BUREAU D'ETUDE GYROPODE

SUJET DE TD & TP

DEPARTEMENT DE GENIE ÉLECTRIQUE ET INFORMATIQUE

Responsable pédagogique : Claude BARON
Support : Sébastien DI MERCURIO

INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES TOULOUSE

Table des matières

1. INTRODUCTION

Dans le cadre de ce projet, votre rôle sera de concevoir, coder et tester le programme de contrôle et de supervision d'un gyropode.

2. LEXIQUE

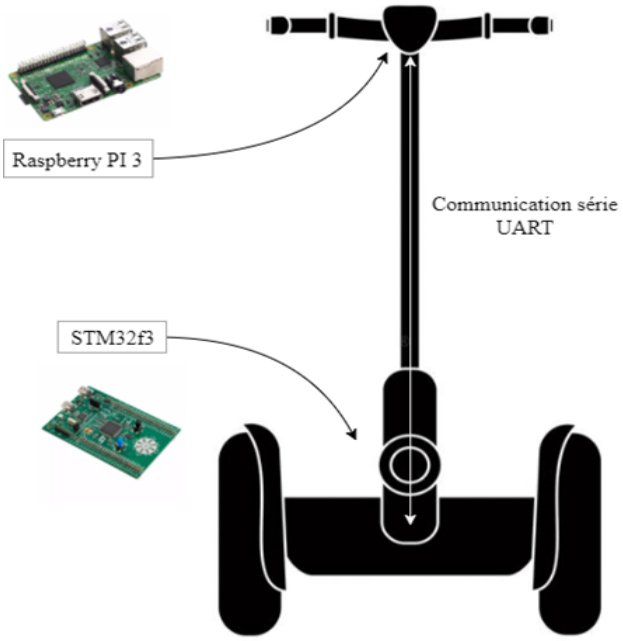
- Dépôt (git) : En informatique, un dépôt correspond à l'ensemble des documents informatiques d'un projet gérés en gestion de version. Cela se traduit par un archivage en réseau qui, une fois téléchargé sur l'ordinateur, correspond au contenu d'un répertoire donné.
- Git : Git est un gestionnaire de version couramment utilisé en informatique. Son rôle est de gérer les évolutions faites dans les fichiers d'un dépôt et assurer la synchronisation avec archivage réseau, dans notre cas Github.
- Moniteur : Dans le cadre de ce projet, le moniteur correspond au logiciel s'exécutant sur votre poste informatique (PC des salles de TP). Il s'agit d'une interface graphique simulant la tablette physique située au niveau du gyropode dont le rôle est d'afficher l'état du gyropode (vitesse, angle, présence de l'utilisateur…) ainsi que différents messages. Il vous sera fourni.
- Superviseur : Le superviseur correspond au logiciel de supervision du gyropode s'exécutant sur la carte Raspberry. C'est ce programme que vous aurez à concevoir et à coder pour qu'il réponde aux exigences.
- Netbeans : c'est un environnement de développement (ou IDE, pour Integrated Development Environment) qui vous servira à écrire, compiler et télécharger le programme du superviseur dans la carte Raspberry du gyropode.
- Carte Raspberry : Un ordinateur minimaliste et embarquable, équivalent en puissance d'un smartphone d'entrée de gamme.
- Carte STM32 : Une puce programmable (microcontrôleur) en charge du contrôle des capteurs et actionneurs du gyropode.
- Terminal : Une fenêtre (généralement noire), où l'on peut taper des commandes exécutées par le système Ubuntu sur le PC de la salle de TP.

3. LE PROTOTYPE INSA DE GYROPODE ET LA VERSION SIMULEE DU TP

L'objectif du gyropode est de déplacer une charge utile de 100 kg maximum sur un chemin goudronné, de pente maximale 5 %, à une vitesse maximale de 20 km/h en toute sécurité.
Trois batteries de 12 V en série de capacité 12 Ah servent de source d'énergie. Ces batteries alimentent les deux moteurs à courant continu de 500 W qui permettent de mettre en mouvement chacun une roue. En effet, pour que le gyropode puisse tourner, il faut que la vitesse de chaque roue soit gérée indépendamment l'une de l'autre. Par exemple, pour tourner à droite, la roue gauche doit tourner plus vite que la roue droite. La vitesse des moteurs étant trop importante, un réducteur sous forme de pignon chaîne est utilisé en sortie de moteur. Les roues sont de diamètre 25 cm. Tous ces composants se trouvent sur la partie basse du gyropode, au niveau du plateau sur lequel se tient l'utilisateur.

Le guidon permet de piloter le gyropode. Il est en liaison pivot par rapport au plateau, ce qui permet de le faire pivoter de quelques degrés autour de la direction d'avancée du système. Au niveau du guidon, une tablette sert à l'affichage de diverses informations : évolution de l'angle θ et la vitesse angulaire du guidon par rapport à la verticale, évolution de l'angle β, niveau de batterie, couple

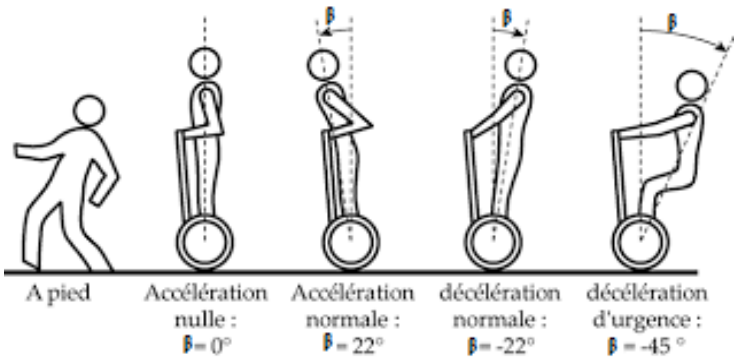
moteur, présence de l'utilisateur, arrêt d'urgence, état de la communication entre les deux cartes électroniques, et des informations supplémentaires éventuelles.

