

Dossier technique

Projet Rover

Dudzinski Théo / Dufour Jordan / Hadoux Benjamin / Gerzynski Gaetan

SOMMAIRE

Partie Commune	05
<i>Introduction et mise en contexte du projet</i>	06
<i>Spécifications :</i>	08
Diagrammes de cas d'utilisation	09
Description des cas d'utilisation	09
Diagrammes des exigences	12
Diagramme de déploiement	14
Diagramme de séquences point de vue système	15
<i>Étude de conception préliminaire</i>	16
Diagramme de définition de bloc	16
Diagramme de définition de bloc interne	17
<i>Répartition des tâches</i>	18
Partie 1 attribué à Dudzinski Thé Eau	18
Partie 2 attribué à Dufour Jordan	18
Partie 3 attribué à Hadoux Benjamin	18
Partie 4 attribué à Gerzynski Gaëtan	18
<i>Planification prévisionnelle</i>	19
Partie individuelle : Étudiant 1	20
<i>Planning prévisionnel</i>	21
<i>Solution retenue envisagée</i>	22
<i>Diagrammes de conception</i>	23
Diagramme de définition de bloc	23
Diagramme de définition de bloc interne	24
Flux d'information	25
Diagramme de classe	26
Diagramme de séquence	28
<i>Choix matériels</i>	30
Caractéristique principales	30
Choix matériels	31

Schéma électrique	32
Schéma électrique pour un capteur	32
Simulations	33
Simulation partie électronique	33
Prototypage	34
Prototypage	34
Test unitaire	37
Partie individuelle : Étudiant 2	39
Planning prévisionnel	40
Diagrammes de conception	41
Diagramme de définition de bloc	41
Diagramme de définition de bloc interne	42
Diagramme de classe	43
Diagramme de séquences	44
Langage de codage	47
Matériel mise à disposition	47
La maquette	48
IHM Principale	48
IHM Menu Automatique	49
IHM Menu Manuel	50
IHM Fiche d'intervention	51
IHM Configuration	53
IHM Pilotage	56
Codes	58
Connexion au Rover	58
Récupération des informations capteurs	59
Prototypage	60
Tests unitaires	60
Partie individuelle : Étudiant 3	62
Diagrammes de conception	63
Diagramme de définition de bloc	63
Diagramme de définition de bloc interne	64

Diagramme de séquences	64
<i>Étude du protocole</i>	65
<i>Prototypage</i>	66
Tests unitaires	66
Partie individuelle : Étudiant 4	67
<i>Diagrammes de conception</i>	68
Diagramme de définition de bloc	68
Diagramme de définition de bloc interne	69
Diagramme de classe	70
Diagramme de séquences	71
<i>Prototypage</i>	72
Tests unitaires	72
Annexes	75

Partie Commune

Cahier des Charges

Dudzinski Théo / Dufour Jordan / Hadoux Benjamin / Gerzynski Gaetan

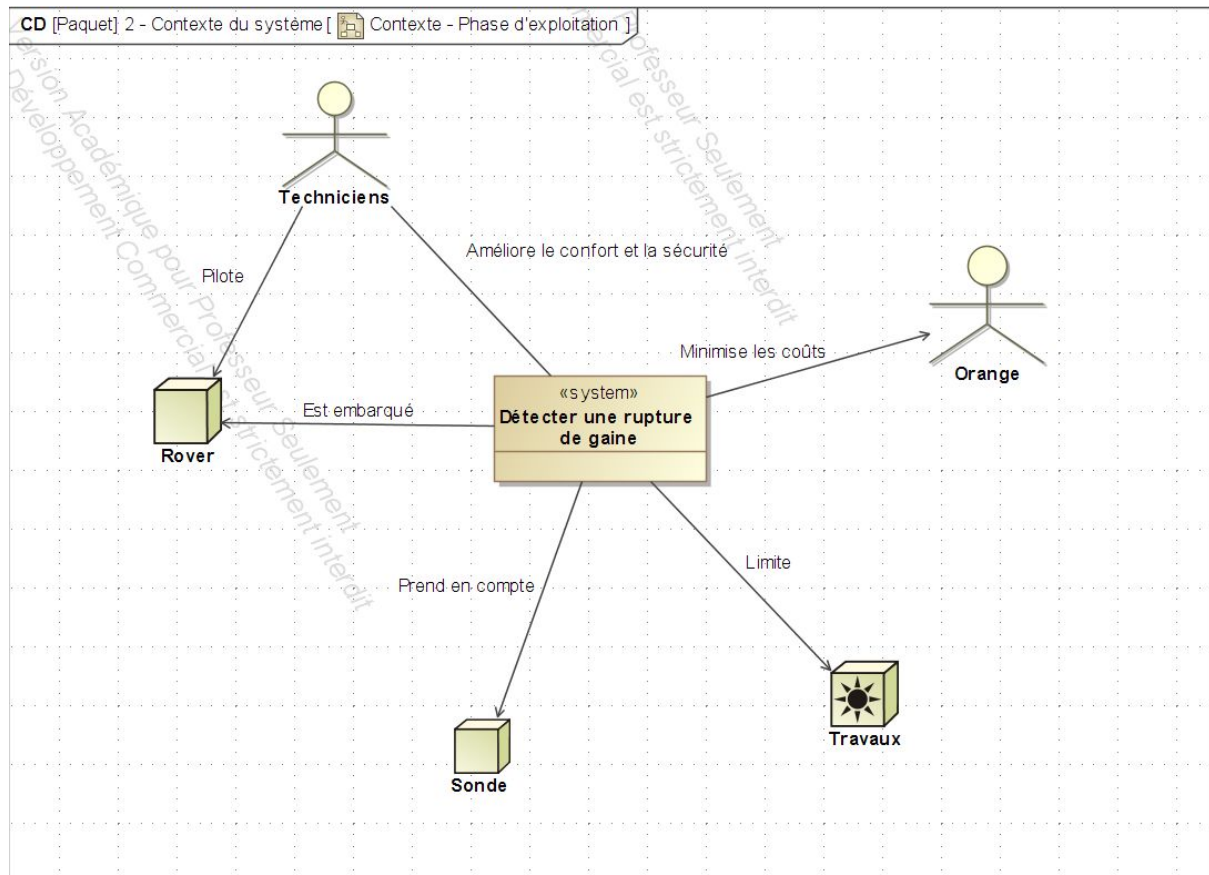
Les besoins de l'entreprise :



Suite à des obstructions et des ruptures des conduites, l'entreprise Orange nous à contacter pour que l'on puisse localiser les emplacements de futurs ruptures via un dispositif afin de résoudre ces problèmes. Ce dispositif doit bien entendu pouvoir être contrôlé et surveillé par un technicien lors d'interventions. Cela permettra de minimiser les besoins en techniciens lors des interventions.

Dudzinski Théo / Dufour Jordan / Hadoux Benjamin / Gerzynski Gaetan

Contexte du projet :



