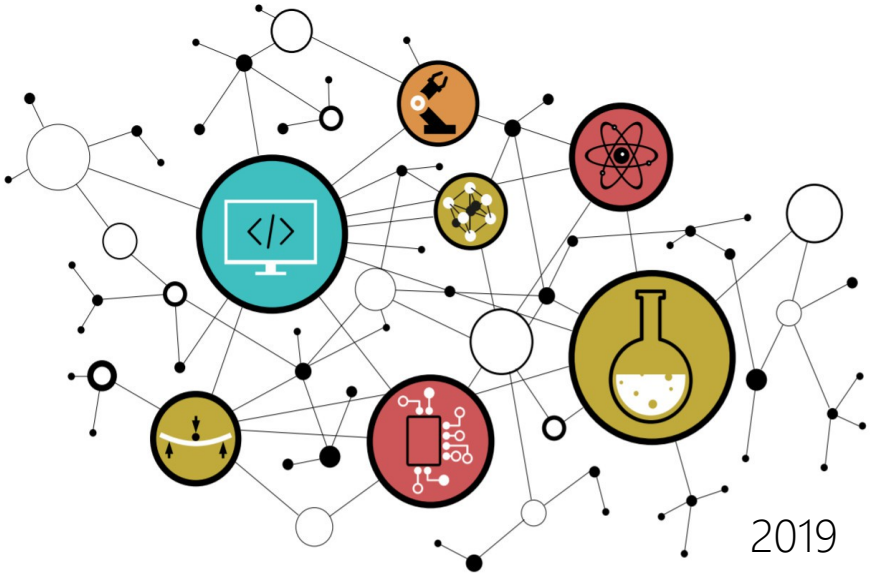
SPECIFICATIONS TECHNIQUES DE BESOIN – v1.0



2019

CONTACTS

Établissement

ENSICAEN
6 boulevard Maréchal Juin
CS 45053
14050 CAEN cedex 04

Référents

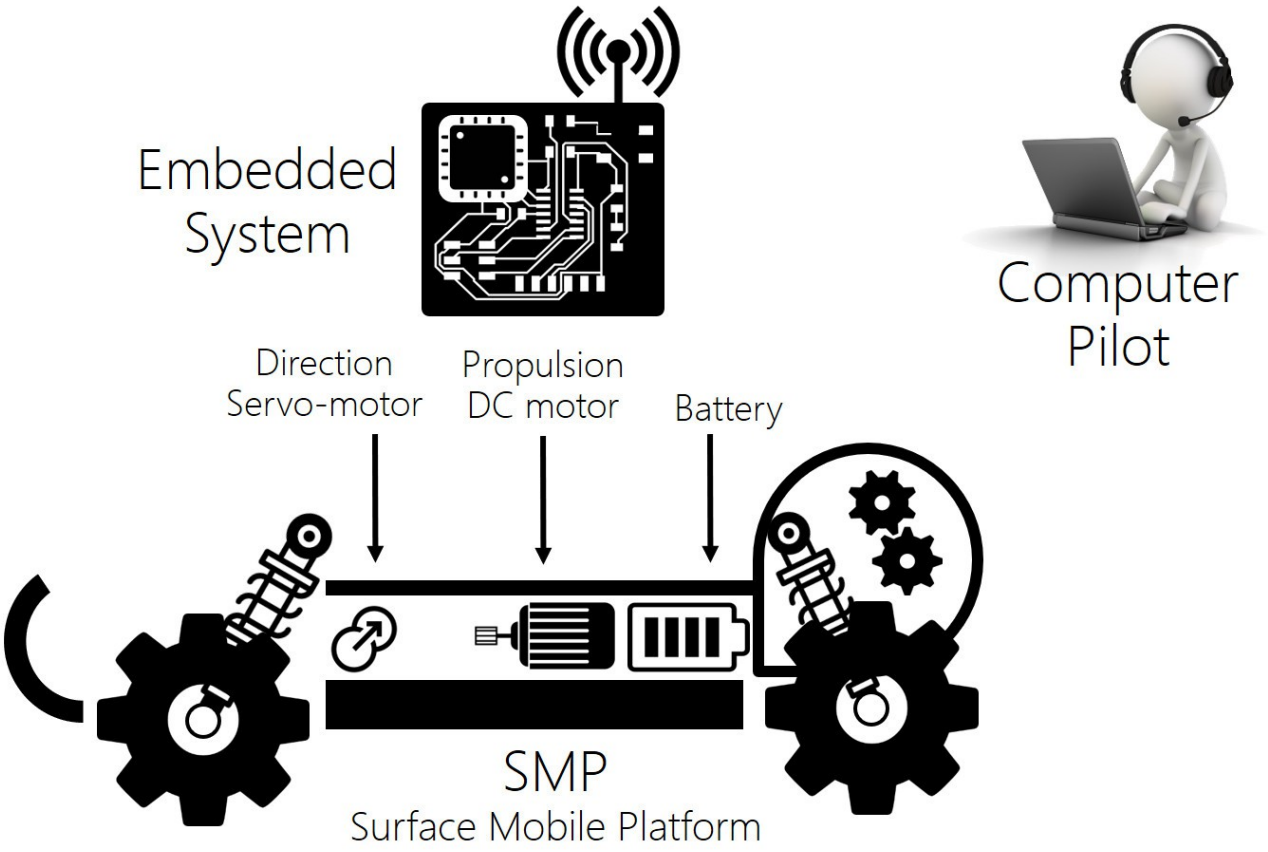
- hugo descoubes – Chef de projet MOA et Directeur Technique Systèmes Embarqués - hugo.descoubes@ensicaen.fr
- Philippe Lefebvre – Directeur Technique Réseau - philippe.lefebvre@ensicaen.fr
- Emmanuel Cagniot – Directeur Technique Logiciel - emmanuel.cagniot@ensicaen.fr
- Olivier Clouard – Ingénieur responsable du Service Technique Électronique - olivier.clouard@ensicaen.fr
- Miloud Frikel - Responsable majeure SATE - miloud.frikel@ensicaen.fr

SOMMAIRE

1. PRESENTATION DU PROJET RACE

1. PRESENTATION DU PROJET RACE

1.1. Besoins et objectifs



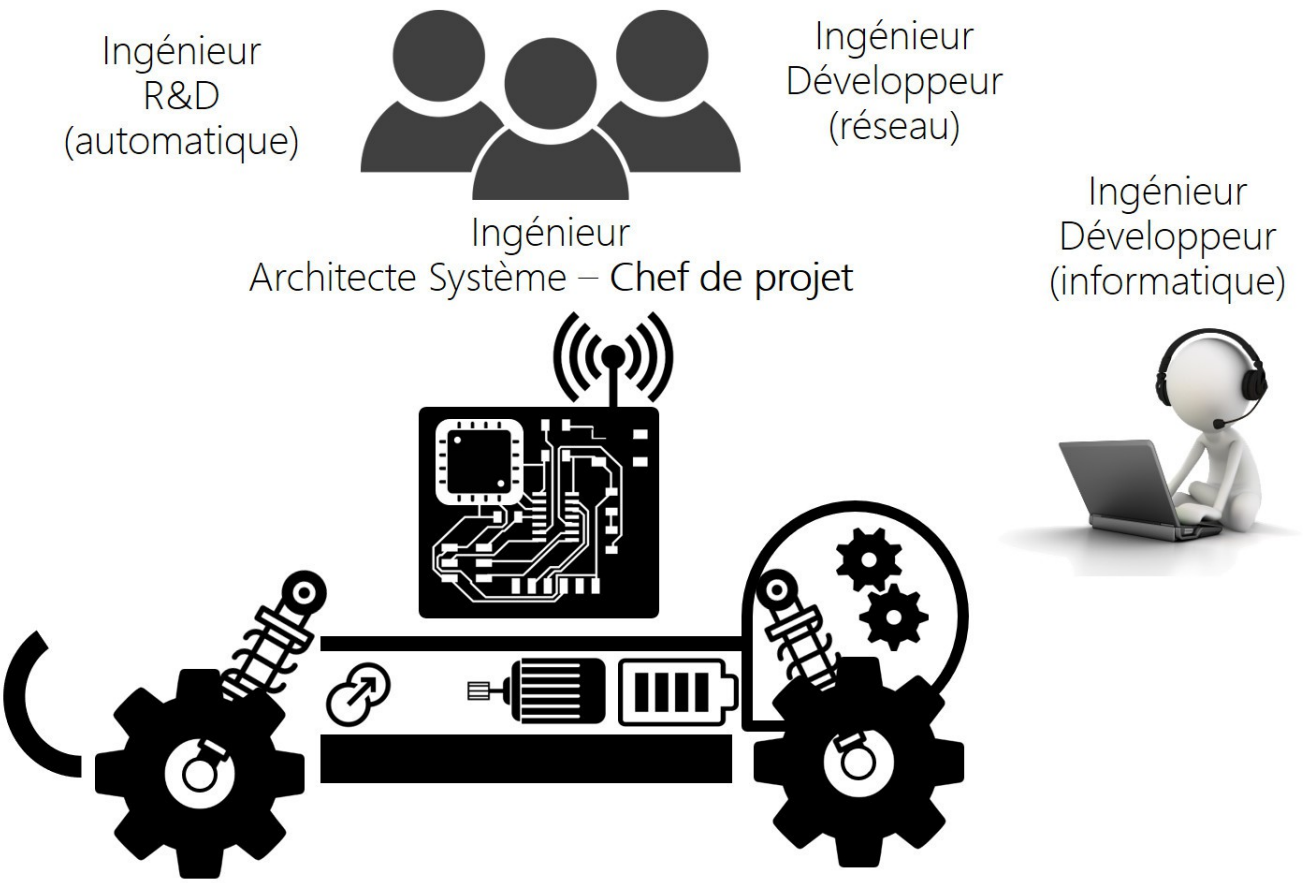
Besoins et pédagogie

Ce projet de spécialisation en majeure SATE (Signal, Automatique, Télécoms et systèmes Embarqués) a pour objectif de lier les compétences enseignées dans le plan de formation en spécialité Électronique et Physique Appliquée et en majeure SATE depuis l'entrée à l'école. Il fait notamment référence aux domaines suivants :