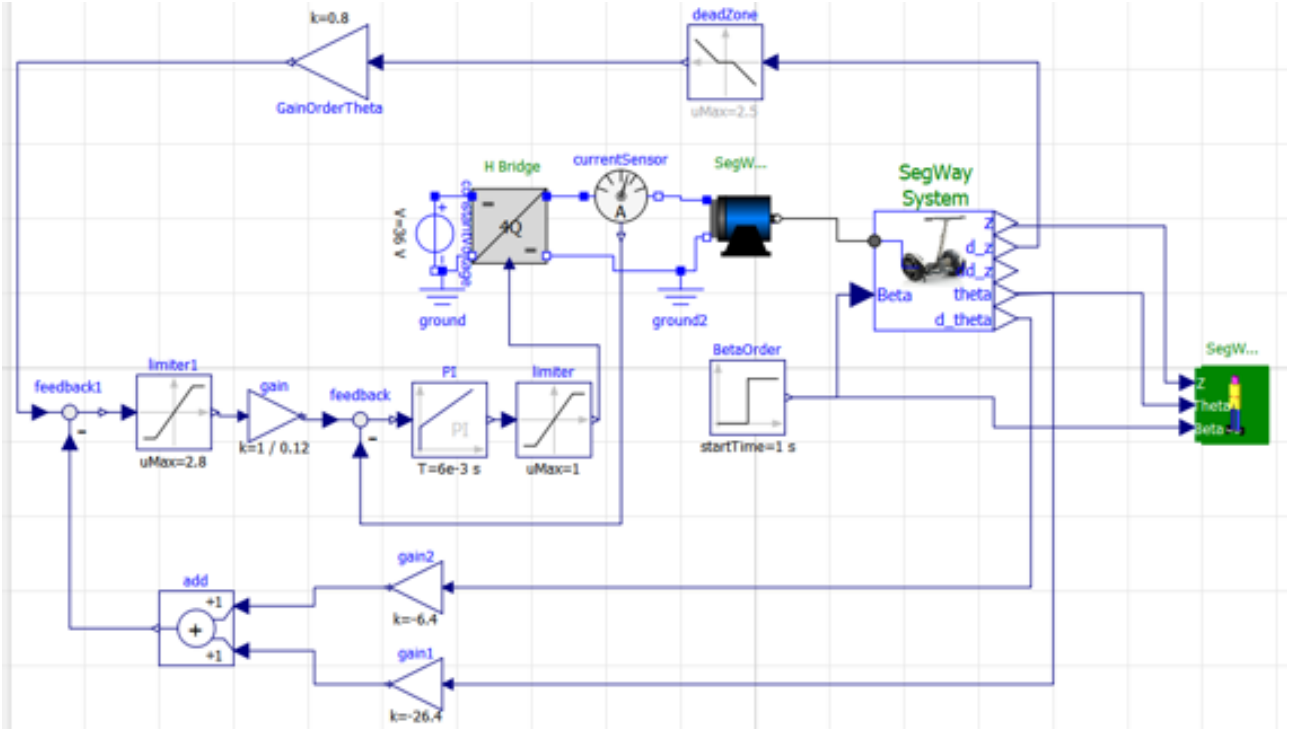


Deux boutons de part et d'autre du guidon doivent être maintenus enfoncés pour que le gyropode puisse fonctionner. Lorsque les boutons sont relâchés, le gyropode s'immobilise et ne peut être utilisé. Ce sont donc des boutons de sécurité qui permettent de s'assurer de la présence du pilote sur le plateau. Ils assurent la sécurité du conducteur lorsqu'il descend subitement du gyropode (si au moins un bouton est relâché, par exemple en cas de chute, le pilote est considéré comme absent).
Si le système est en mouvement, le gyropode décélère alors pour atteindre une vitesse linéaire nulle.

Le système peut être décomposé en deux sous-systèmes : un sous-système de commande de bas niveau, comprenant les capteurs et une carte STM32, et un sous-système de supervision de haut niveau, comprenant une carte Raspberry Pi 3. Ce dernier effectue le contrôle et la supervision du gyropode, et se charge aussi de la communication avec l'interface graphique de la tablette ; c'est au niveau de ce sous-système que s'exécutera le programme de supervision (appelé par la suite superviseur) que vous allez développer.

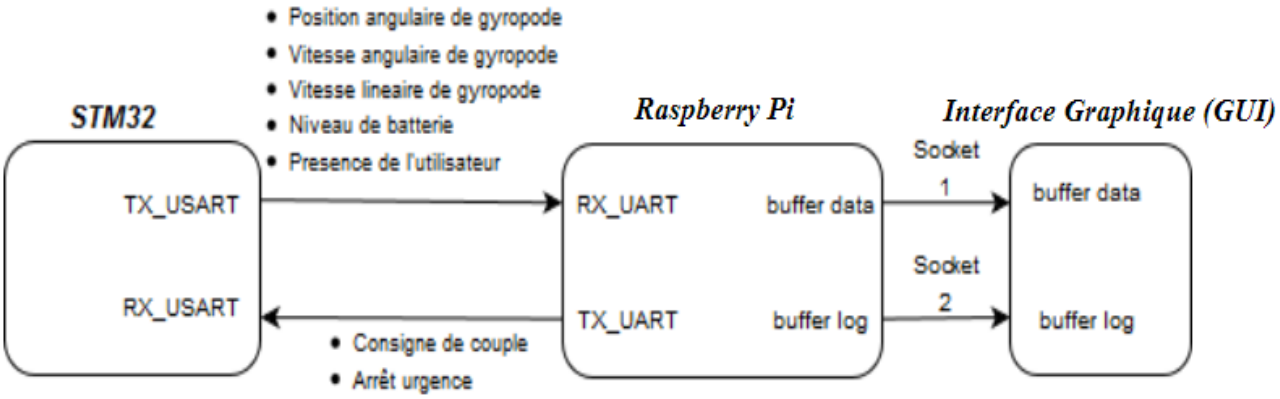
Le sous-système de bas niveau est en charge de la gestion des capteurs et actionneurs, ainsi que des boucles d'asservissement de bas niveau, proches du matériel, et du filtrage des capteurs. Il communique l'état du système (position angulaire, vitesses angulaire et linéaire, niveau de batterie, présence de l'utilisateur sur le plateau) qu'il perçoit via un ensemble de capteurs, incluant un accéléromètre et un gyroscope, au sous-système de supervision. Il calcule également l'angle d'inclinaison β de la plateforme (voir figure) et le transmet au sous-système de supervision.
Il reçoit par ailleurs de ce sous-système les ordres (consignes de couple, arrêt d'urgence…) via une liaison série asynchrone. Puis, à l'aide d'un correcteur PI, le sous-système de bas niveau contrôle les moteurs par MLI (Modulation à Largeur d'Impulsions) en leur appliquant une consigne en puissance (boucle interne sur la figure).