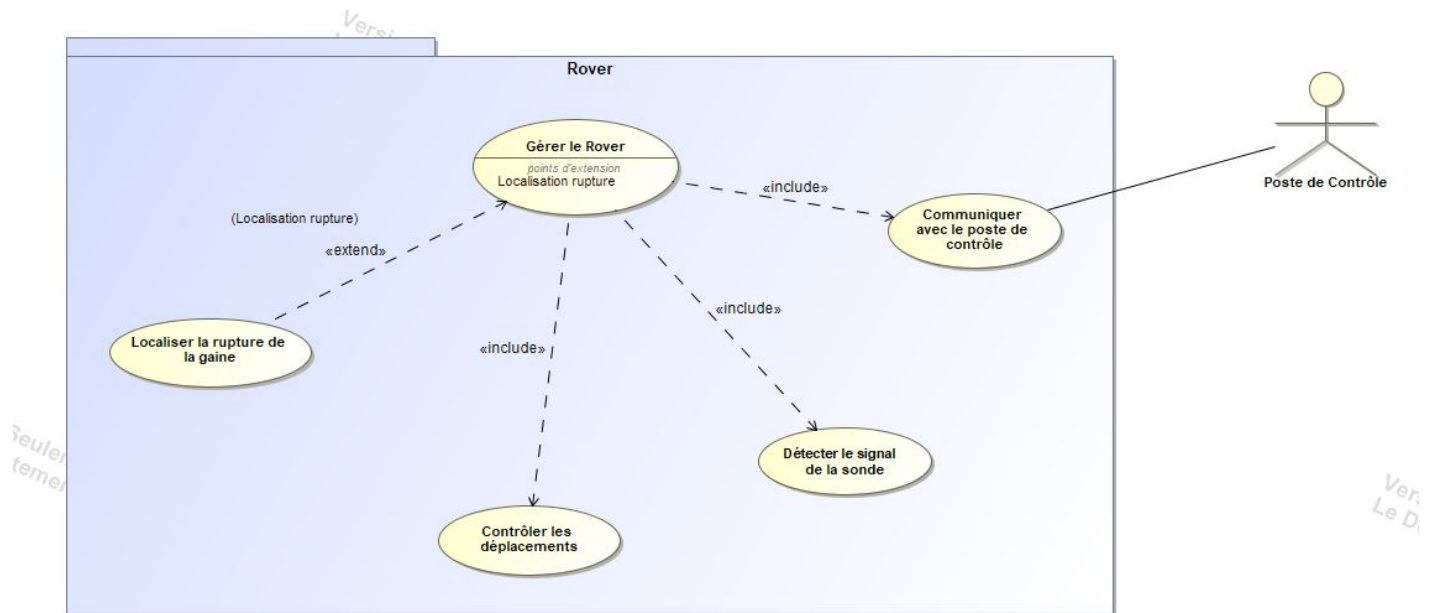
Le technicien utilise un rover afin de détecter les problèmes dans la gaine (obstruction, effondrement, etc...). Cela permet l'amélioration du confort et de la sécurité pour le technicien tout en minimisant les coûts pour l'entreprise car il y aura des travaux plus limités.

Partie Commune

Modélisations UML

Dudzinski Théo / Dufour Jordan / Hadoux Benjamin / Gerzynski Gaetan

Diagramme des cas d'utilisations : **Partie Rover :**



Descriptif :

Communiquer avec le poste de contrôle :

Cette fonctionnalité permet aux deux sous systèmes (poste de contrôle et le rover) de communiquer entre eux. Le protocole d'échange s'appuiera sur le TCP/IP. Les informations échangées sont les suivantes :

Rover au poste de contrôle : Le rover renvoie les informations sur le niveau du signal capté, cette information est utile au technicien pour commander les déplacements du rover.

Contrôler les déplacements :

Cette fonctionnalité permet de commander les déplacements du rover, on commande deux groupes de moteurs (ceux de droite et ceux de gauche). La commande des moteurs est assurée par un circuit spécialisé associé à un microcontrôleur qui se charge de les interpréter pour ensuite commander les déplacements.

Dudzinski Théo / Dufour Jordan / Hadoux Benjamin / Gerzynski Gaetan

Détecter le signal de la sonde :

Deux micros sont positionnées à l'avant et deux autres micros sont positionnées à l'arrière de chaque côtés du rover, ils sont connectés à un module de détection de signal qui est chargé de filtrer et quantifier le signal puis de le transmettre à la demande de la tâche "Gérer le Rover".

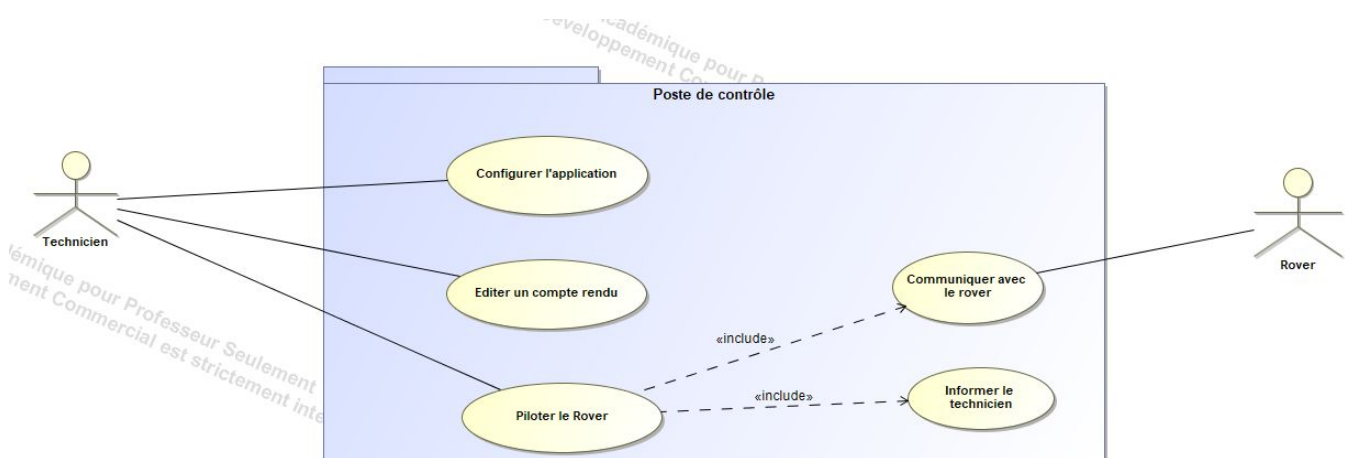
Localiser la rupture :

Ici le rover travaille de façon autonome, c'est à dire qu'il suit le signal émis par la sonde jusqu'à ce qu'il le perde. Une fois arrivé au point de rupture, il le signale au technicien à l'aide d'un signal sonore.

Gérer le Rover :

C'est l'application principale qui permet de contrôler le robot en mode automatique ou en mode manuel.

Partie Poste de contrôle :



Dudzinski Théo / Dufour Jordan / Hadoux Benjamin / Gerzynski Gaetan

Descriptif :

Configurer l'application :

A partir du poste de commande, le technicien peut régler les paramètres suivants :

- Limites de vitesse de déplacement du rover.
- La sensibilité de la raquette ainsi que sa résolution.
- Les paramètres de communication réseau.

Éditer un compte rendu :

Cette fonctionnalité permet au technicien de rédiger un compte rendu d'intervention, cela se présentera sous la forme d'un formulaire à compléter.

Piloter le rover :

A partir du pad du poste contrôle, le technicien peut commander manuellement chacun des déplacements du rover. Dans ce mode fonctionnement c'est le technicien qui contrôle tout. Il commande les mouvements du rover grâce à une raquette virtuelle.

Communiquer avec le rover :

Cette fonctionnalité permet aux deux sous systèmes (poste de contrôle et le rover) de communiquer entre eux. Le protocole d'échange s'appuiera sur le TCP/IP. Les informations échangées sont les suivantes :