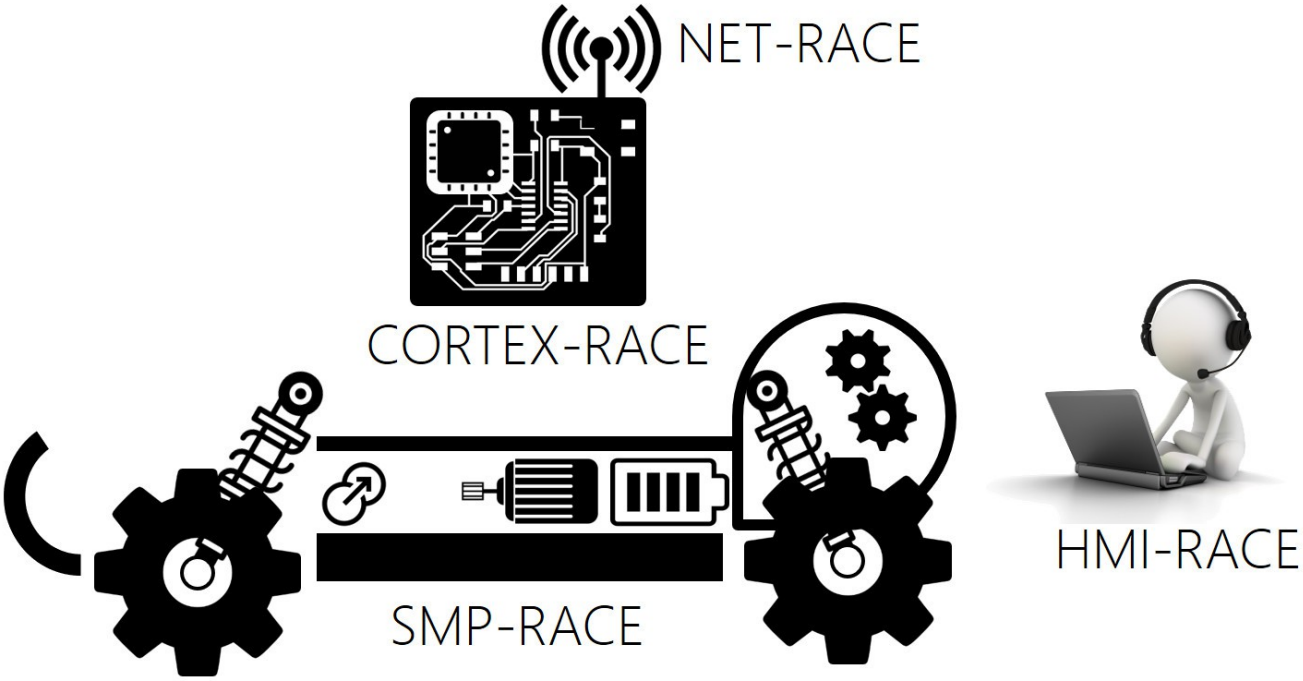
- Systèmes embarqués et électronique numérique
- Systèmes d'exploitation temps réel
- Programmation procédurale en langage C
- Programmation orientée objet C++ et modélisation
- Réseaux de communication IP
- Automatique, identification, modélisation et régulation de procédé

Objectifs

L'objectif scientifique et technique du projet RACE (Remote Automotive Challenge of ENSICAEN) est de développer un firmware (micrologiciel) embarqué assurant la supervision (système) et le contrôle (automatique) d'une voiture télécommandée depuis un ordinateur. Les développements porteront sur la modélisation, la conception et le développement en langage C de l'application embarquée, ainsi que sur le développement de l'application sur ordinateur en C++. L'équipe projet sera composée de 4 élèves ingénieurs en formation. 3 élèves seront chargés du développement du logiciel embarqué et 1 de l'application sur ordinateur. L'élève ingénieur en charge de la conception de l'architecture logicielle embarquée jouera également le rôle de chef de projet technique et chef de projet MOE (Maîtrise d'œuvre). Il aura notamment à planifier et séquencer l'ensemble des développements, initier les rencontres de direction technique avec le MOA (Maîtrise d'Ouvrage ou Client), piloter voire assurer les phases d'intégration ainsi que garantir les phases de test et de validation des solutions déployées.



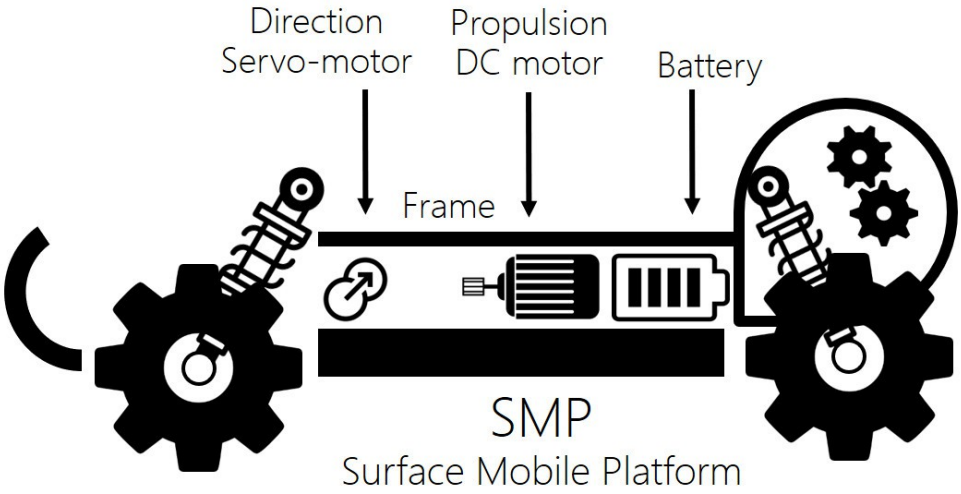
1.2. Fonctions techniques



Cette section du document a pour objectif de poser le jeu de vocabulaire associé aux différentes fonctions métier techniques du projet RACE. Les spécifications techniques de besoins, explicitant notamment les exigences et contraintes physiques comme technologiques suivront la partie présentation. Cette section du document a été pensée et est découpée selon la structure même du document de STB (Spécifications Techniques de Besoin). Elle reflète les grands pans des domaines de l'ingénierie et les différents organes physiques du projet RACE :

- Système mécanique et énergétique - SMP-RACE
- Système embarqué – CORTEX-RACE et NET-RACE
- Application sur ordinateur - HMI-RACE

Système mécanique et énergétique



Le système électronique a été pensé pour s'adapter à un très grand nombre de véhicules télécommandés du marché. L'ensemble des solutions mécaniques et énergétique d'emport sera nommée SMP-RACE (Surface Mobile Platform) ou Plateforme Mobile de Surface. Ses principaux éléments constitutifs sont présentés ci-dessous :

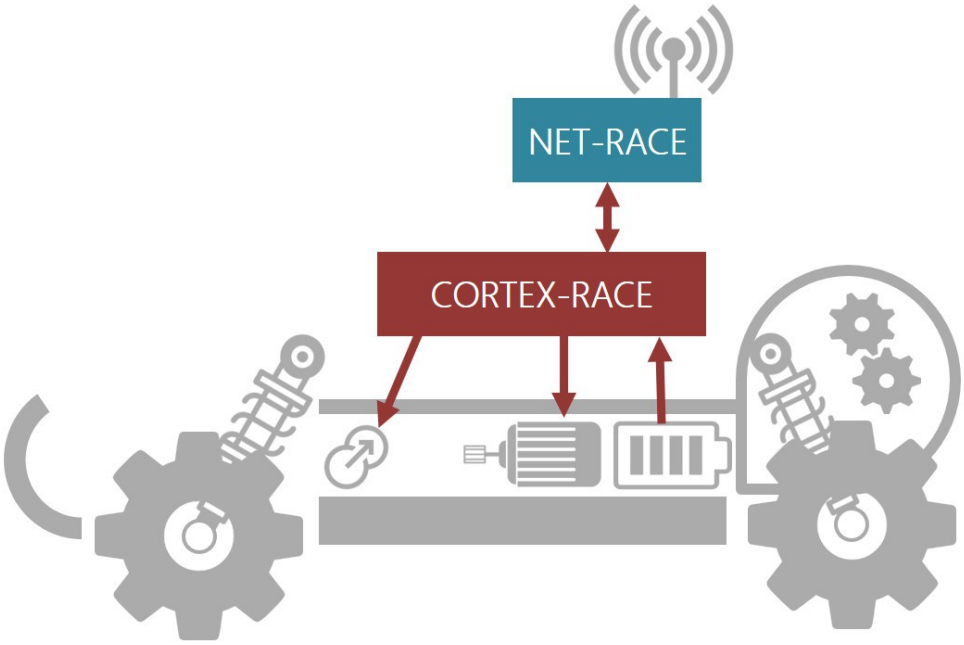
- Châssis mécanique (frame) : chargé de l'emport de l'ensemble des actionneurs (servomoteur de direction et moteur de propulsion), de la batterie assurant le stockage d'énergie électrique du système et de l'électronique embarquée de supervision et de contrôle

- Servomoteur de direction : chargé de la direction du véhicule. Seules les roues avant sont directrices

- Moteur DC de propulsion : chargé de la propulsion du véhicule. Les 4 roues sont motrices

- Batterie : chargée du stockage d'énergie électrique pour le fonctionnement de l'électronique embarquée et de l'électronique de puissance

Système embarqué

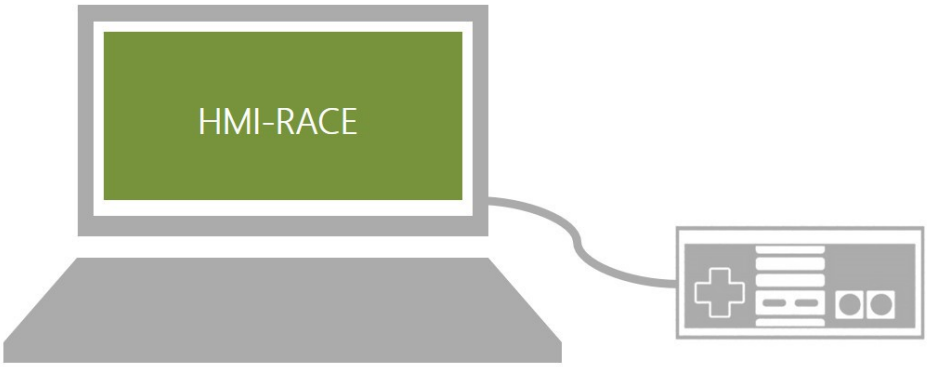


Cette partie présente le système embarqué (électronique et informatique embarquées) enfouis dans le SMP et assurant la supervision et le contrôle du système. Observons les fonctions embarquées :

- NET-RACE (Network RACE) : gestion du point d'accès réseau WIFI déployé par le véhicule et accessible depuis n'importe quel appareil (ordinateur, tablette, smartphone, etc) possédant une carte réseau WIFI. Le contrôle du système ne sera néanmoins possible qu'après installation de l'application sur la plateforme host (maître) de commande

- CORTEX-RACE : sous système électronique et informatique embarqué assurant la supervision du système, les contrôles de la direction et de la propulsion par algorithme automatique de régulation, ainsi que la communication réseau.