Le sous-système de supervision est en charge de la supervision en temps réel du gyropode et de la mise en œuvre de la loi de commande de haut niveau. Pour cela, il calcule la consigne de couple à appliquer aux moteurs en fonction de l'inclinaison de la plateforme (angle β) et de la vitesse de rotation du gyropode (vitesse angulaire, qui est simulée), envoyées par la carte STM32. C'est la boucle d'asservissement en angle (boucle externe sur la figure). Il transmet au sous-système de commande cette consigne de couple. Il assure également la surveillance du niveau de batterie et de l'angle β, et la gestion de l'arrêt d'urgence. Il évalue aussi quand il est nécessaire de déclencher un arrêt d'urgence et transmet alors une demande d'arrêt d'urgence au sous-système de commande.

NB : Dans la configuration du TP où le gyropode est simulé, le sous-système de supervision gère également la communication réseau (Wi-Fi) avec un poste de travail (PC), et, comme la maquette de simulation ne possède pas d'écran, il s'occupe aussi d'envoyer régulièrement les différents paramètres du système à un moniteur (fenêtre graphique du PC) qui simule la tablette se situant sur le guidon.

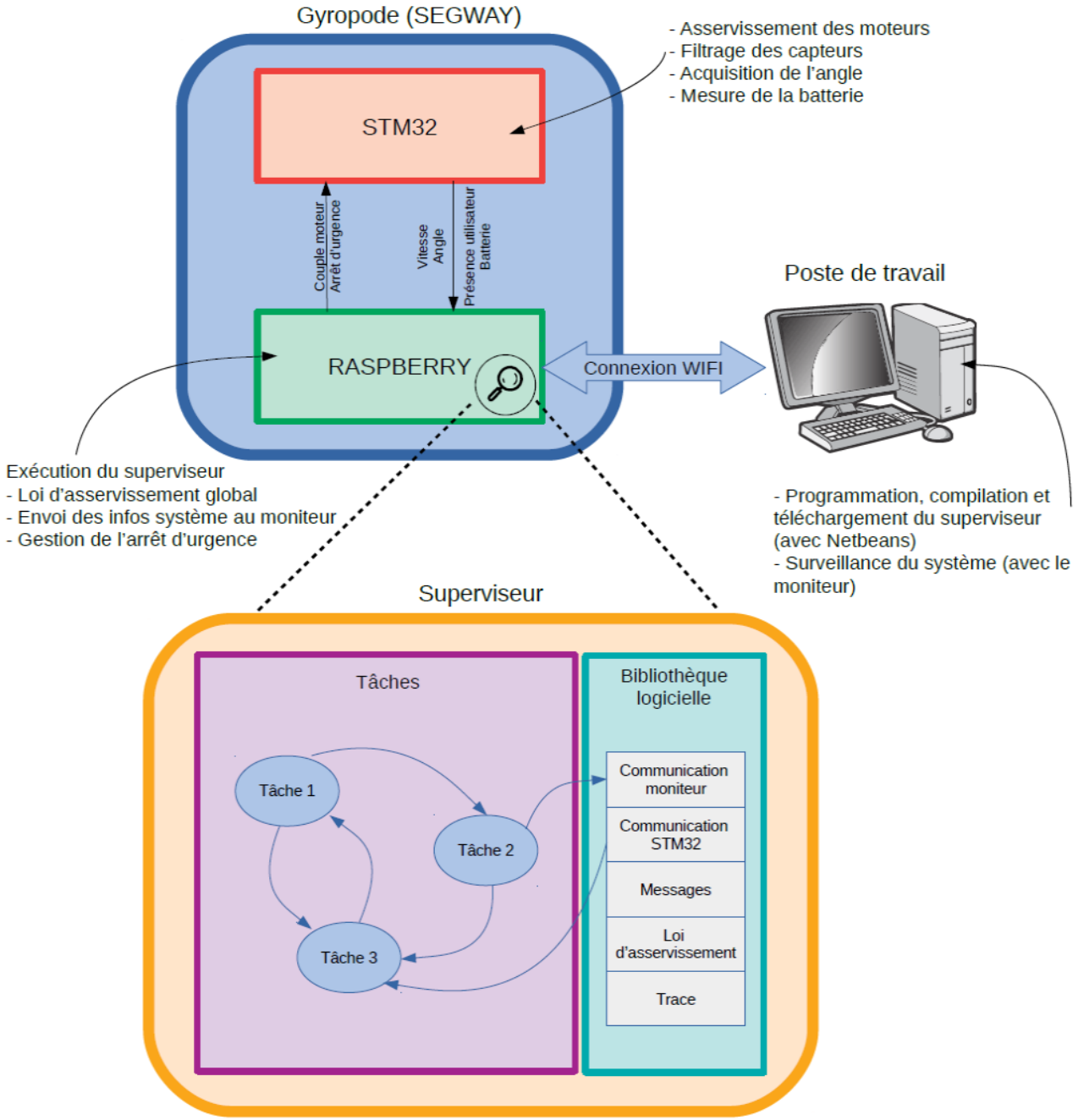


4. LE SOUS-SYSTEME DE CONTROLE ET DE SUPERVISION

La carte Raspberry Pi est un nano-ordinateur mono-carte à processeur ARM, sur lequel s'exécute une version du système d'exploitation Linux. Le noyau de base a été modifié pour lui ajouter un co-noyau temps réel, Xenomai. La carte exécute le programme de supervision en temps réel. L'interaction avec Linux ne se fait qu'en ligne de commande (pas d'IHM) et, virtuellement tout ce que l'on peut faire sur un PC standard peut être réalisé sur la Raspberry. Pour les opérations qui s'effectuent en temps contraint (comme la gestion de l'état du système et le calcul de la loi de commande), le programme de supervision utilise l'exécutif temps réel Xenomai.

Le programme de supervision (ou 'superviseur') que vous allez développer sera écrit en C++ (programmation objet).