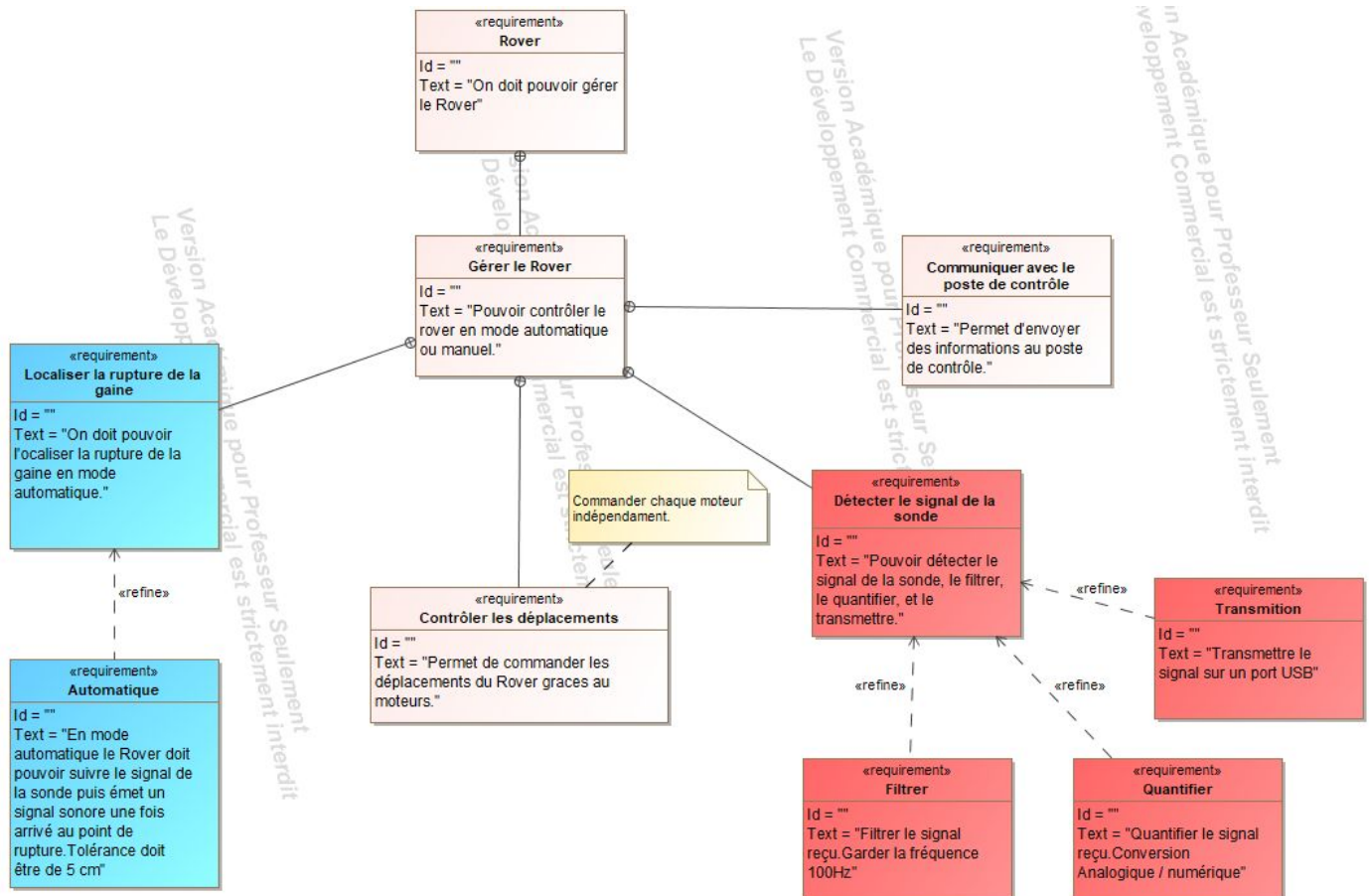
Poste de contrôle au rover : on envoie le mode fonctionnement (manuel ou automatique), en mode manuel on envoie aussi les ordres de déplacement.

Informé le technicien :

Le poste de contrôle reçoit régulièrement des informations sur la qualité du signal reçu par le dispositif de détection du signal de la sonde. Ces informations permettent au technicien de diriger son rover dans la bonne direction il aura à sa disposition une information visuelle qui sera par exemple.

Dudzinski Théo / Dufour Jordan / Hadoux Benjamin / Gerzynski Gaetan

Diagramme des exigences Partie 1 :



Localiser la rupture de la gaine :

On doit pouvoir localiser la rupture de la gaine peu importe le mode de contrôle. La tolérance est de 5 cm.

Gérer le Rover:

Il doit y avoir un mode automatique et un mode manuel.

Contrôler les déplacements :

Le Rover doit pouvoir se déplacer grâce aux moteurs.

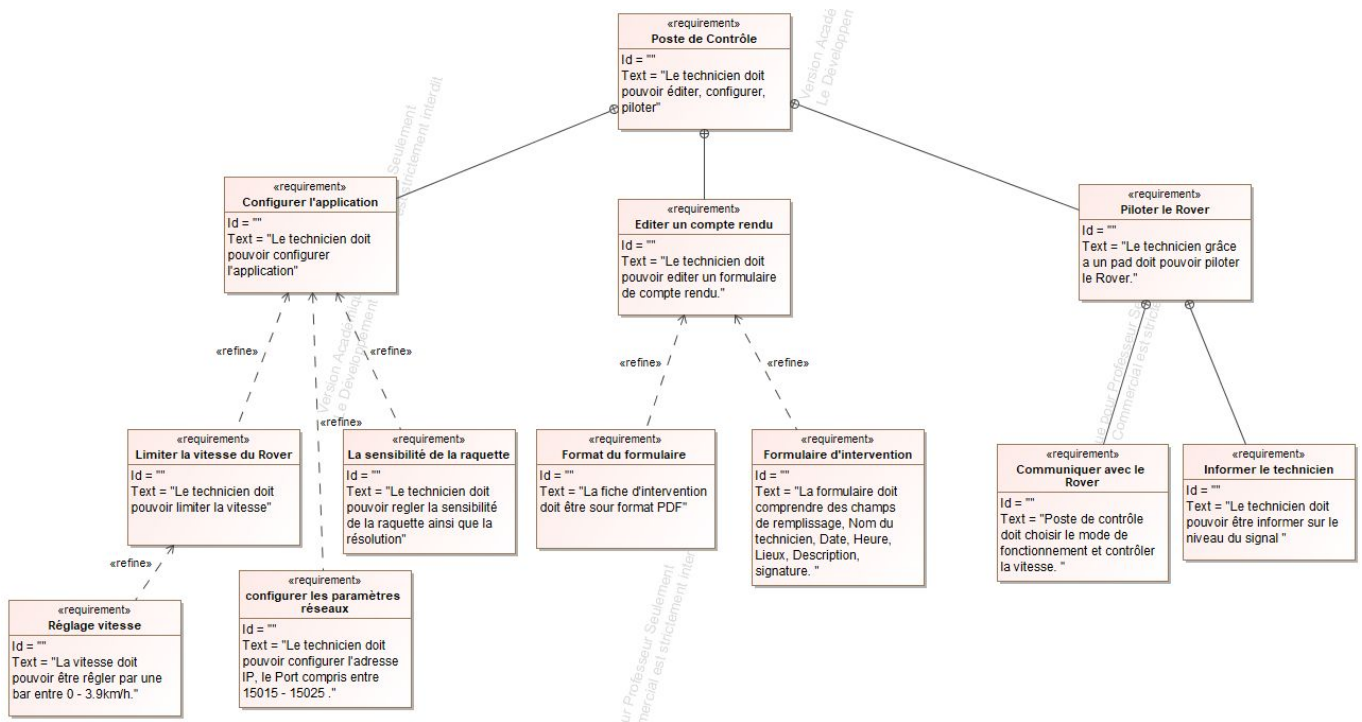
Détecter le signal de la sonde:

On doit pouvoir détecter le signal de la sonde. La filtrer c'est-à-dire que l'on doit atténuer les autres fréquences que celle retenue (100Hz). On doit pouvoir quantifier le signal reçu, une conversion analogique vers un signal numérique et pour finir on doit pouvoir transmettre le signal sur un port USB.

Communiquer avec le poste de contrôle :

Le rover doit permettre de communiquer avec le poste de contrôle afin de communiquer des informations.

Diagramme des exigences Partie 2 :



Configurer l'application :

On doit pouvoir configurer les paramètres de communication entre le rover et la tablette peu importe le mode de contrôle. Plus précisément contrôler la vitesse en y limitant ou non celle-ci, en sachant que la vitesse doit être entre 0 et 3,9 km/h. Le technicien doit aussi pouvoir régler la sensibilité ainsi que les paramètres réseaux.