Application sur ordinateur



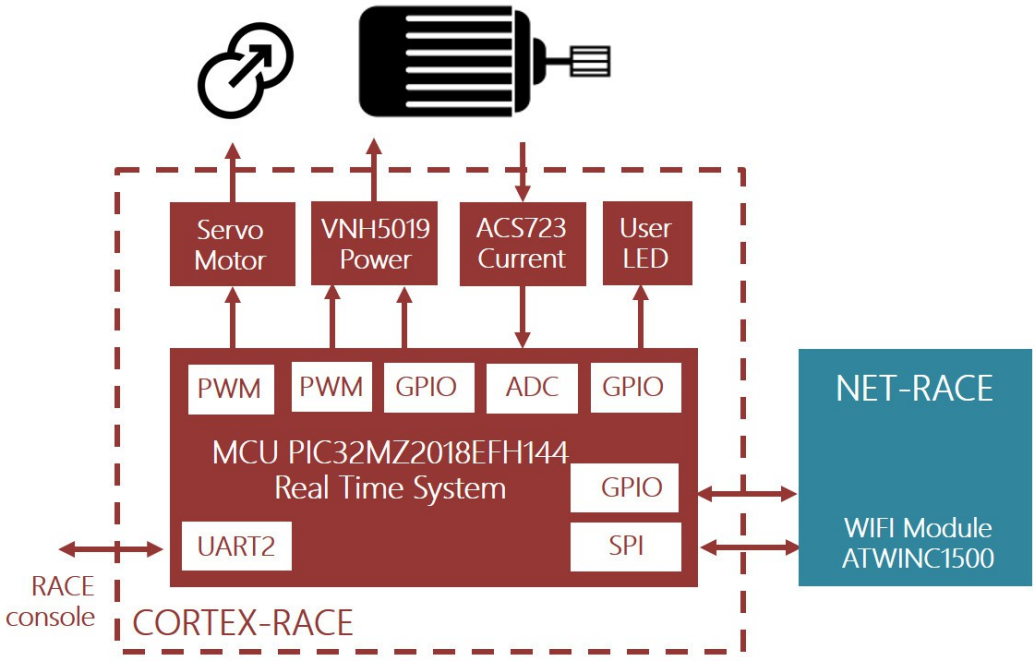
Cette partie présente l'ensemble des interfaces utilisateur pour le pilote (retours visuels et gestion des interfaces de commande) :

- HMI-RACE (Human Machine Interface ROV) : sous système informatique de commande et d'interface du SMP. L'application logicielle HMI-RACE sera chargée d'afficher la consommation d'énergie en temps réel et de rediriger les ordres de direction et de propulsion vers le système embarqué NET-RACE/CORTEX-RACE

Synthèse des fonctions techniques

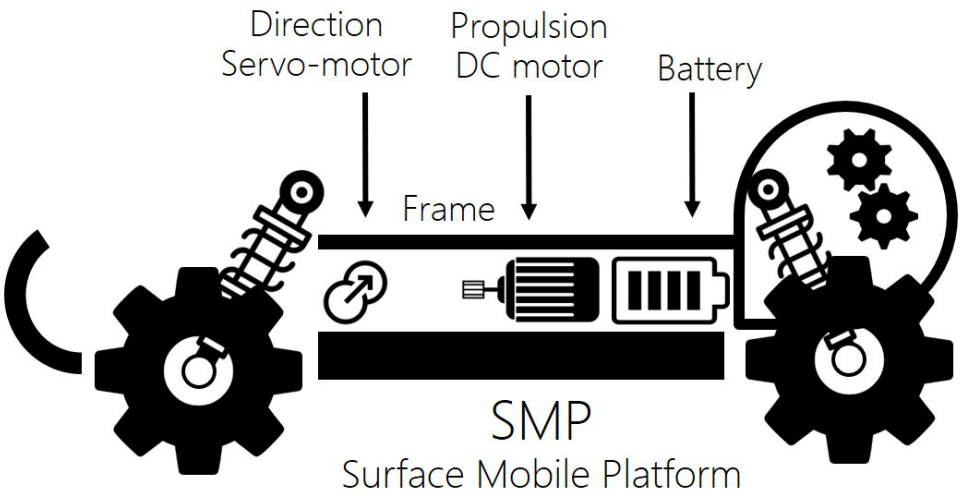
1.3. Schéma fonctionnel

La représentation fonctionnelle ci-dessous schématise les différentes fonctionnalités capteurs, actionneurs, instrumentales, chemins et directions de l'information, passerelles de communication, entités de contrôle et de supervision du système embarqué. Le système sera présenté et spécifié en détail dans la suite du document de STB.



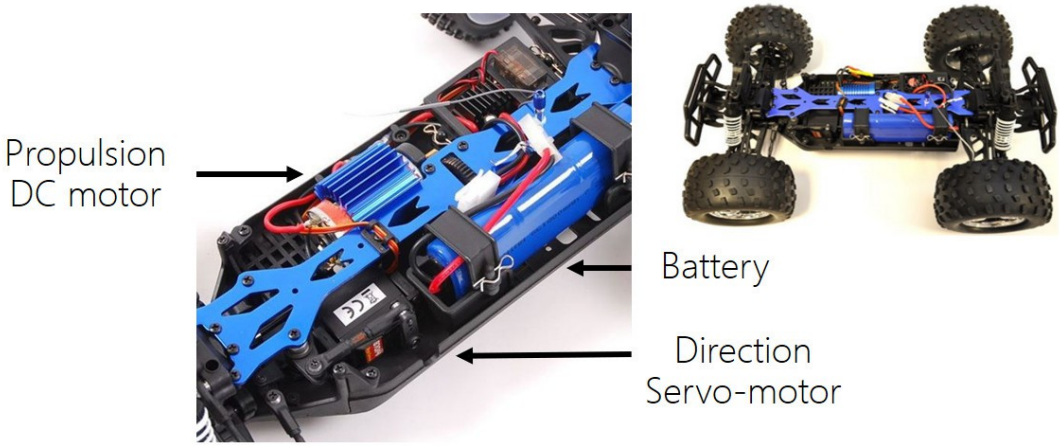
2. SYSTEME MECANIQUE ET ENERGETIQUE

2. SYSTEME MECANIQUE ET ENERGETIQUE



2.1. Plateforme Mobile de Surface - SMP-RACE

La plateforme sélectionnée afin de réaliser le SMP-RACE est une solution du marché, un Digger Truggy RTR 4x4. L'électronique initialement embarquée a été remplacée par le système CORTEX-RACE/NET-RACE. Ce système a été pensé pour s'adapter sur la grande majorité des solutions à moteur DC du marché grâce notamment à son module de puissance VNH5019-E supportant 30A maximal et 12A nominal en charge.



Caractéristiques

- Poids : 1,6Kg
- Motorisation : Brushed 550 inclus 15T (plus de 50Km/h en pointe)
- Servomoteur : pignons métal
- Empattement : 305mm
- Voies Avant/Arrière : 302mm

2.2. Stockage d'énergie électrique

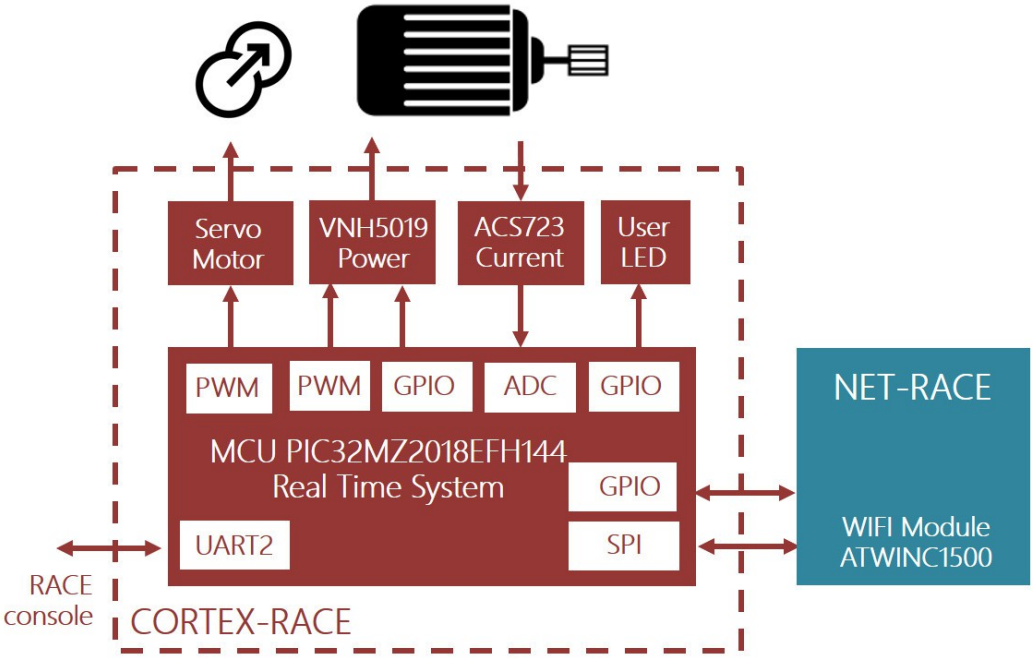
Le stockage d'énergie électrique sera assuré par voie électrochimique à travers l'utilisation d'une batterie NiMh de 4600mAh sous 7,2V (6 éléments de 1,2V). D'autres batteries offrant des capacités de stockage différentes pourront éventuellement être utilisées et fournies. Toutes les connectiques de puissance seront assurées avec des solutions Dean-T mâle/femelle.

Caractéristiques

- Type : NiMh
- Tension : 7,2V
- Dimensions : 135x47x25mm
- Capacité : 4600mAh (ou autres)

3. SYSTEME EMBARQUE

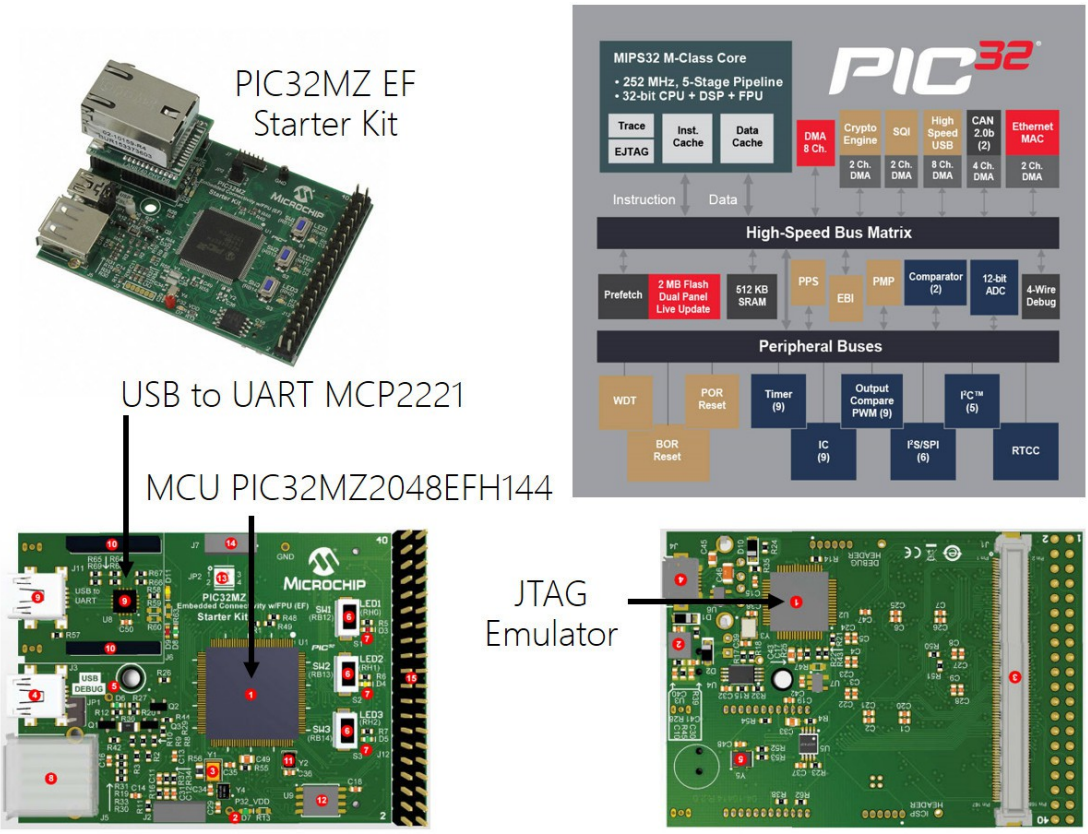
3. SYSTEME EMBARQUE



Nous allons amorcer cette partie par présenter les composantes matérielles, logicielles, outils et règles communes au développement du système embarqué CORTEX-RACE/NET-RACE :