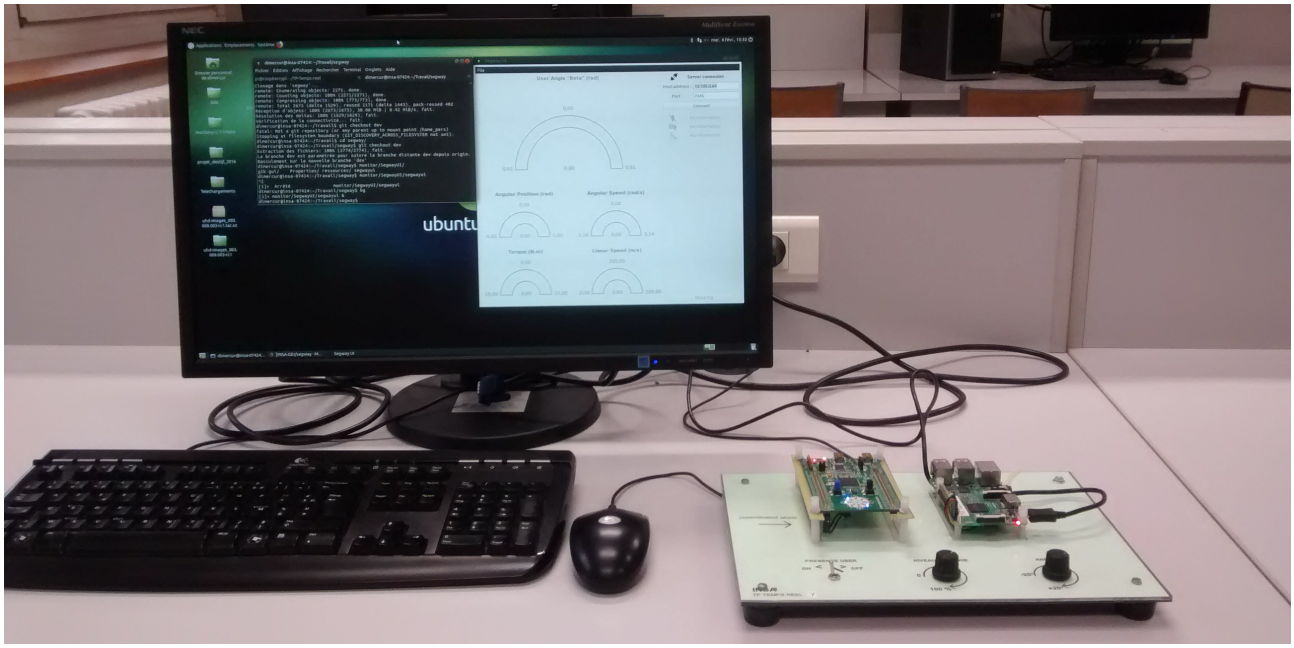
NB : Une ébauche de ce programme, compilable, vous sera fournie en TP ; en l'état, le programme ne fait rien d'autre que transmettre les messages en provenance du STM32 vers l'interface graphique, sans autre traitement. Il faudra le compléter / modifier pour répondre au cahier des charges et coder toutes les fonctionnalités attendues. Cette version est livrée avec un ensemble de classe déjà écrites, notamment en ce qui concerne la gestion de la liaison série et la mise en place d'un serveur TCP pour remonter l'état du système (variables) vers un moniteur (interface graphique) s'exécutant sur le poste de travail. L'objectif en vous fournissant cette version est d'illustrer comment utiliser les classes, mais aussi de vous montrer qu'elle sera, une fois que vous l'aurez complétée, l'architecture générale du programme, organisée en threads. Il est aussi de vous montrer comment créer et utiliser les différents outils de communication et de synchronisation offerts par Xenomai (sémaphore, mutex, file de message …).



5. LE SIMULATEUR DE GYROPODE

Plutôt que de travailler directement sur le gyropode, avec tous les risques que comporte la mise au point du programme, un simulateur a été développé, sur lequel vous allez travailler.

Le simulateur du gyropode est composé de la même carte STM32 que celle utilisée dans le prototype physique ; toutes les interruptions et fréquences des tâches sont similaires à celles du prototype afin de retranscrire au mieux le comportement du gyropode réel. Le code s'exécutant sur la carte STM32 est une version un peu modifiée afin de simuler la présence des roues et le comportement de la plateforme mais la partie dédiée à la communication des données avec le superviseur est identique au prototype. Par contre, l'écran présent sur le guidon ayant été supprimé ; à la place, une interface graphique s'exécutant sur un PC s'occupe de jouer le rôle de l'affichage. Le superviseur s'exécutant sur la Raspberry est en charge de transmettre les informations à l'interface graphique (voir chapitre **Erreur ! Source du renvoi introuvable.** - **Erreur ! Source du renvoi introuvable.**).



Deux parties spécifiques à la simulation ont été rajoutées. Une partie dédiée à la simulation physique du gyropode (angle d'inclinaison, simulé par un bouton rotatif), tandis que l'autre sert à simuler certains paramètres tels que le niveau de batterie (bouton rotatif) ou la présence de l'utilisateur (interrupteur).

5.1 Simulation physique du gyropode

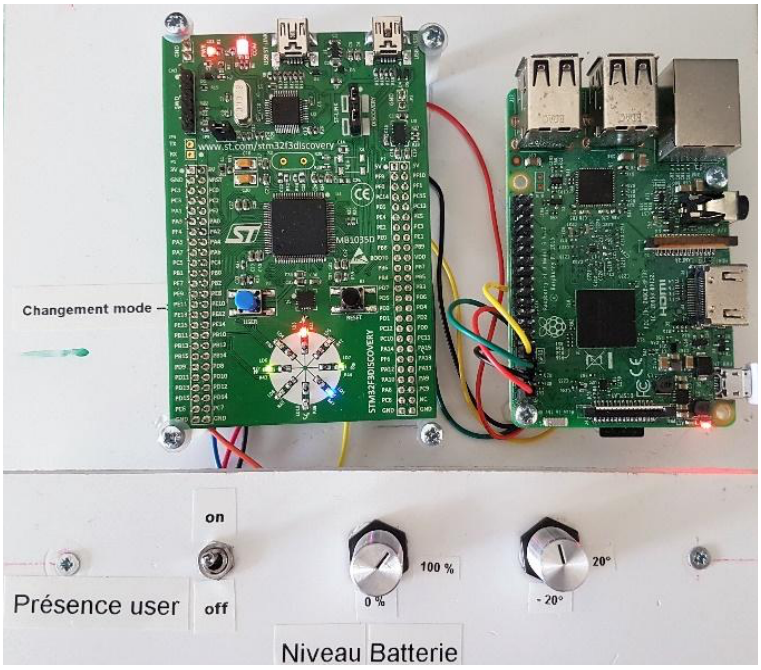
La simulation du comportement physique du gyropode est effectuée grâce au modèle du système qui nous a été fourni. Il s'agit d'un modèle physique linéarisé du gyropode qui n'est valable qu'entre une inclinaison de -20° à +20°. Mais cela est suffisant pour simuler le comportement du système.