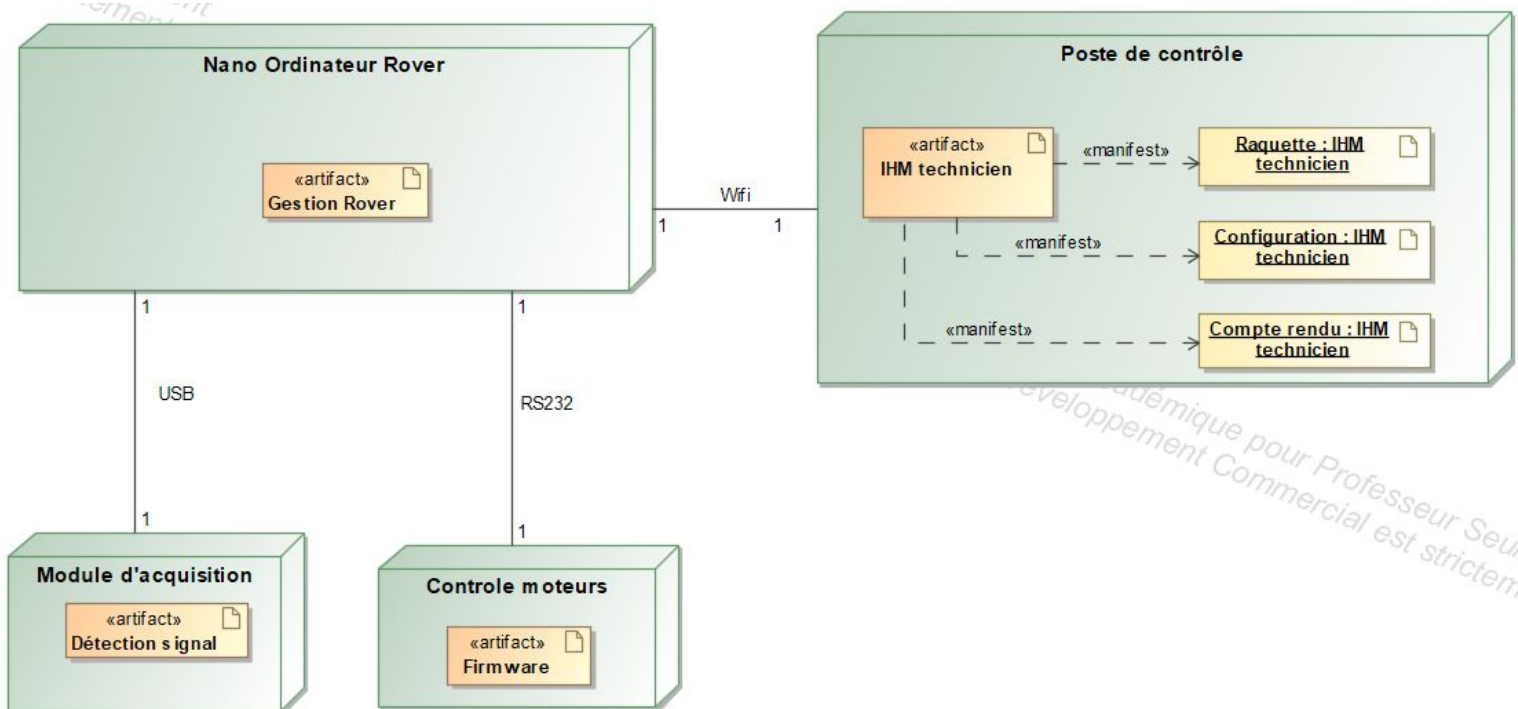
Editer un compte rendu :

La fiche d'intervention doit être sous le format PDF, ce formulaire doit être rempli par le technicien. Les informations suivantes doivent être spécifiées : Nom du technicien, la date, l'heure, le lieu, la description ainsi que la signature.

Piloter le Rover :

Le poste de contrôle doit pouvoir piloter le Rover c'est-à-dire qu'il contrôle la vitesse ainsi que son mode de fonctionnement. Le poste de contrôle doit pouvoir aussi informer le technicien du niveau de signal.

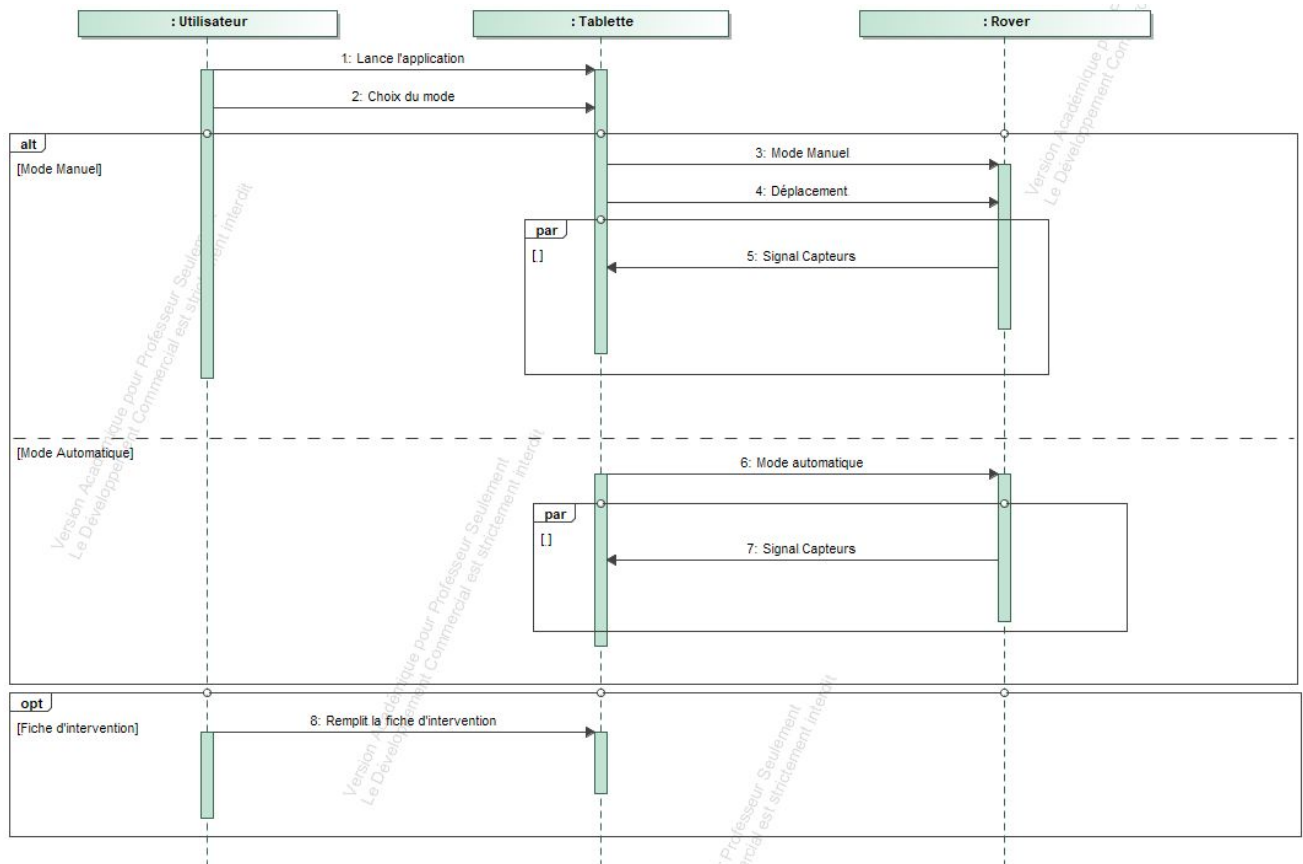
Diagramme de déploiement :



Le nano ordinateur est connecté via le Wi-fi au poste de contrôle qui est la tablette. Nous avons ensuite le contrôleur moteurs qui est connecté au nano ordinateur Rover via une liaison RS232 et pour finir le nano ordinateur Rover est connecté au module d'acquisition via USB