- MCU 32bits Microchip (MIPS core) PIC32MZ2048EFH144
- RTOS (Real Time Operating System) FreeRTOS
- Interface série par UART de prototypage et de test
- IDE (Integrated Development Environment) MPLABX
- Framework de développement Harmony

MCU 32bits Microchip (MIPS core) PIC32MZ2048EFH144



Le processeur porté pour le projet RACE est un MCU haut de gamme proposé par Microchip et embarquant un cœur MIPS Warrior M-Class travaillant à 252MHz (jusqu'à 415DMIPS), des unités matérielles de calcul DSP (Digital Signal Processing), une unité de calcul flottante FPU (single/double précision), 2Mo de mémoire flash, 512 Ko de SRAM et de nombreuses interfaces périphériques. Cela permettra donc si nécessaire l'emport d'une algorithmique de régulation avancée implémentée au format flottant simple précision. La carte mère du projet a été pensée pour s'interfacer avec la carte de développement PIC32MZ EF Starter Kit proposé également par Microchip.

RTOS (Real Time Operating System) FreeRTOS

L'application logicielle embarquée sera portée sur le système d'exploitation temps réel FreeRTOS v9.0.0. Rappelons que FreeRTOS est la solution leader du marché des RTOS pour l'embarqué à notre époque et qu'il reste protégé par une licence MIT.

Interface série par UART de prototypage et de test

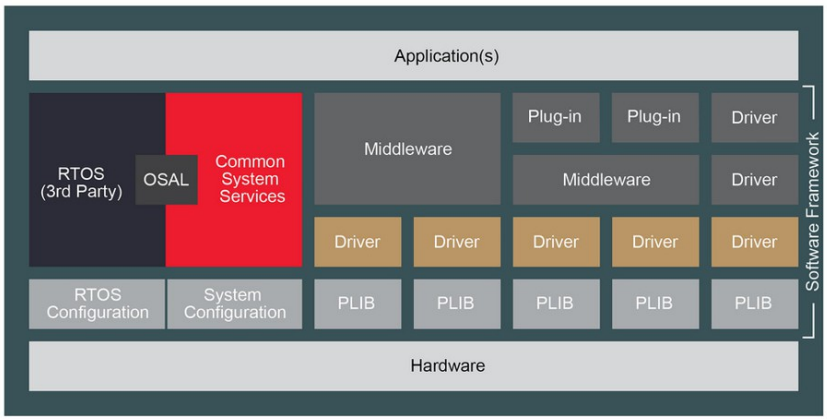
Le système CORTEX-RACE proposera également une interface UART utilisé uniquement durant les phases de prototypage, de développement et de test. Cette interface a pour objectif d'offrir aux équipes de développement une solution de communication (commande et diagnostique) avec des fonctions métiers accessibles durant le développement, et donc avant l'intégration et le déploiement du réseau NET-RACE. Un interpréteur de commande sera développé avec un jeu de commandes spécifiques aux besoins du projet. Observons l'interface standard retournée au démarrage du système. Le jeu de commandes sera précisé dans la suite des spécifications techniques. Observons une séquence retournée par la console série. Les parties entre chevrons seront à adapter.

```
ENSICAEN
RACE project - <date of the last release>
RACE # version
RACE # help
<command name>  -  <short description>
<command name>  -  <short description>
<command name>  -  <short description>
...
RACE # 42
unrecognized command, try help
RACE #
```

La carte PIC32MZ EF Starter Kit embarque un module bridge USB-to-UART MCP221 développé par Microchip. Précisons néanmoins que l'un des principaux acteurs du marché des bridges USB-to-UART reste FTDI, un fondeur spécialisé.

IDE (Integrated Development Environment) MPLABX

Framework de développement Harmony

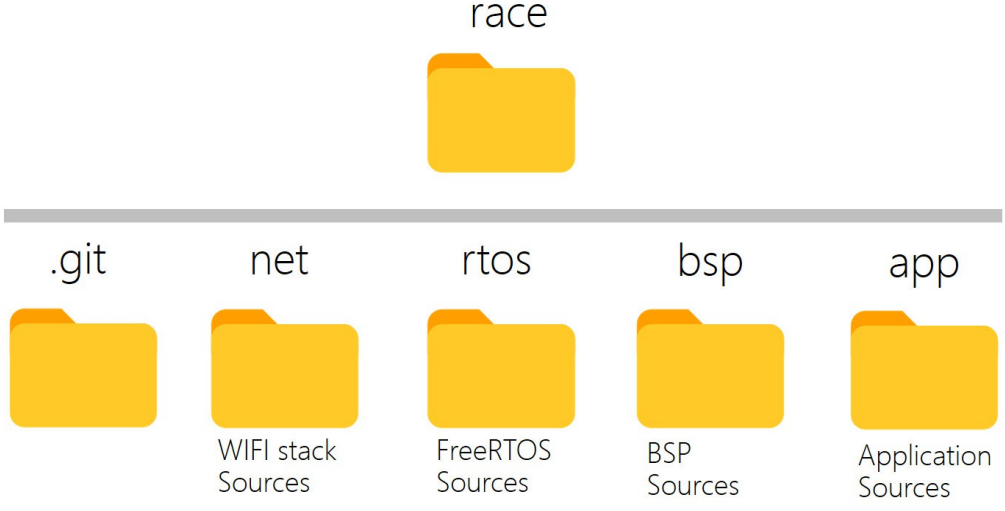


Harmony Framework

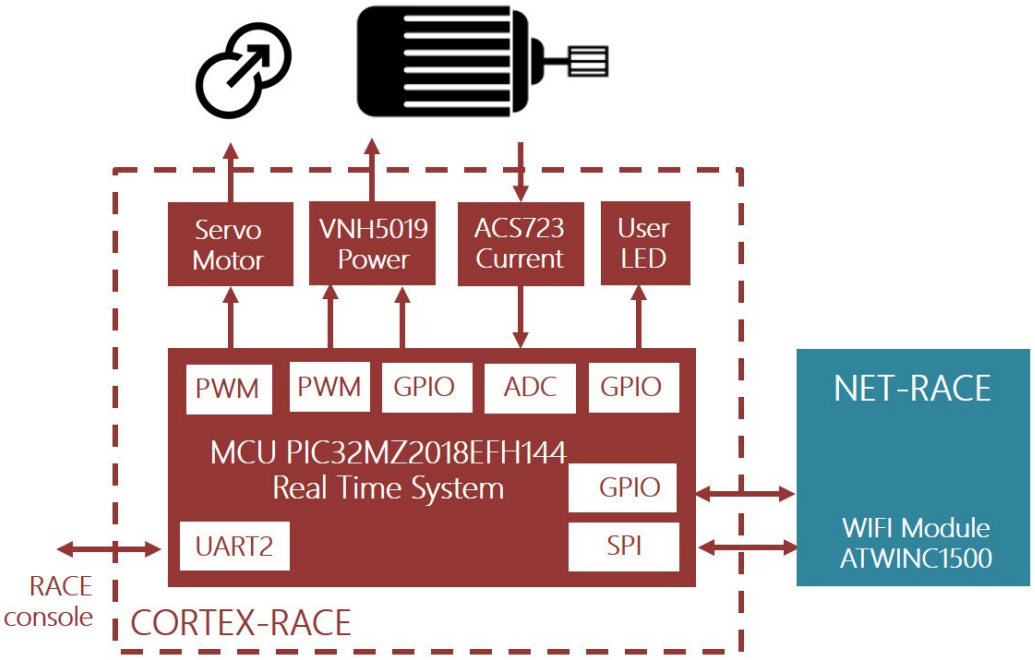
Les développements à destination du PICM32MZ seront réalisés sous IDE MPLABX (v4.15 ou version supérieure) proposé par Microchip. Cet IDE utilise un framework Netbeans et est libre d'utilisation. La configuration du BSP (Board Support Package), l' intégration de la stack WIFI ainsi que celle du RTOS FreeRTOS se feront impérativement en utilisant le framework Harmony v2.06. De même, la chaîne de compilation XC32 sera figée à la version 2.15 et sera utilisée en version gratuite (niveau d'optimisation -O1 préconisé pour utiliser Harmony).

Arborescence du projet et coding style

Le projet devra ''si possible'' respecter l'arborescence présentée ci-dessous. Un coding style C ENSICAEN sera fourni et à respecter impérativement. Chaque ingénieur devra explicitement préciser ses noms, prénoms et coordonnées mail sur chaque fichier développé ou modifié.