Il aurait été intéressant de connaître la vitesse linéaire ; cependant, pour la calculer, il nous faudrait la tension fournie aux moteurs, qui n'est pas disponible dans le simulateur. C'est pourquoi dans le simulateur, la vitesse linéaire est toujours à 0 contrairement au prototype.

5.2 Simulation des variables du système

Dans le but de tester le code du logiciel de supervision, vous allez être amenés à faire varier l'état des variables du système comme le niveau de batterie ou la présence de l'utilisateur.

Le simulateur possède deux potentiomètres (niveau batterie et inclinaison) et un bouton (présence user).



La modification des variables se fait directement grâce à ces éléments. Les variations des variables sont instantanées et peuvent être continues, ce qui se rapproche de la réalité.

- Potentiomètres
  Toutes les lectures des valeurs de potentiomètre sont effectuées par l'ADC (Convertisseur Analogique vers Digital) du STM32. Un potentiomètre sert à simuler le niveau de batterie, un autre potentiomètre sert à simuler la valeur de l'angle Bêta entre -20 degrés et +20 degrés.

- Boutons
  Le bouton sert à simuler le bouton vérifiant la présence de l'utilisateur, présent sur le guidon de la maquette.

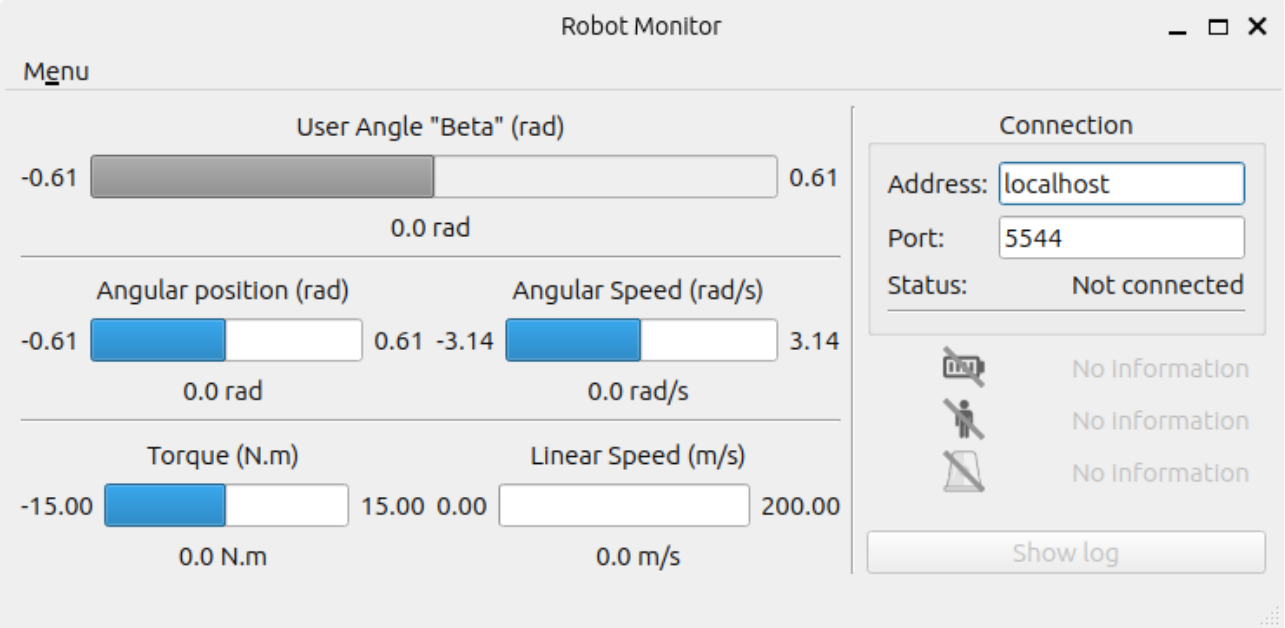
NB : Le bouton bleu sur la carte STM32 (notée « Changement mode » sur le simulateur) sert à changer le mode de modification de l'angle utilisateur : soit ce dernier est fixé par le potentiomètre, soit il est simulé par accéléromètre présent dans la carte STM32.

Pour faciliter la compréhension des opérations effectuées dans le STM32, des LEDs (visibles sur l'image, en bas de la carte de gauche, autour d'un cercle blanc) servent à indiquer différents états du simulateur.

Numéro de LED	Action
LED 3	Clignote à la réception de la trame de consigne
LED 4	Mode d'entrée de l'angle relatif entre l'utilisateur et le guidon => ON : Mode accéléromètre => OFF: Mode potentiomètre
LED 6	État de présence de l'utilisateur => ON : Présence => OFF: Absence
LED 8	État d'arrêt d'urgence => ON : arrêt déclenché
LED 5	Clignote à la réception de données par la Raspberry via USART
LED 7	Clignote quand il n'y a pas de connexion USART avec la Raspberry
LED 9	Clignote à l'envoi de données via USART à la Raspberry

6. LE MONITEUR

L'interface graphique du moniteur s'exécute sur le poste de travail de la salle de TP. Lors du démarrage initial, l'interface a l'apparence suivante :



Les différents paramètres sont initialement grisés, jusqu'à ce que le moniteur soit connecté au superviseur. Ils le resteront encore jusqu'à recevoir les premières données en provenance du superviseur (voir les annexes du TP).

Les messages donnant des informations sur les paramètres sont affichés directement sur l'IHM, les messages de traces (et les messages texte en général) ne s'affichent que dans la fenêtre de log. Pour information, si la loi d'asservissement à réaliser dans le superviseur n'est pas faite (ou mal), les valeurs de position angulaire et vitesse angulaire varient sans cesse.