Dudzinski Théo / Dufour Jordan / Hadoux Benjamin / Gerzynski Gaetan

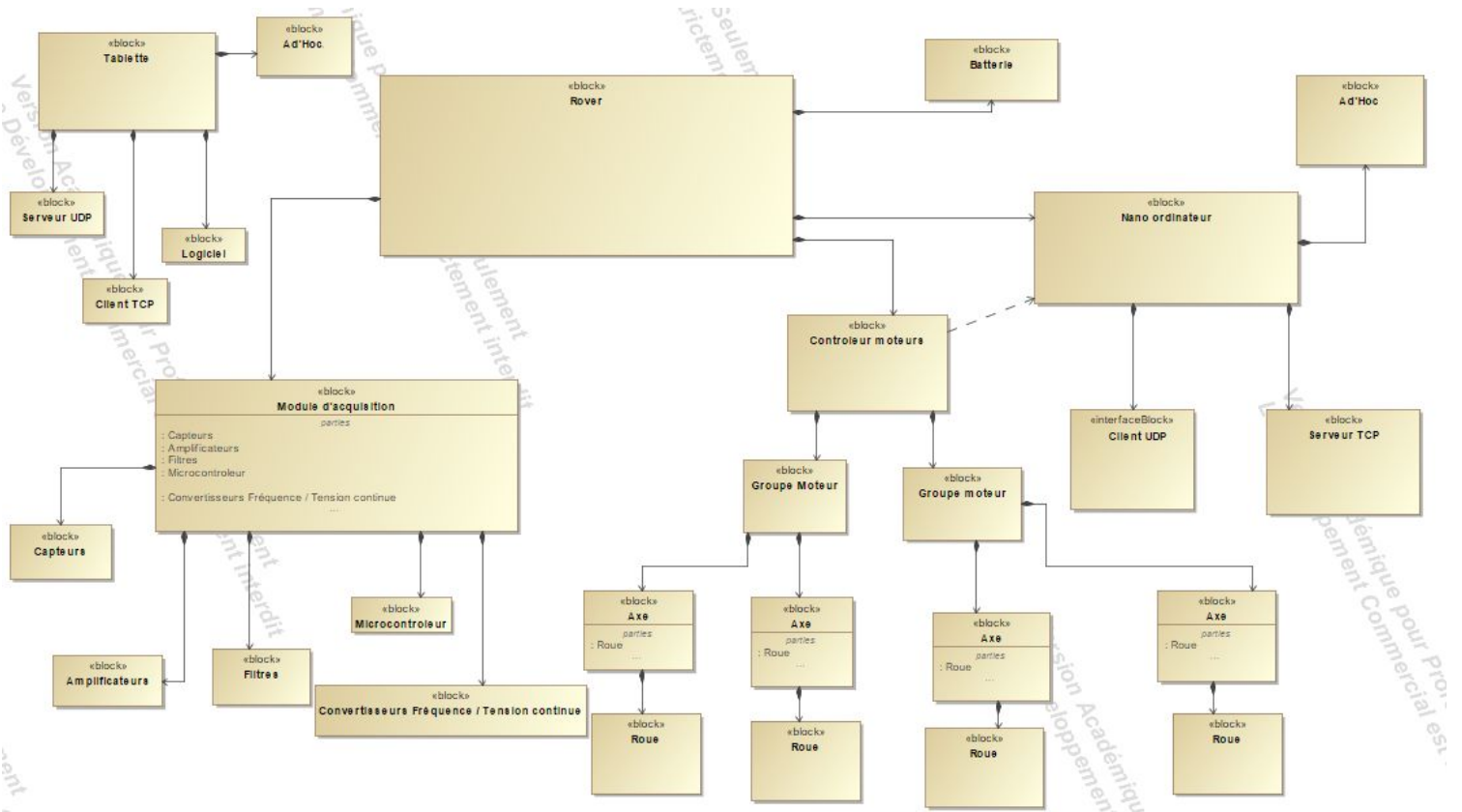
Diagramme de séquence (Système) :



Tout d'abord l'utilisateur lance l'application, ensuite l'utilisateur choisit le mode manuel ou le mode automatique. Si l'utilisateur choisit le mode manuel, il envoie les déplacements et le Rover renvoie les signaux des capteurs (en pourcentage). Pour le mode automatique, le Rover se déplace tout seul mais il renvoie aussi les informations des capteurs sur la tablette. Il y a aussi une option qui permet à l'utilisateur de remplir une fiche d'intervention sur la tablette.

Dudzinski Théo / Dufour Jordan / Hadoux Benjamin / Gerzynski Gaetan

Diagramme de définition de bloc :



Le rover est composé d'un module d'acquisition, d'un contrôleur moteur, d'un nano ordinateur et d'une batterie.

Le module d'acquisition est composé de capteurs d'amplificateurs, de filtres, d'un microcontrôleur et de convertisseurs fréquence / tension continue.

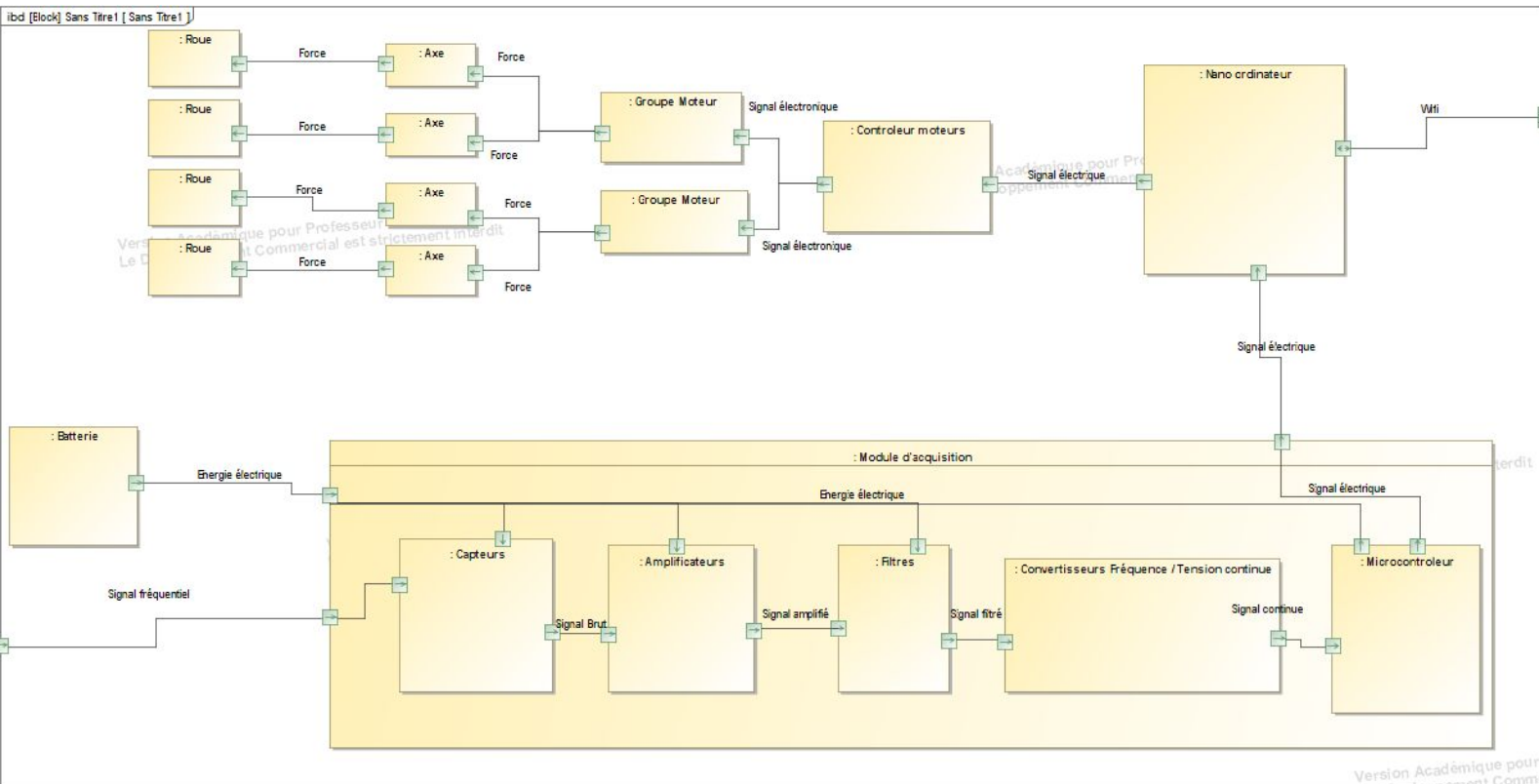
Le contrôleur moteurs est composé de deux groupe moteur qui eux même à deux axes et chaque axe à une roue.