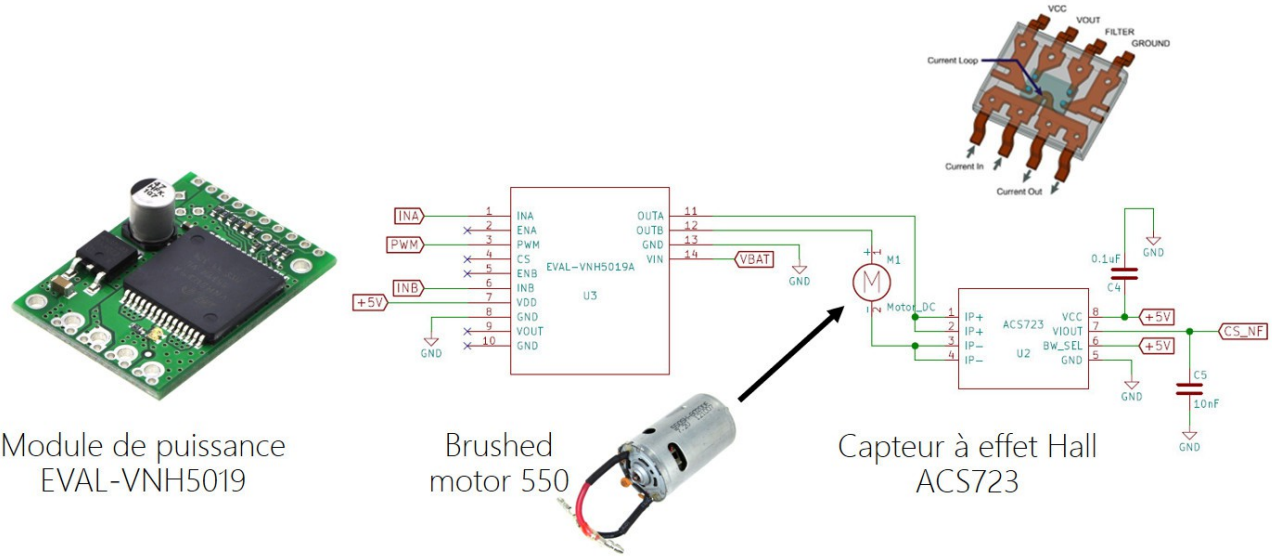
3.1. Propulsion et direction - SMP-RACE



Cette partie s'attache dans un premier temps à présenter l'architecture et les solutions matérielles retenus pour le système CORTEX-RACE. Le sous système NET-RACE sera présenté dans un second temps. Le firmware embarqué sera démarré par connexion physique de la batterie au système. Tant que la batterie sera connectée et chargée, l'application embarquée sera exécutée. Le système recevra les ordres (consignes de propulsion et direction) toutes les 20ms (fréquence de 50Hz) depuis l'ordinateur. Néanmoins, il devra garantir un déterminisme temps réel dur quant à l'exécution de l'algorithme de commande (acquisition ADC, calcul de la sortie du régulateur puis commande PWM) ainsi que pour la commande de direction. La formalisation des attentes client concernant l'applicatif temps réel embarqué sera réalisée dans cette section du document de STB. Présentation du découpage de l'étude :

- Module de puissance et mesure de courant
- Carte mère ENSICAEN
- Application logicielle embarquée
  - Identification, modélisation et régulation du procédé
  - Interface série de test et de prototypage
  - Bibliothèque réseau
  - Architecture du système logiciel embarqué

Module de puissance et mesure de courant



Le module de puissance retenu pour le design est la solution VNH5019 proposée par STMicroelectronics, une solution dédiée à la commande de moteur à courant continu (DC) absorbant un courant maximal en charge allant jusqu'à 30A.

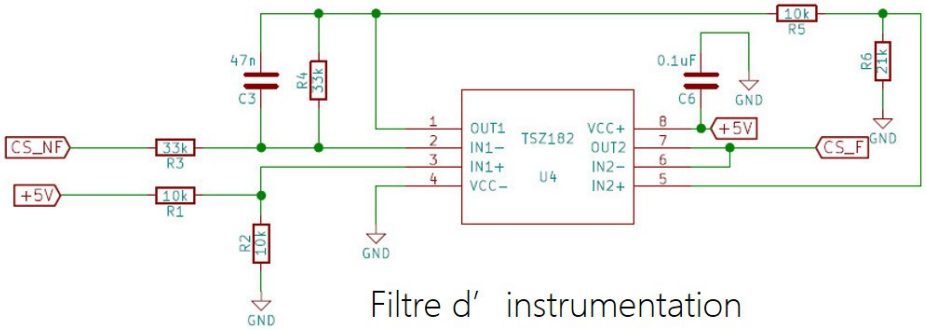
Module de puissance VNH5019 :

- Tension d'entrée : de 5,5V à 24V
- Intensité nominale de sortie : 12A en continu (30A max)
- Fréquence PWM : 20KHz

Capteur de courant à effet Hall AC723 (AC/DC) :

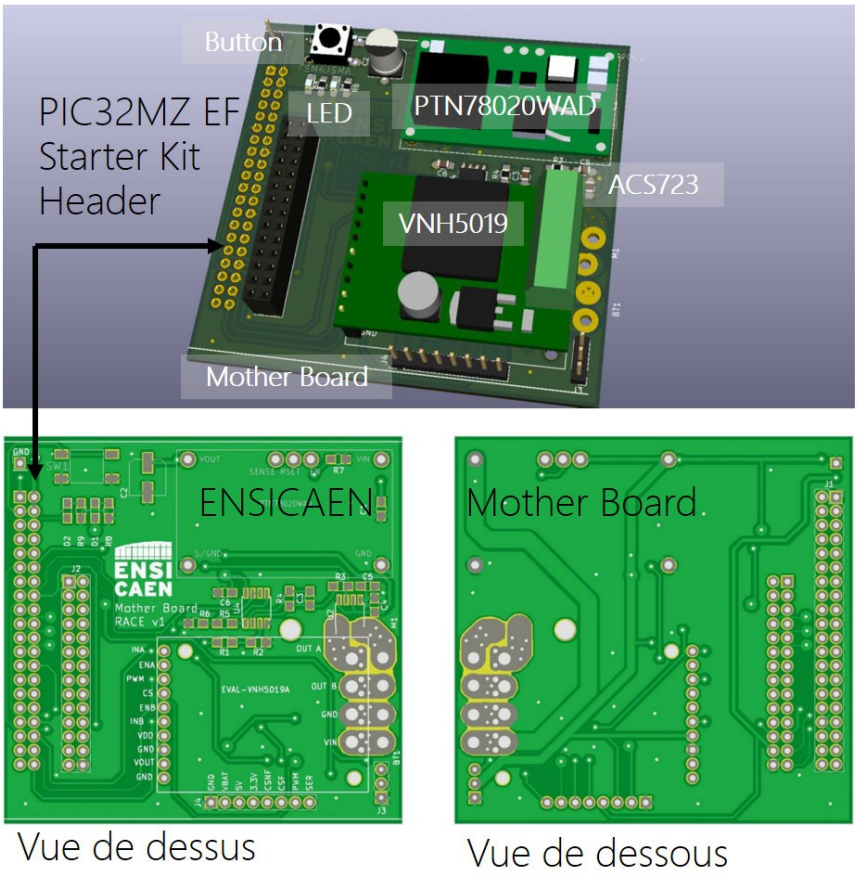
- Résistance primaire : 0,65mΩ
- Courant d'entrée : +/-40A

La mesure de courant sera réalisée par un capteur à effet Hall ACS723 proposé par la société Allegro Microsystems, société notamment spécialisée dans la mesure et le développement de modules de puissance. L'image du courant après captation étant fortement bruitée, la chaîne d'instrumentation intègre un filtre passe bas de fréquence de coupure à 100Hz et amenant une atténuation garantissant l'utilisation de la pleine échelle de l'ADC du MCU PIC32MZ (0-3,3V, correspondant à la plage -/+40A).



Filtre d' instrumentation

Carte mère ENSICAEN



Vue de dessus

Vue de dessous

La carte mère du projet RACE interface différentes cartes filles du marché (VNH5019, PTN78020WAD, WINC1500 PICTAIL et PIC32MZ EF Starter Kit). Pour un premier design, et compte-tenu du temps dévolu au développement de l'électronique du projet (quelques semaines), cette solution a été préférée. En effet, chaque carte fille sélectionnée implémente de grandes difficultés de routage :

- VNH5059 : Circulation et commutation de courants allant jusqu'à 30A max à 20KHz. Problématiques de dissipation thermique et CEM fortes pour tout nouveau design
- PTN78020WAD : Alimentation à découpage de type Buck-Boost pouvant proposer des commutation de courant de 6A max à haute fréquence. Problématiques de dissipation thermique et CEM fortes pour tout nouveau design
- PICM32 MF EF Starter Kit : Boîtier 144 broches demandant potentiellement le développement d'un PCB multicouches en fonction des périphériques matériels externes utilisés
- ATWINC1500 : Le module le moins complexe à intégrer. 2 semaines complémentaires (conception, réalisation, test et validation) auraient suffit pour valider l'intégration du module à la carte mère. Suffisamment de place a été laissé (zone autour du connecteur pour le module WINC1500 PICTAIL) sur le PCB pour intégrer le module WIFI seul (avec composant de cryptographie), sans pour autant utiliser la carte fille complète

Application logicielle embarquée

Le logiciel embarqué dans le système CORTEX-RACE devra assuré la supervision complète de la plateforme SMP-RACE, ainsi que garantir l'exécution temps réel de l'algorithme de régulation en courant (image du couple). Dans un premier temps, listons les interfaces du système ainsi que les niveaux de priorité (de PRIO0 à PRIO2, PRIO0=priorité facultative, PRIO1=priorité faible et PRIO2=priorité impérative) :

- PRIO2 : Commande par module PWM (Pulse Width Modulation) du servomoteur
- PRIO2 : Commande par module PWM et GPIO (General Purpose Input Ouput) du moteur de propulsion à travers le driver de puissance VNH5019
- PRIO2 : Mesure par module ADC (Analog to Digital Converter) du courant absorbé par le moteur de propulsion
- PRIO2 : Interface de communication WIFI par module SPI (Serial Peripheral Interface) et GPIO
- PRIO1 : Pilotage d'une LED utilisateur. Indication de l'état du système

Le projet peut être découpé en trois tâches pouvant être développées en parallèles avant intégration par 3 ingénieurs distincts. L'intégration sera pilotée voire réalisée par le chef de projet architecte. Présentons les 3 grands pans de développement du projet.

Identification, modélisation et régulation du procédé

Un ingénieur R&D sera dédié à l'identification, la modélisation, la synthèse d'un régulateur et son intégration dans l'applicatif embarqué. L'identification pourra se faire en temps différé par récupération des données du PIC32MZ depuis un ordinateur afin d'assurer une post analyse sous environnement Matlab/Simulink. Après modélisation et synthèse du régulateur sous Matlab (format flottant simple précision), l'intégration C devra être assurée sur le système CORTEX-RACE. Une protection en courant sera réalisée par l'asservissement afin de protéger l'électronique de puissance de contrôle du moteur. Cette protection se fera par saturation. Le système recevra les ordres (ou consignes de propulsion) toutes les 20ms (fréquence de 50Hz) depuis l'ordinateur. Néanmoins, il devra garantir un déterminisme temps réel dur (théorème de Shannon ou période d'échantillonnage à définir) quant à l'exécution de l'algorithme de commande (acquisition ADC, calcul de la sortie du régulateur puis commande PWM). Un rapport d'étude devra être produit afin de présenter les choix et justifications du régulateur implémenté (structure, synthèse, contraintes, limitations, etc). Voici un exemple chronologique conseillé des phases de développement et conception :

- Projet de test unitaire Baremetal permettant de piloter le module de puissance VNH5019 et donc le moteur en BO (Boucle Ouvert)
- Projet de test unitaire Baremetal permettant de lire le courant absorbé par le moteur en BO