Les différents widgets redeviennent gris lorsque l'on se déconnecte du superviseur.

7. DESCRIPTION DES FONCTIONNALITES ATTENDUES DU SUPERVISEUR

7.1 Asservissement

Le superviseur contrôle le couple moteur (accélération, freinage, arrêt). Il effectue régulièrement (à une fréquence d'environ 50 Hz), à partir des informations reçues du STM32, le calcul du couple qui sera envoyé au STM32 pour être appliqué aux moteurs.

7.2 Surveillance batterie

Le superviseur surveille le niveau de la batterie toutes les secondes (fréquence d'environ 1Hz), qu'il reçoit du STM32. Il déclenche l'arrêt du système lorsque le niveau de batterie commence à être trop bas (par exemple inférieur à 15%) en moins de 2 secondes.

7.3 Vérification de la présence du pilote

Le superviseur détecte en moins de 500 ms (tous les 2Hz) si l'utilisateur n'est plus présent sur le système (cette information est remontée par le STM32) et déclenche le système d'arrêt d'urgence (arrêt des moteurs).

7.4 Arrêt d'urgence

Il est déclenché par le superviseur, pour des raisons de sécurité, sur détection de l'absence du pilote, si le niveau de batterie est trop faible ou si l'angle ß reste sort de l'intervalle (-20, +20) degrés (l'intervalle (-20, +20) degrés ne peut être dépassé qu'en inclinant physiquement la platine).

(NB : En première intention, on déclenchera l'arrêt d'urgence sans distinguer les trois situations (chute du pilote, batterie faible, angle ß trop grand). En seconde intention, une fois que le logiciel fonctionnera, on distinguera ces situations et on les gérera différemment au niveau du superviseur, par exemple en faisant décélérer le gyropode, etc…).

Lorsque l'arrêt d'urgence est déclenché, il faut que le système s'arrête. Pour cela, il faut envoyer très régulièrement des trames d'arrêt au STM32. Côté STM32, un compteur vérifie si le STM32 reçoit toujours des trames d'information ; à chaque réception d'une trame, il incrémente un compteur. Dès que le STM32 reçoit une trame d'arrêt d'urgence, ce compteur est remis à zéro. Dès la réception du premier message de demande d'arrêt d'urgence, le STM32 envoie un couple nul au moteur, ce qui arrête le système. Quand le compteur atteint 300 (environ 3 secondes si la fréquence est 100 Hz dans l'envoi de données de la Raspberry Pi), le STM32 considère qu'il peut remettre en route le système, et traite à nouveau les messages reçus.

7.5 Affichage

Le gyropode informe l'utilisateur de sa vitesse angulaire, de l'angle ß (Bêta) de la barre du gyropode, de la consigne de couple appliquée, du niveau de batterie, de l'état de l'arrêt d'urgence, de la présence de l'utilisateur, et de la vitesse linéaire. Ces informations doivent être affichées sur l'interface graphique au moins toutes les 500 ms (fréquence d'au moins 2Hz).

8. TRAVAIL DE TD

Ce projet intègre plusieurs compétences scientifiques et techniques (conception d'application temps réel, programmation orientée objet, modélisation SysML, processus d'ingénierie système, approche systémique), des compétences en gestion de projet (agile) et en langues (anglais).

Durant le TD vous allez progressivement :

- **Préciser le système étudié** - dans son contexte de simulation - sous forme des diagrammes de contexte organique et fonctionnel, précisant les constituants externes, ses frontières et ses interactions avec les constituants externes, ainsi que ses interfaces (liens physiques, sous forme de liens dirigés et nommés), en considérant les différents modes d'utilisation du système (démarrage, fonctionnement nominal, arrêt, situation d'urgence) et différentes phases de vie du système (conception, réalisation, opération, maintenance).

  NB : nous nous intéressons aux interfaces concernant les échanges d'information

  Vous pouvez également dès cette étape établir des **scénarios opérationnels** pour chaque mode pour vous aider.

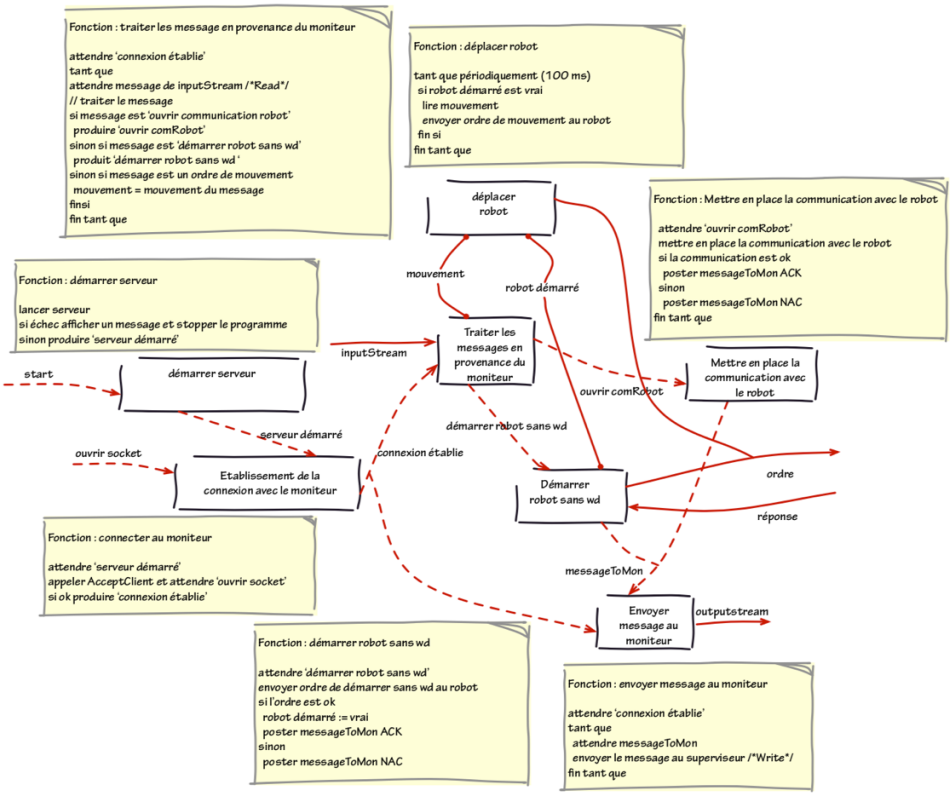
- **Exprimer les exigences** système :
  - identifier les besoins de services et performances,
  - exprimer ces besoins en exigences et raffiner si besoin les exigences selon les bonnes pratiques (voir Guide INCOSE sous moodle),
  - numéroter les exigences,
  - classer les exigences (voir taxonomie sous moodle),
  - vérifier et valider les exigences (voir document sous moodle) : assurez-vous notamment de la précision, cohérence, non-ambiguïté, mesurabilité et complétude des exigences.