La nano ordinateur du rover est composé d'un client UDP puis d'un serveur TCP et il est configuré en Ad'Hoc

Il y aussi la tablette qui elle a une configuration Ad'Hoc, d'un serveur UDP, un client TCP et d'un logiciel

Dudzinski Théo / Dufour Jordan / Hadoux Benjamin / Gerzynski Gaetan

Diagramme de définition de bloc interne :



Voici le diagramme de définition de bloc interne. La batterie alimente le module d'acquisition, un signal fréquentiel puis en sort une signal électrique qui est envoyé au nano ordinateur. Celui ci envoie le signal au contrôleur moteur ensuite au groupe moteur et celui-ci la transforme en force pour les axe qui entraîne les roues.

Répartition des tâches :

Tâches	Étudiants
Détection du signal	Dudzinski Théo
IHM Technicien	Jordan Dufour
Déplacements	Hadoux Benjamin
Gestion du Rover	Gerzynski Gaëtan

Descriptif :

Détection du signal :

La partie "Détection du signal" sert à réaliser un module d'acquisition permettant de capter le signal provenant d'un émetteur dans une sonde afin de savoir la position du rover selon celle-ci ou il se trouve la ou il y a un problème dans la gaine. Ce module devra filtrer, quantifier puis transmettre les informations des capteurs.

IHM Technicien :

La partie "IHM Technicien" sert à réaliser une application permettant de pouvoir contrôler le rover à distance, que cela soit automatique ou géré par le technicien, de configurer les paramètres afin de pouvoir changer comme bon nous semble en fonction des paramètres du rover, ou encore, de récupérer les informations perçues par les capteurs afin d'en informer le technicien.

Déplacements :

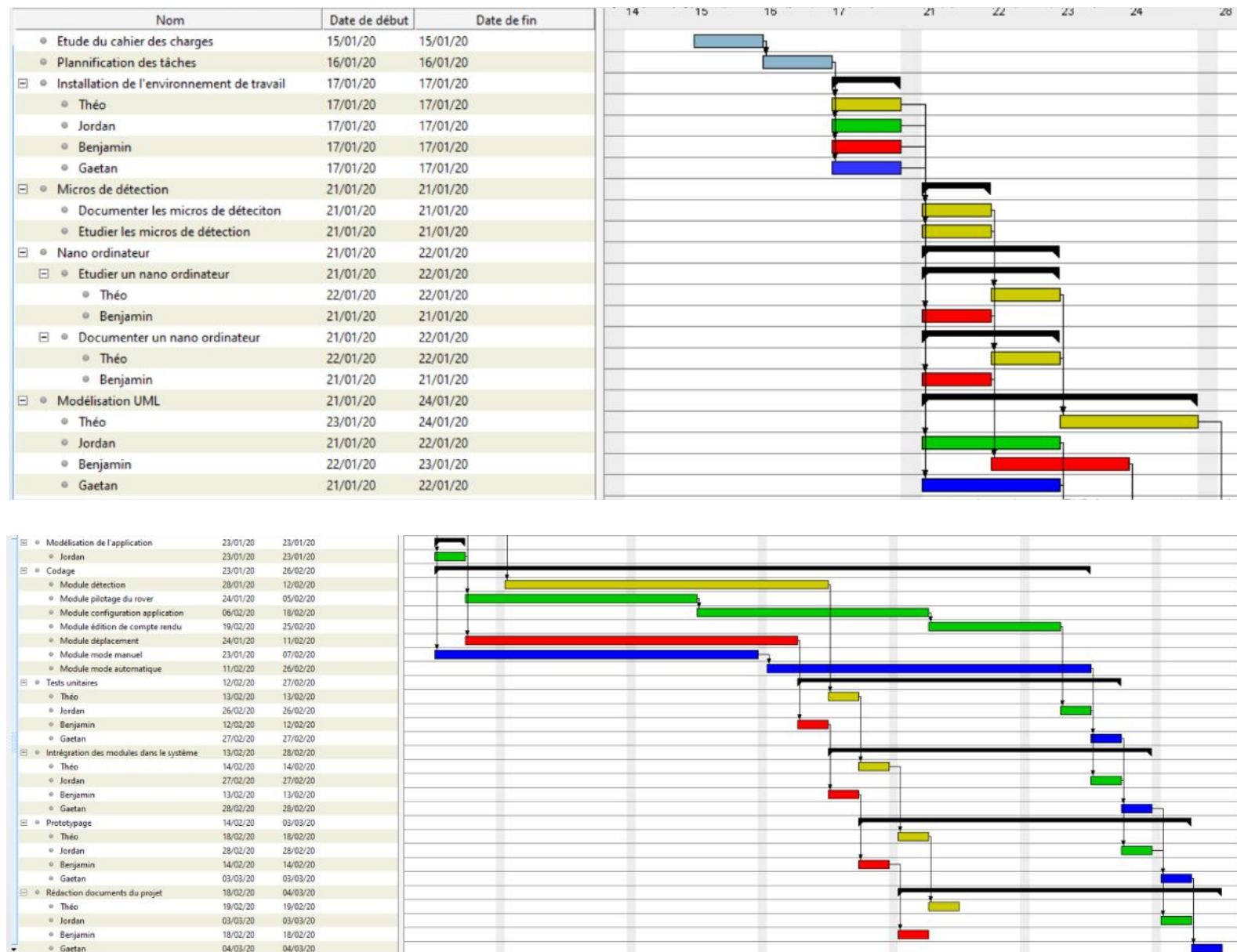
La partie "Déplacements" sert à réaliser les trames permettant de pouvoir commander le rover pour lui permettre de se déplacer

Gestion du Rover :

La partie "Gestion du Rover" sert à réaliser les calcul de direction du Rover selon les information reçu par les capteur afin d'avoir un déplacement automatique.

Dudzinski Théo / Dufour Jordan / Hadoux Benjamin / Gerzynski Gaetan

Planning général :