- Projet de test unitaire Baremetal assurant la génération d'un profil d'identification en BO autour d'un point de fonctionnement (échelon, SBPA, etc) et l'enregistrement en temps réel du profil de courant absorbé. Après sauvegarde en mémoire interne des vecteurs de test, l'application de test renverra les données vers ordinateur par UART
- Projet d'identification, de modélisation, de synthèse d'un régulateur et de validation sous environnement Matlab/Simulink. Prototypage au format flottant simple précision puis validation de la structure du régulateur
- Projet de test unitaire Baremetal assurant le test et la validation du régulateur en BF (Boucle Fermée) sur PIC32MZ
- Intégration du régulateur au projet complet avec RTOS
- Documentation du projet de test embarqué implémentant l'algorithme de commande (architecture du projet de test, profils et méthodologie d'identification, jeu de commandes de la console série, méthodologie de synthèse du régulateur et de l'algorithme de commande, etc)

Interface série de test et de prototypage

L'applicatif de prototypage et de test offrira, en plus des commandes standards précédemment présentées dans le document, les commandes suivantes :

- command (prioritaire) : Commande du module PWM en boucle ouverte (BO). Valeurs numériques limitées par logiciel comprises entre 0 et 5000
- order (prioritaire) : Consigne de courant du système asservi et régulé en boucle fermée (BF). Valeurs numériques limitées par la plage d'acquisition de l'ADC 12bits (0-4091)
- step-open (facultatif) : Profil d'identification de type échelon en boucle ouverte. Valeurs numériques limitées par logiciel comprises entre 0 et 5000
- step-closed (facultatif) : Profil de poursuite de type échelon pour validation en boucle fermée. Valeurs numériques limitées par la plage d'acquisition de l'ADC 12bits (0-4091)
- D'autres commandes peuvent être implémentées en fonction de vos besoins

```
RACE # help
command  -  open loop direct PWM value between 0 and 5000
order  -  closed loop order between 0 and 4091
step-open  -  direct PWM step pattern between 0 and 5000
step-closed  -  closed loop step pattern order between 0 and 4091
RACE # command
Enter a value between 0 and 5000 : 7777
Out of range value
RACE # command
Enter value between 0 and 5000 : 3333
RACE # step-open
RACE #
```

Bibliothèque Réseau

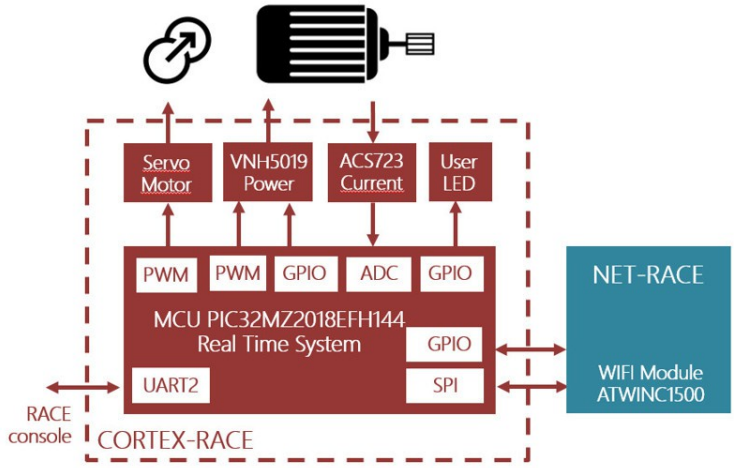
Un ingénieur développeur sera dédié à cette tâche. Cette partie sera détaillé dans la suite du document à travers le chapitre NET-ROV.

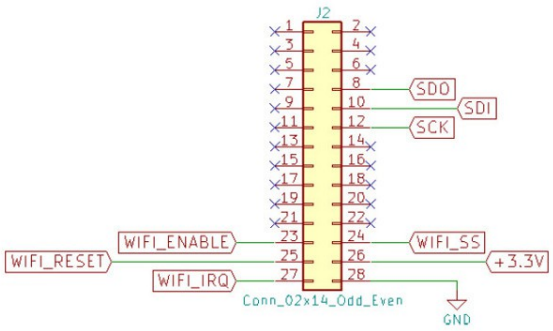
Architecture du système logiciel embarqué

Le chef de projet, également architecte système du projet, sera en charge de définir et développer l'architecture logicielle du système embarqué (environnement multi-tâches, outils systèmes, API applicative, nommage des interfaces, gestion et protection des ressources partagées, etc). Il aura également la charge de déployer et gérer le dépôt de gestion de version Git distant. Une fois les tests et les validations unitaires de chaque sous projet réalisées, il aura également en charge de planifier voire réaliser l'intégration globale du projet. Voici un exemple chronologique conseillé des phases de développement et conception :

- Projet avec RTOS permettant de déployer l'architecture logicielle du système embarqué complet (sans implémentation des séquences et définitions des tâches applicatives, ni les configurations des périphériques)
- Projet de test unitaire Baremetal permettant de piloter le servomoteur (cette tâche peut être réalisée par un autre ingénieur développeur du projet)
- Projet de test unitaire Baremetal permettant de piloter la LED utilisateur (cette tâche peut être réalisée par un autre ingénieur développeur du projet)
- Intégration de la commande du servomoteur au projet complet avec RTOS
- Intégration de la commande de la LED au projet complet avec RTOS
- Intégration de la bibliothèque WIFI au projet complet avec RTOS
- Intégration du régulateur au projet complet avec RTOS
- Développement, test validation de l'applicatif complet. Si le temps le permet, mettre le processeur en veille si aucune tâche applicative ne fonctionne
- Documentation du projet logiciel embarqué (modélisation de l'architecture logicielle embarquée, contraintes et limitations, résultats et mesures, etc). Le formalisme de spécification de l'architecture logicielle est laissé libre (exemple à titre illustratif donné ci-dessous). Bien penser à le documenter

3.2. Réseau WIFI - NET-RACE

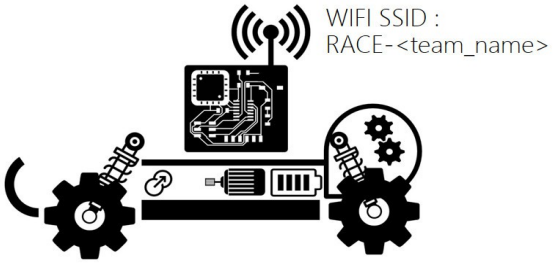




RACE Mother Board
ATWINC1500 Header

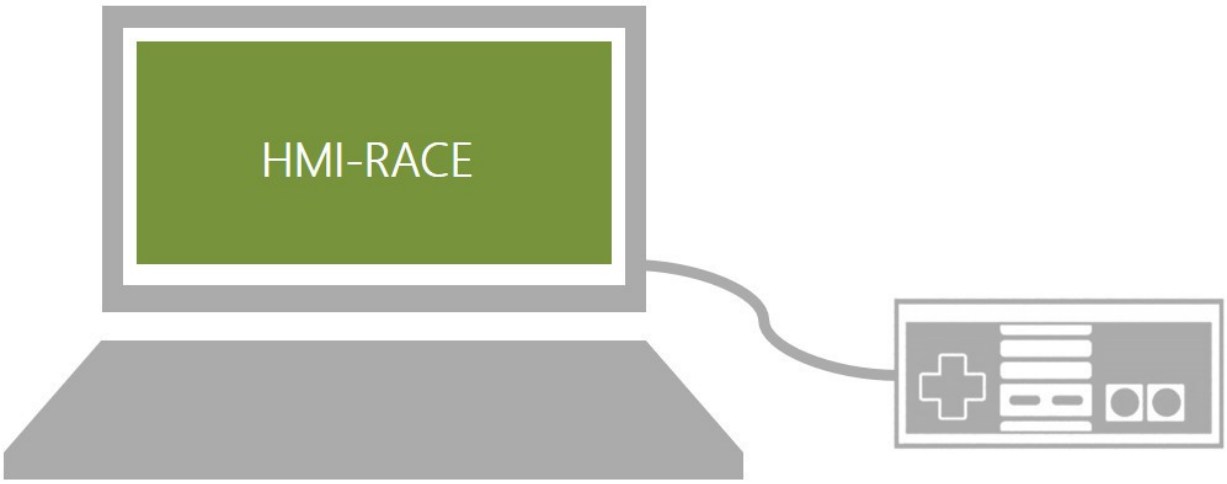
L'ingénieur responsable de cette partie aura en charge la validation de l'interface réseau par WIFI de l'application. Le module ATWINC1500 est en 2018 la dernière solution WIFI proposée par Atmel (propriété de Microchip) et est celle conseillée pour tout nouveau design sur solution processeur de Microchip. Notre choix s'est donc naturellement tourné vers cette solution. Nous souhaitons à terme libérer un maximum de ressource mémoire processeur pour l'emport d'éventuelles nouvelles bibliothèques. La principale difficulté de cette partie consistera à réduire au maximum les services de la librairie réseau. Les échanges avec l'ordinateur host se feront en TCP. Cela permettra la validation par acquittement des échanges d'informations tout en assurant un débit de communication suffisant. La donnée montante sera la mesure de courant. Les données descendantes seront la consigne de propulsion (le courant est l'image du couple sur une machine électrique) et la consigne de direction (consigne pour le servomoteur gauche/droite). Le SSID (Service Set IDentifier) devra impérativement suivre le nommage suivant : RACE-<team_name>. Voici un exemple chronologique conseillé des phases de développement et conception :

- Projet de test unitaire Baremetal permettant de déployer une communication WIFI avec l'ordinateur host (aucune réduction de la taille demandée). Validation des données montantes et descendantes en TCP
- Projet de test unitaire Baremetal assurant une diminution optimale de l'empreinte mémoire processeur de la pile réseau
- Intégration de la bibliothèque réseau au projet complet avec RTOS
- Documentation de la configuration Harmony utilisée (justification des dépendances et de la solution retenue), modélisation architecturale de la pile réseau de Microchip et des applicatifs de test