- **Déduire des exigences les fonctionnalités du système** (étape de spécification : Qu'est-ce que le système doit faire ?) que le superviseur devra assurer (une fonction transforme un flux de données d'entrée en flux de sortie, en consommant du temps et des ressources).

  L'objectif final est d'obtenir un **diagramme d'architecture fonctionnelle** (statique et dynamique) montrant les échanges et les enchaînements entre ces fonctions (caractérisation des fonctions et de leurs interfaces, flux de données par les échanges d'entrées et sorties, flux de contrôles (activations, synchronisations), point de vue temporel).

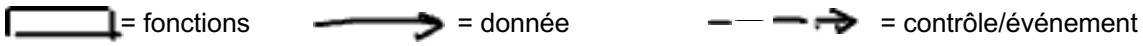
  Démarche à suivre : Sur la base de la mission que le système doit assurer (par exemple « Commander et superviser un gyropode »), raffiner progressivement en fonctions. Vous obtenez un ensemble de fonctions que vous pouvez caractériser par un nom, des entrées, des sorties, un comportement. L'assemblage de ces fonctions vous donne un schéma d'architecture fonctionnelle statique.

  Vous pouvez, pour le représenter, utiliser le formalisme de la figure ci-dessous, inspiré de SA-RT, une méthode d'analyse fonctionnelle adaptée au temps réel (voir mémo 'Analyse fonctionnelle' sous Moodle).



Exemple de représentation d'un diagramme d'architecture fonctionnelle en SA-RT like
(sur un exemple de supervision de robot mobile)

Légende : Les fonctions apparaissent dans des rectangles blancs, les flux de contrôle (événements) qui transitent entre les fonctions sont représentés sous forme de flèches en traits pointillés et les flux de données (entrées et sorties) en traits pleins (sous forme de post-it, une description succincte de la fonction)



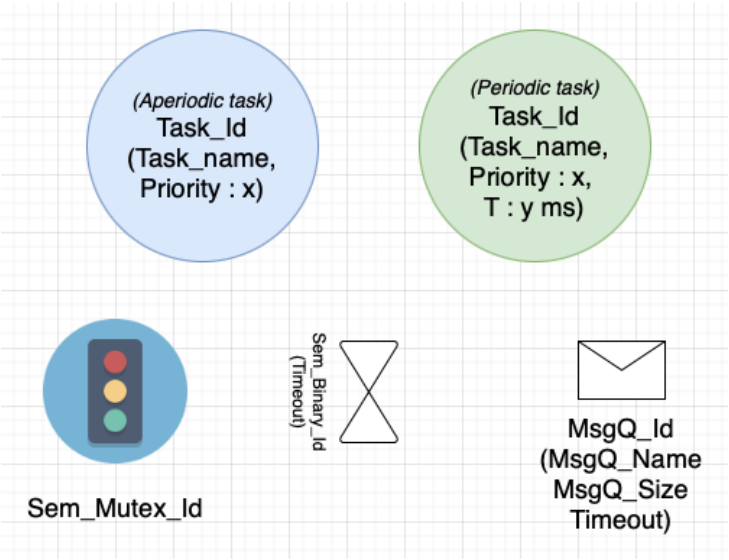
Ce diagramme sera complété par la description du comportement et de l'enchaînement des fonctions dans le temps à l'aide de **diagrammes faisant apparaitre les flux de données et la temporalité** (architecture fonctionnelle dynamique.

Assurez-vous que les résultats de cette étape sont cohérents avec le diagramme de contexte et les exigences. Assurez et démontrez pour cela la traçabilité entre fonctions et exigences (cf. cours SysML).

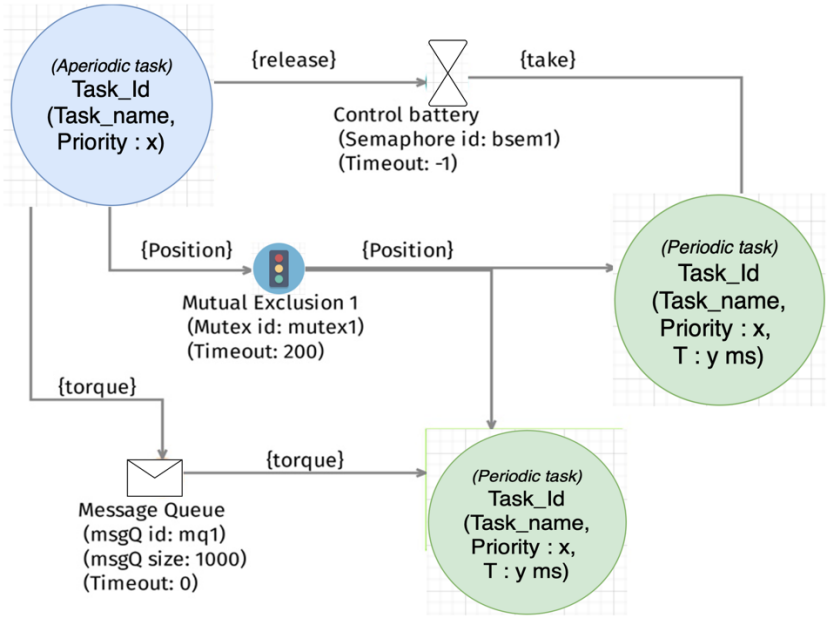
- **Identifier et caractériser un ensemble de tâches** (étape de conception : Comment le système va faire ?) permettant l'exécution de ces fonctions sur la plateforme logicielle choisie (Xenomai).