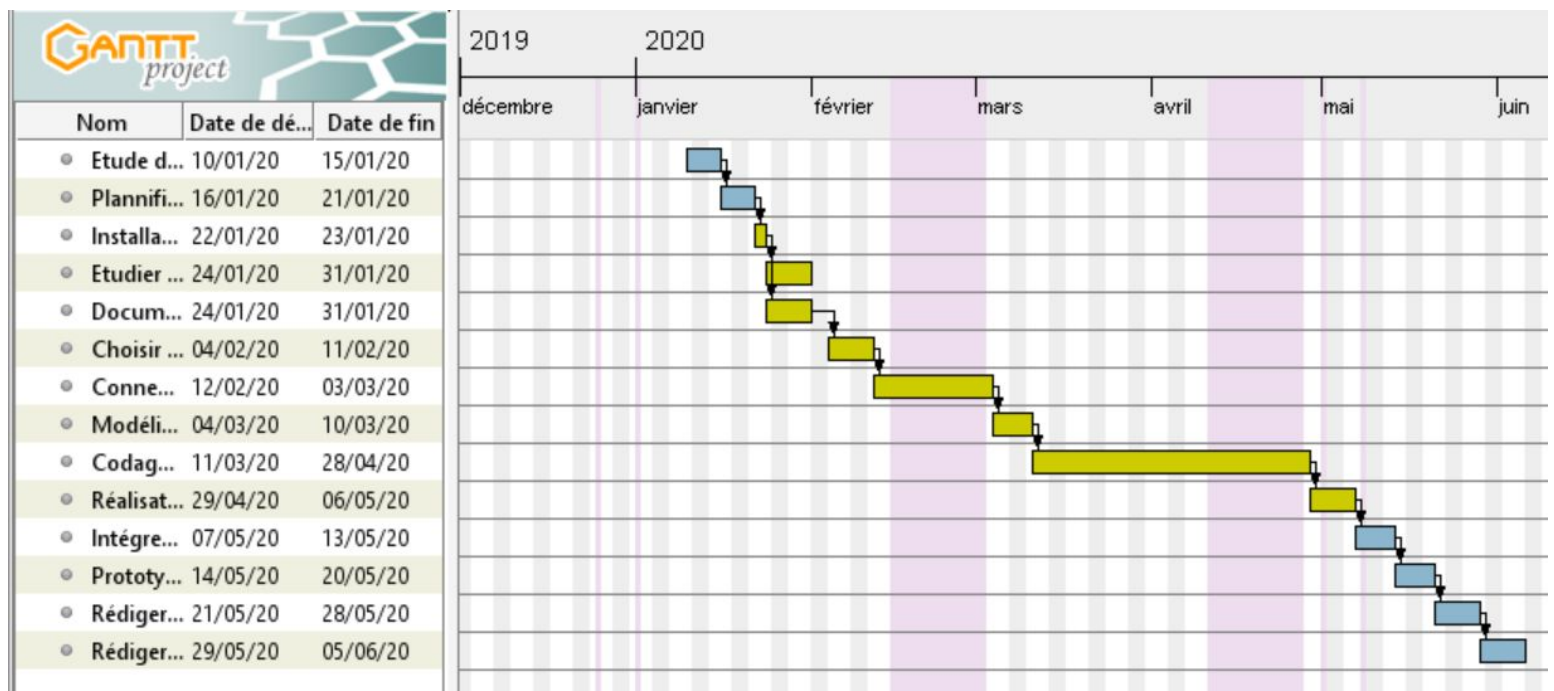
Dudzinski Théo / Dufour Jordan / Hadoux Benjamin / Gerzynski Gaetan

Partie

Détection signal

Dudzinski Théo

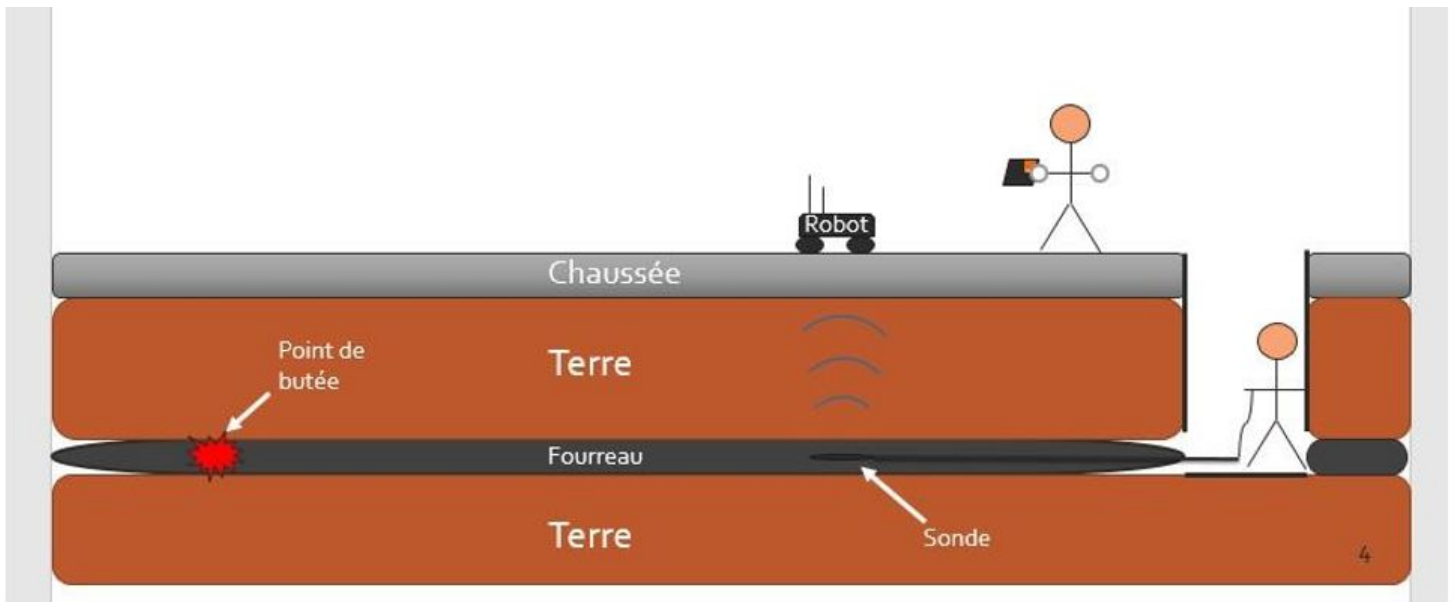
Planning prévisionnel :



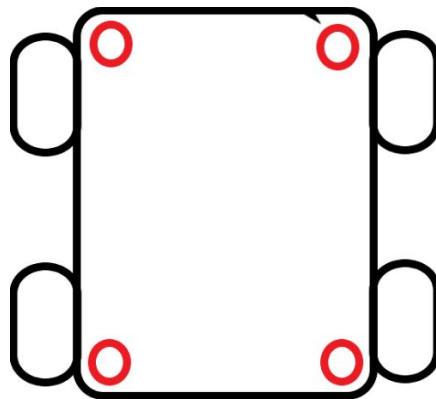
Notre projet débute en début Janvier avec l'étude du cahier des charges et celui-ci finit en début Juin avec la rédaction du dossier technique. Notre projet devrait durer 5 mois. Ce planning comprend l'étude, le choix des matériels, la programmation et le prototypage et la rédaction du dossier technique.

Dudzinski Théo

Solution technique envisagée :