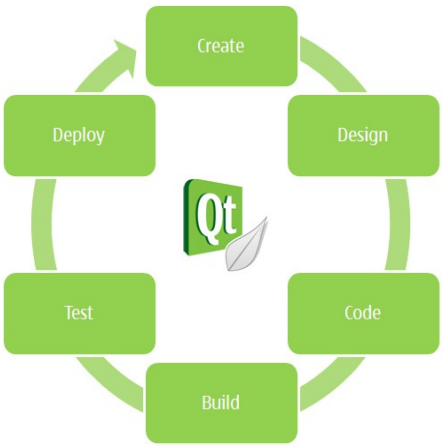


4. APPLICATION SUR ORDINATEUR

4. APPLICATION SUR ORDINATEUR

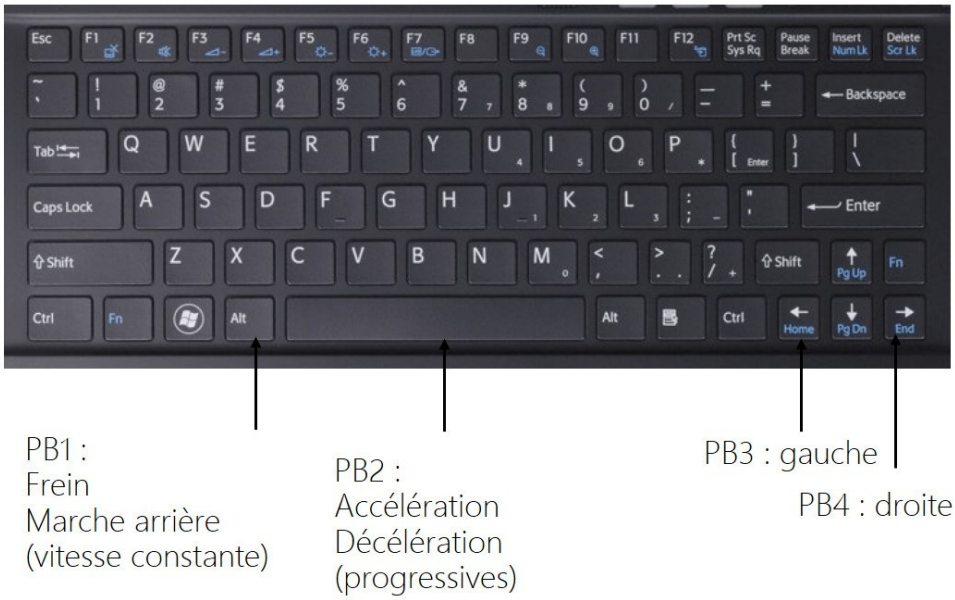


4.1. Interfaces de contrôle



L'application HMI-RACE (Human Machine Interface for RACE project) proposera potentiellement 2 interfaces de contrôle du véhicule télécommandé. Une interface depuis le clavier de l'ordinateur host (développement impératif) et l'autre depuis une manette USB de console de jeu vidéo (développement non prioritaire). Les deux solutions sont présentées dans la suite du document. L'interface graphique sera développée en utilisant l'API orientée objet Qt. Nous figerons l'API à la version Qt v5.11. De même, l'IDE Qt Creator v4.6.1 sera à utiliser pour l'ensemble des développements. La communication réseau de l'applicatif vers le module WIFI ATWINC1500 embarqué se fera à travers le module Qt Network. L'applicatif devra gérer la période de lecture puis d'envoi des ordres vers le véhicule. Un déterminisme de 20ms devra être garanti par l'application sur ordinateur. Cette période garantira notamment la descente de 50 consignes par seconde, ce qui, une fois asservi, assurera une bonne réactivité du véhicule.

Interface de contrôle depuis le clavier d'ordinateur



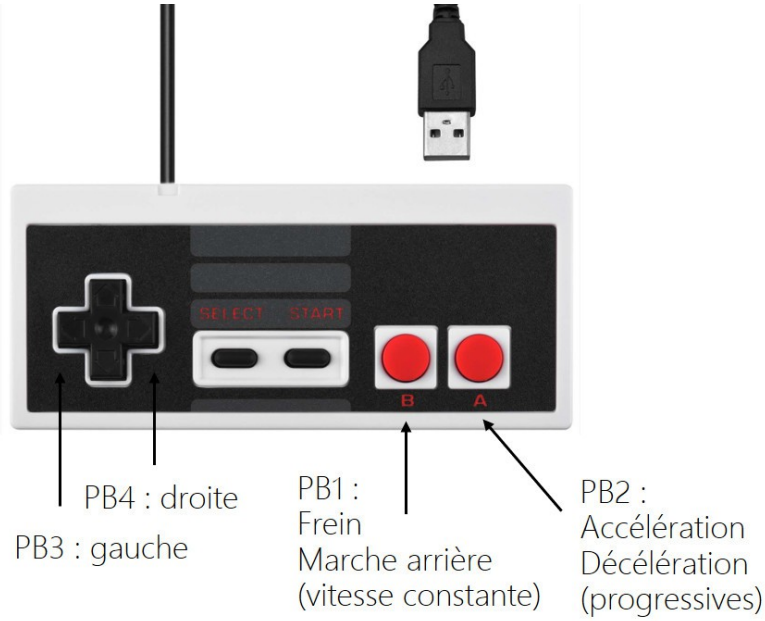
- PB1 (touche Alt) : freinage et marche arrière du véhicule. Au repos (aucune action sur PB1 et PB2), la consigne de propulsion envoyée au véhicule est nulle. Une action sur PB1 (sans action sur PB2) force la marche arrière du véhicule. La marche arrière se fera à vitesse constante, suffisamment lente pour garantir un bon contrôle de la voiture, mais suffisamment rapide pour assurer un dégagement rapide. Si PB2 est actionné (accélération), une action simultanée sur PB1 force une consigne de propulsion nulle et donc un arrêt brusque du véhicule. Le relâchement de PB2, tout en actionnant sur PB1 force donc une marche arrière.

- PB2 (barre Espace) : accélération et décélération progressives du véhicule. Au repos (aucune action sur PB1 et PB2), la consigne de propulsion envoyée au véhicule est nulle. Une action sur PB2 lance la procédure d'accélération progressive du véhicule. Un indicateur visuel sur l'interface graphique permettra au pilote d'avoir un retour de la consigne envoyée. L'accélération de vitesse nulle à vitesse maximale se fera en quelques secondes. Un relâchement de PB2 durant une phase d'accélération forcera une décélération progressive jusqu'à arrêt. Une décélération complète se fera également en quelques secondes.

- PB3 (flèche gauche) : ordre de tourner progressivement les roues avant à gauche. En une seconde, les roues devront être pleinement orientées en butée. Une action simultanée sur PB3 et PB4 force les roues à s'aligner dans l'axe du châssis (aller tout droit)

- PB4 (flèche droite) : ordre de tourner progressivement les roues avant à droite. En une seconde, les roues devront être pleinement orientées en butée. Une action simultanée sur PB3 et PB4 force les roues à s'aligner dans l'axe du châssis (aller tout droit)

Interface de contrôle depuis une manette de jeu vidéo



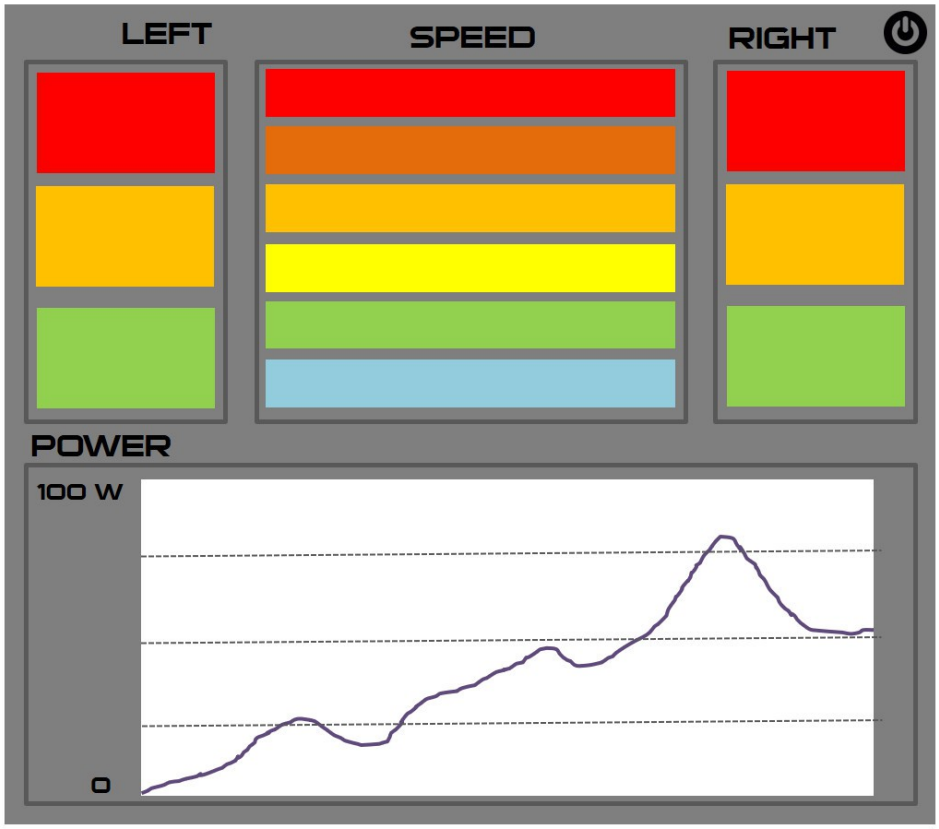
- PB1 (touche B) : freinage et marche arrière du véhicule. Au repos (aucune action sur PB1 et PB2), la consigne de propulsion envoyée au véhicule est nulle. Une action sur PB1 (sans action sur PB2) force la marche arrière du véhicule. La marche arrière se fera à vitesse constante, suffisamment lente pour garantir un bon contrôle de la voiture, mais suffisamment rapide pour assurer un dégagement rapide. Si PB2 est actionné (accélération), une action simultanée sur PB1 force une consigne de propulsion nulle et donc un arrêt brusque du véhicule. Le relâchement de PB2, tout en actionnant sur PB1 force donc une marche arrière.

- PB2 (touche A) : accélération et décélération progressives du véhicule. Au repos (aucune action sur PB1 et PB2), la consigne de propulsion envoyée au véhicule est nulle. Une action sur PB2 lance la procédure d'accélération progressive du véhicule. Un indicateur visuel sur l'interface graphique permettra au pilote d'avoir un retour de la consigne envoyée. L'accélération de vitesse nulle à vitesse maximale se fera en quelques secondes. Un relâchement de PB2 durant une phase d'accélération forcera une décélération progressive jusqu'à arrêt. Une décélération complète se fera également en quelques secondes.