  L'objectif est d'abord de déterminer **les architectures logique et physique** de l'application. La vue statique fait apparaître les tâches (composants logiciels), les échanges de données et les synchronisations entre les tâches, et les moyens choisis (services de Xenomai) pour les implanter. La vue dynamique représente l'exécution dans le temps des tâches.

  NB : Vous pouvez, pour représenter l'architecture physique statique, utiliser le formalisme de la figure ci-dessous. (Voir sous Moodle le fichier « Architecture physique.drawio » (édité sous https://app.diagrams.net/).



Éléments de représentation sous Draw.io



Exemple de représentation d'un diagramme d'architecture physique

Sur le plan de la démarche, vous devez pour cela tout d'abord réfléchir à un découpage en tâches qui vous semble pertinent. Vous effectuerez un découpage 'optimal' de l'application en tâches, que vous justifierez ; ces tâches vont exécuter les différentes fonctions en intégrant et respectant des contraintes de parallélisme, de séquencement dans le temps, de temps, de partage de ressources, … (on peut donc être amené, selon le niveau de détail auquel vous avez fait l'analyse fonctionnelle, à faire exécuter une fonction par plusieurs tâches ou au contraire à regrouper plusieurs fonctions dans une même tâche).

Vous devez ensuite caractériser les tâches (nom, priorité1, période, entrées, sorties), leurs activations (périodique, sur événement…).

Puis choisir, caractériser et justifier des moyens de communication (variable globale protégée ou pas par un mutex, file de messages (taille, format des messages, timeouts…) et de synchronisation (sémaphore binaire, suspension/réveil,

Et enfin décrire l'activité globale de l'application logicielle (entrelacement des flux d'exécutions des différentes tâches dans le temps) ainsi que l'activité interne des tâches (par des diagrammes en SysML : le comportement de chaque tâche pourra par exemple être décrit à l'aide d'un diagramme d'activité.

- **Assurez et démontrez la traçabilité** de votre architecture avec les fonctions et exigences système (cf. Cours SysML).

  NB : Plusieurs options d'architecture sont possibles, identifiez-les et comparez-les pour en choisir une et **argumentez pourquoi vous avez fait ce choix**.

- **Préparer les tests** unitaires et tests d'intégration

  Vous devez dériver des exigences des plans de test fonctionnels (plans de test unitaire, d'intégration et de validation). Ces plans contiennent des séquences de test (entrée à appliquer, comportement attendu). Les tests seront appliqués une fois effectué le codage pour comparer le comportement obtenu au comportement attendu, et obtenir des résultats de test.

A FAIRE sur les séances de TD

Reproduisez cette démarche pour concevoir entièrement le superviseur. Aidez-vous du modèle de rapport pour répertorier quels sont les attendus de ce travail.

A FAIRE aussi, AVANT le premier TP :

Vous constaterez en analysant le code fourni que vous êtes contraints par la plateforme et que certains éléments logiciels sont déjà codés, prêt à être utilisés (interfaces avec le moniteur et le simulateur) ou donnés à titre d'exemples, et qu'il faudra supprimer ou modifier.

Vous devrez donc, avant la première séance de TP, et par une retro-ingénierie du code fourni, ajuster votre analyse précédente (notamment le diagramme d'architecture organique) pour intégrer ces contraintes.

***ATTENTION** : ne considérer que les contraintes aux interfaces du système (communication avec le STM32 et l'interface graphique). Le code contient d'autres tâches que celles proposées aux interfaces du système physique, ainsi que des mécanismes de communication et de synchronisation (sémaphores, mutex, files de messages). Ce code n'est là qu'à visée d'exemple de bonnes pratiques de codage.*

[1] Pour déterminer les priorités, voici quelques règles : pour les tâches périodiques, on attribue les priorités par ordre décroissant des périodes, c'est-à-dire qu'une tâche sera d'autant plus prioritaire que sa période sera petite. Pour les tâches apériodiques, le niveau de priorité va dépendre de la criticité des fonctions.

9. TRAVAIL DE TP

L'objectif des TP est de programmer et valider le logiciel dont vous avez défini l'architecture en TD (attention, le passage à l'implémentation de l'architecture physique construite en TD va nécessiter la prise en compte de contraintes imposées par la plateforme de simulation de TP, qui va potentiellement vous amener à revoir, à la marge, votre conception d'architecture).

Lors de la première séance, vous allez vous familiariser avec l'environnement de développement et le code fourni, **mettre en place votre organisation** (planification du codage et des tests, répartition du travail …), et identifier les modifications à apporter à votre architecture physique pour intégrer les contraintes de plateforme.

A la fin de cette séance, vous devez :
- savoir vous connecter à la cible et avoir compris en quoi consistent les opérations effectuées,
- avoir identifié le travail à faire sur les prochaines séances et planifié ce travail,
- avoir identifié les modifications à faire sur votre architecture physique pour la mettre à jour pour la séance suivante,
- avoir poursuivi la rédaction du rapport, initié durant les TD.

Lors des séances suivantes, vous identifierez des objectifs et vous organiserez pour les atteindre en équipe, en veillant à planifier le travail à faire sur les séances afin que vous puissiez livrer au client (votre encadrant de TP) ce qu'il attend à temps.

Ainsi :
- Décider de quelle tâche vous allez coder en premier (en fonction par exemple des dépendances), la coder, la tester, puis ainsi de suite avec chaque tâche ;
- Indiquer dans quel ordre vous avez choisi de coder vos tâches (stratégie de codage et d'intégration) et le justifier ;

A la fin de chaque séance, vous ferez une livraison au client de l'incrément développé et un point sur le travail restant à faire. Vous travaillerez également sur la rédaction du rapport de façon incrémentale.

NB : Pensez à **gérer en configuration les versions du logiciel** (avec commentaires indiquant le niveau d'avancement), que vous figez à la fin de chaque sprint (pensez à sauvegarder ces versions et à chaque sprint de repartir d'une copie sur laquelle vous allez développer).

Lors de la dernière séance se tiendra la livraison au client : vous lui ferez une présentation formelle du logiciel, démonstration des performances et du respect des exigences, argumentaire sur les choix de conception et réalisation ; **vous lui livrerez le code ainsi que le rapport**.

NB : Les groupes les plus avancés pourront profiter de cette séance pour développer des versions du logiciel offrant un comportement plus avancé.