- PB3 (flèche gauche) : ordre de tourner progressivement les roues avant à gauche. En une seconde, les roues devront être pleinement orientées en butée. Une action simultanée sur PB3 et PB4 force les roues à s'aligner dans l'axe du châssis (aller tout droit)

- PB4 (flèche droite) : ordre de tourner progressivement les roues avant à droite. En une seconde, les roues devront être pleinement orientées en butée. Une action simultanée sur PB3 et PB4 force les roues à s'aligner dans l'axe du châssis (aller tout droit)

4.2. Interfaces graphiques - HMI-RACE



Interface graphique proposée non-contractuelle

Interface graphique de jeu

Dans cette partie du projet, nous souhaitons laisser beaucoup de liberté à l'équipe de développement. De ce fait, nous souhaitons idéalement être surpris pas l'interface proposée. De même, tous les visuels proposés dans ce document sont donc non-contractuels et donnés à titre illustratif. Dans un premier temps, listons les fenêtres ou cadres proposées par l'HMI ainsi que les niveaux de priorité associés (de PRIO0 à PRIO2, PRIO0=priorité facultative, PRIO1=priorité faible et PRIO2=priorité impérative) :

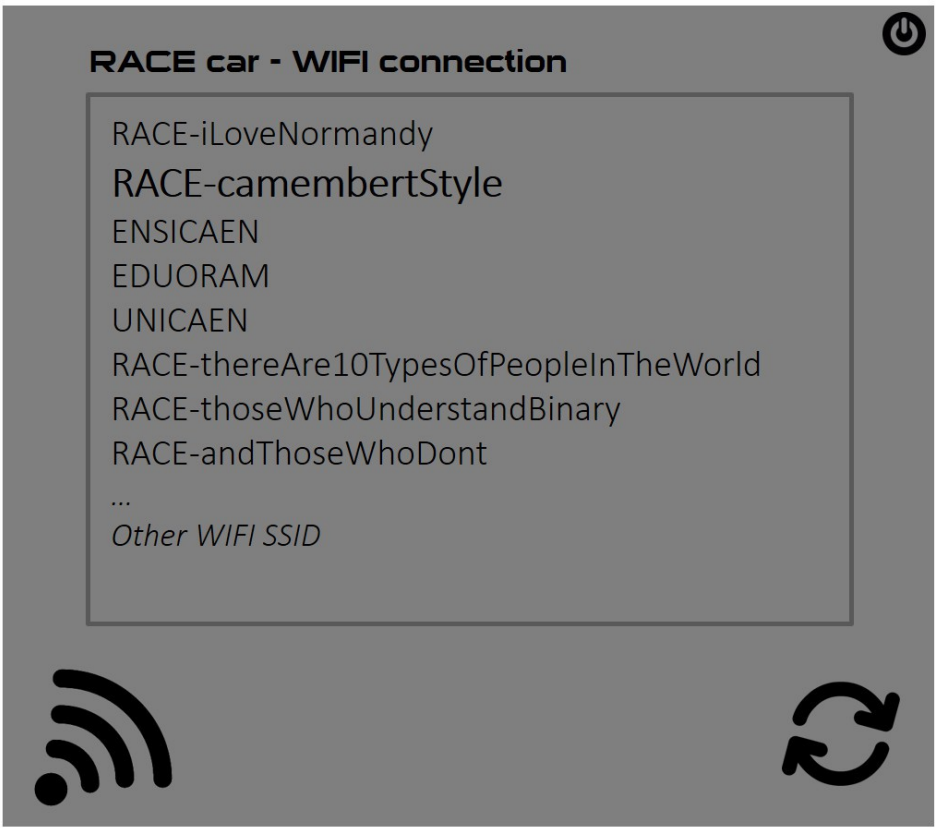
- PRIO2 – fenêtre SPEED (possibilité de la renommer) : Fenêtre présentant un indicateur visuel représentatif de la consigne de propulsion envoyée au véhicule (action sur PB2 et/ou PB1). Rappelons que la consigne de vitesse (image du courant et donc du couple dans la machine) aura une évolution progressive durant les phases d'augmentation (accélération) et de réduction (décélération). Se référer à la présentation des interfaces de contrôle

- PRIO1 – LEFT et RIGHT (possibilité de les renommer) : Fenêtre présentant un indicateur visuel représentatif de la consigne de direction envoyée au véhicule (action sur PB3 et/ou PB4). Rappelons que la consigne de direction (pilotage du servomoteur) aura une évolution progressive. Se référer à la présentation des interfaces de contrôle

- PRIO0 - POWER (possibilité de la renommer) : Affichage en temps réel de la consommation électrique estimée de la voiture. L'affichage sera borné en 0 et 100 Watts. L'axe des abscisses sera représentatif du temps écoulé durant 3 minutes et se fera par une fenêtre glissante. Dernière valeur affichée à gauche

- PRIO2 : L'icône de fermeture de l'application (en haut en droite) arrêtera le programme et libérera les ressources mémoire utilisées.

Interface graphique de connexion



Interface graphique proposée non-contractuelle

L'interface graphique de connexion sera celle ouverte par défaut au démarrage de l'application. Par utilisation des flèches haut et bas du clavier de l'ordinateur puis action sur la touche ENTER pour sélectionner, le pilote pourra choisir la voiture à laquelle il souhaite se connecter. Tous les SSID des points d'accès réseaux WIFI visibles de l'ordinateur host seront listés. Rappelons que le SSID proposé par chaque véhicule RACE devra respecter le nommage suivant : RACE-<team_name>. Une fois connecté, l'interface graphique de jeu sera celle affichée par l'application (cf. présentation précédente). En cas de perte de connexion avec le réseau WIFI du véhicule, l'application devra ré-afficher l'interface graphique de connexion actuellement présentée. Une fois le réseau sélectionné et la connexion demandée, un icône d'attente (en bas à droite dans l'exemple ci-dessus) devra s'afficher pour prévenir le pilote que l'application a bien tenu compte de son choix et que la connexion est en cours. L'icône de fermeture de l'application (en haut en droite) arrêtera le programme et libérera les ressources mémoire utilisées.

Voici un exemple chronologique conseillé des phases de développement et de conception :

- Application de niveau 1 avec interface minimale (seulement les tâches de priorité impérative) et contrôle depuis le clavier. La connexion WIFI sera également intégrée
- Application de niveau 2 est une évolution de l'application de niveau 1 avec toutes les interfaces graphiques et contrôle depuis le clavier. Elle correspond néanmoins à un nouveau projet et ne sera réalisée que si l'application de niveau 1 est validée
- Application de niveau 3 complète et respectant pleinement les spécifications techniques. Contrôle depuis une manette USB de console de jeu vidéo. Elle correspond néanmoins à un nouveau projet et ne sera réalisée que si l'application de niveau 2 est validée
- Documentation de l'application sur ordinateur (architecture et modélisation logicielle UML, contraintes, limitations, etc)

Contraintes et limitations

Contraintes temporelles

L'applicatif sur ordinateur devra être le logiciel garant du rythme de lecture puis d'envoi des ordres (ou consignes) vers l'applicatif embarqué temps réel. Un déterminisme optimal devra être garanti, même si non impératif. La période de traitement et d'actualisation de l'application est fixée à 20ms (50Hz). Cela inclus, de façon prioritaire, la communication WIFI (envoi des consignes), mais également la mise à jour toutes les 200ms de l'affichage graphique de l'HMI (interfaces de jeu et de connexion).

Graphisme, ergonomie, police et colorisation

Aucun formalisme ni contraintes imposées. Tant que la solution proposée respecte les limitations et interfaces précédemment présentées, le choix global de l'ergonomie, des graphismes, des polices et des couleurs choisis sont pleinement libres. Idéalement, nous souhaitons être agréablement surpris par l'interface pilote proposée. Bien penser à mettre en avant votre nom d'équipe dans l'HMI et ainsi le nom de votre véhicule.

5. LIVRABLES

5. LIVRABLES

Cette dernière partie des spécifications synthétise les livrables à produire. Cette synthèse propose une présentation chronologique conseillée des productions par sous partie du projet. Afin de garantir la faisabilité du projet RACE, des deadlines seront à proposer par le chef de projet et à respecter dans la mesure du possible par les ingénieurs développeurs de l'équipe. Elles feront l'objet de discussions et de validation entre direction technique et équipe de développement. A chaque livrable sera associé un niveau de priorité :

- PRIO2 : Priorité impérative
- PRIO1 : Priorité basse
- PRIO0 : Priorité facultative

Système Embarqué - Automatique

- PRIO2 : Projet de test unitaire Baremetal permettant de piloter le module de puissance VNH5019 et donc le moteur en BO (Boucle Ouvert)
- PRIO2 :Projet de test unitaire Baremetal permettant de lire le courant absorbé par le moteur
- PRIO2 : Projet de test unitaire Baremetal assurant la génération d'un profil d'identification en BO (échelon, SBPA, etc) autour d'un point de fonctionnement et l'enregistrement en temps réel du profil de courant absorbé. Après sauvegarde en mémoire interne des vecteurs de test, renvoi des données vers ordinateur par UART.
- PRIO2 : Projet d'identification, de modélisation, de synthèse et de validation sous environnement Matlab/Simulink. Prototypage au format simple précision puis validation de la structure du régulateur.
- PRIO2 : Projet de test unitaire Baremetal assurant le test et la validation du régulateur en BF (Boucle Fermée)
- PRIO2 : Intégration du régulateur au projet complet avec RTOS
- PRIO2 : Documentation du projet de test embarqué implémentant l'algorithme de commande (architecture du projet de test, profils et méthodologie d'identification, jeu de commandes de la console série, méthodologie de synthèse du régulateur et de l'algorithme de commande, etc)

Système Embarqué - Architecture

- PRIO2 : Projet avec RTOS permettant de déployer l'architecture logicielle du système embarqué complet (sans implémentation des séquences et définitions des tâches applicatives, ni les

configurations des périphériques)

- PRIO2 : Projet de test unitaire Baremetal permettant de piloter le servomoteur (cette tâche peut être réalisée par un autre ingénieur développeur du projet)
- PRIO2 : Projet de test unitaire Baremetal permettant de piloter la LED utilisateur (cette tâche peut être réalisée par un autre ingénieur développeur du projet)
- PRIO2 : Intégration de la commande du servomoteur au projet complet avec RTOS
- PRIO0 : Intégration de la commande de la LED au projet complet avec RTOS
- PRIO2 : Intégration de la bibliothèque WIFI au projet complet avec RTOS
- PRIO2 : Intégration du régulateur au projet complet avec RTOS
- PRIO2 : Développement, test validation de l'applicatif complet. Si le temps le permet, mettre le processeur en veille si aucune tâche applicative ne fonctionne
- PRIO2 : Documentation du projet logiciel embarqué (modélisation de l'architecture logicielle embarquée, contraintes et limitations, résultats et mesures, etc). Le formalisme de spécification de l'architecture logicielle est laissé libre (exemple à titre illustratif donné ci-dessous). Bien penser à le documenter.

Système Embarqué - Réseau

- PRIO2 : Projet de test unitaire Baremetal permettant de déployer une communication WIFI avec l'ordinateur host (aucune réduction de la taille demandée). Validation des données montantes et descendantes en TCP
- PRIO1 : Projet de test unitaire Baremetal assurant une diminution optimale de l'empreinte mémoire processeur de la pile réseau
- PRIO2 : Intégration de la bibliothèque réseau au projet complet avec RTOS
- PRIO2 : Documentation de la configuration Harmony utilisée (justification des dépendances et de la solution retenue), modélisation architecturale de la pile réseau de Microchip et des applicatifs de test

Application sur Ordinateur

- PRIO2 : Application de niveau 1 avec interface minimale (seulement les tâches de priorité impérative) et contrôle depuis le clavier. La connexion WIFI sera également intégrée

- PRIO1 : Application de niveau 2 est une évolution de l'application de niveau 1 avec toutes les interfaces graphiques et contrôle depuis le clavier. Elle correspond néanmoins à un nouveau projet et ne sera réalisée que si l'application de niveau 1 est validée

- PRIO0 : Application de niveau 3 complète et respectant pleinement les spécifications techniques. Contrôle depuis une manette USB de console de jeu vidéo. Elle correspond néanmoins à un nouveau projet et ne sera réalisée que si l'application de niveau 2 est validée

- PRIO2 : Documentation de l'application sur ordinateur (architecture et modélisation logicielle, contraintes, limitations, etc)