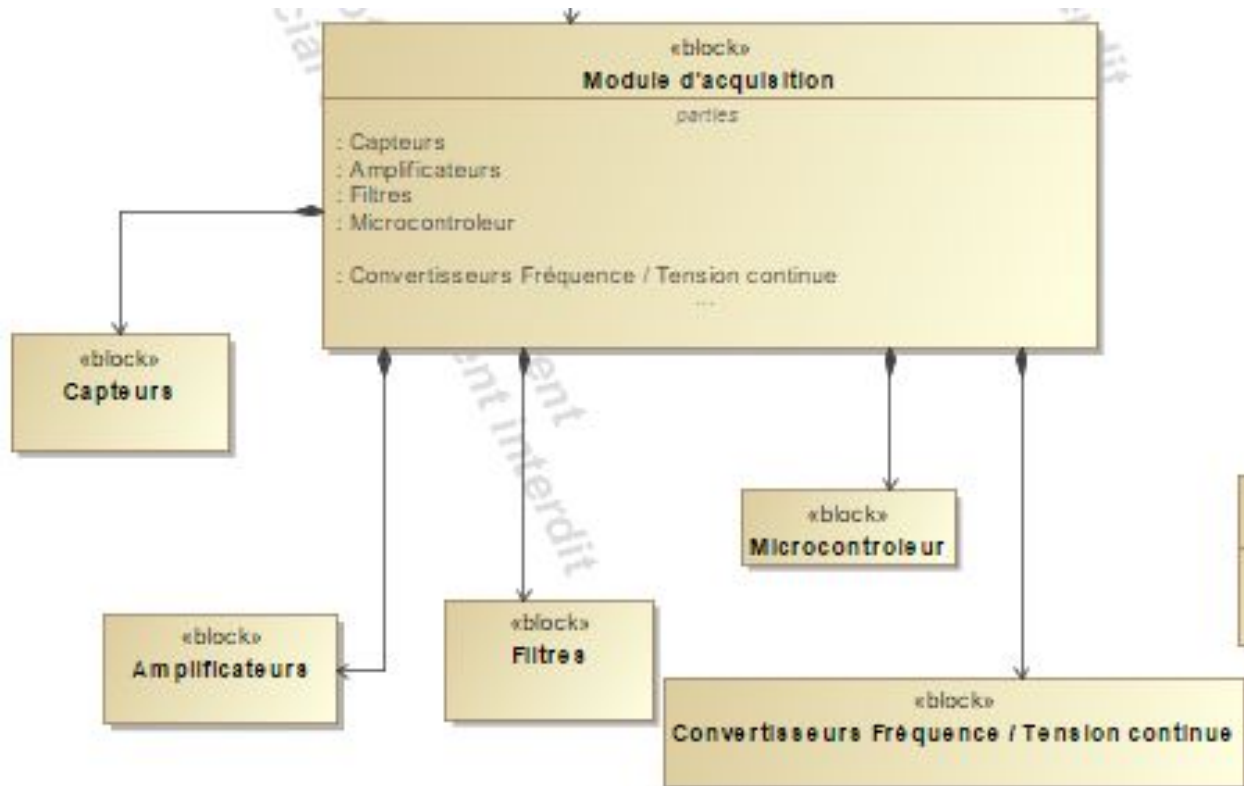


Une sonde sera mise dans la gaine, cette sonde émettra à une certaine fréquence qui sera de 100 Hz. Le robot sera doté de 4 capteurs afin de capter la fréquence émise. La sonde sera bloquée au point de butée donc si le robot avance encore l'amplitude du signal sera moins bonne en conséquence, on pourra situer l'endroit où il y a la rupture. Les capteurs sont situés aux 4 coins du rover. Voir représentation ci-dessous



Dudzinski Théo

Diagramme de définition de bloc :

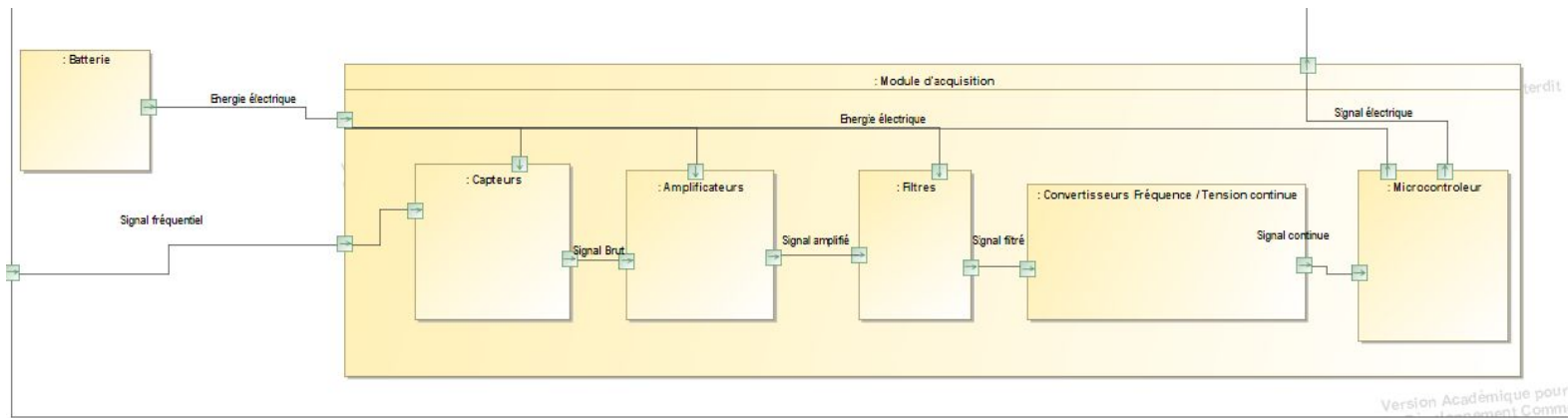


Module d'acquisition :

Le module d'acquisition est composée de capteurs, d'amplificateur, de filtre , d'un microcontrôleur et de convertisseur Fréquence / Tension continue. Comme il y a 4 capteurs, il y aura 4 module d'acquisition mais un seul microcontrôleur.

Dudzinski Théo

Diagramme de définition de bloc interne :

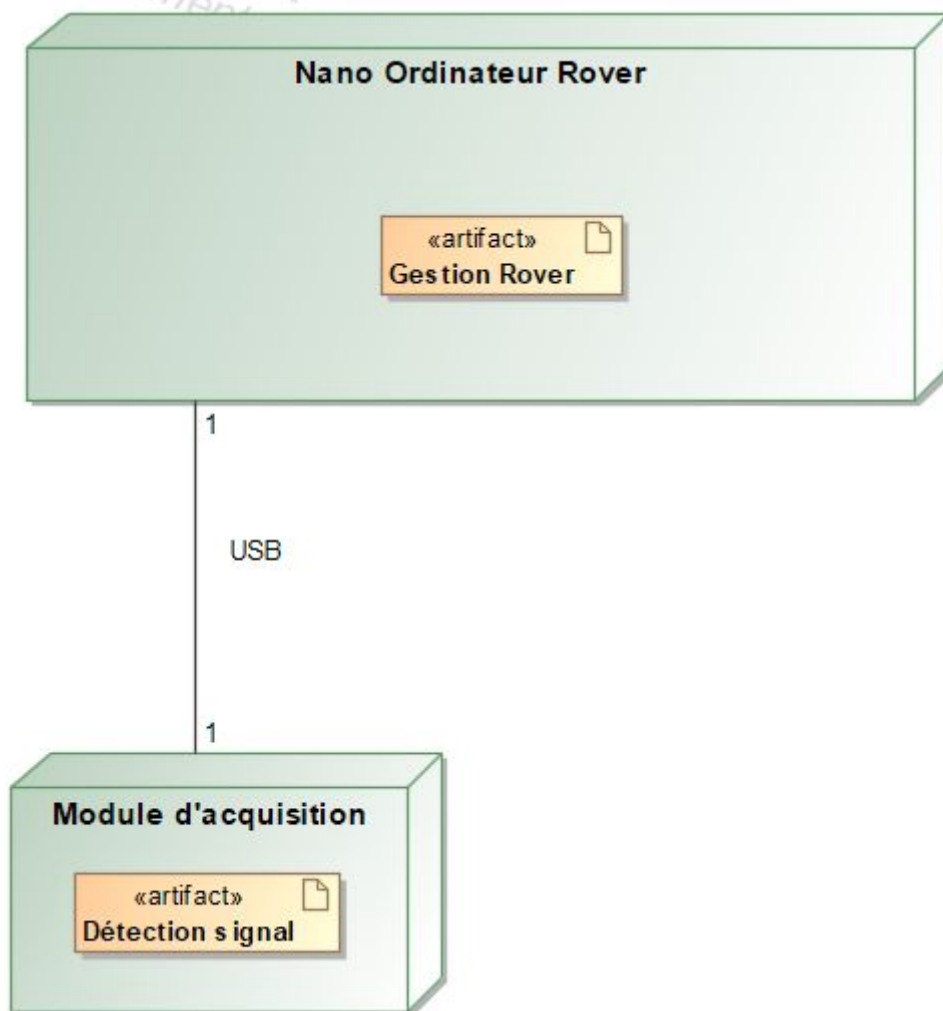


Module d'acquisition :

Tout d'abord, Une batterie se chargera d'alimenter le module d'acquisition. Ensuite, le capteurs va capter le signal fréquentiel puis va envoyer le signal brut à l'amplificateur, celui-ci va envoyer le signal amplifié au filtre pour que celui-ci ne garde que la fréquence voulus ce filtre va envoyer le signal filtré au convertisseur Fréquence / Tension continue pour transformer un signal sinusoïdale en signal continue puis le signal continue est transmise au microcontrôleur qui traite ce signal pour enfin l'envoyer au Nano Ordinateur du Rover.

Dudzinski Théo

Flux d'information :



Détection signal -> Nano Ordinateur Rover :

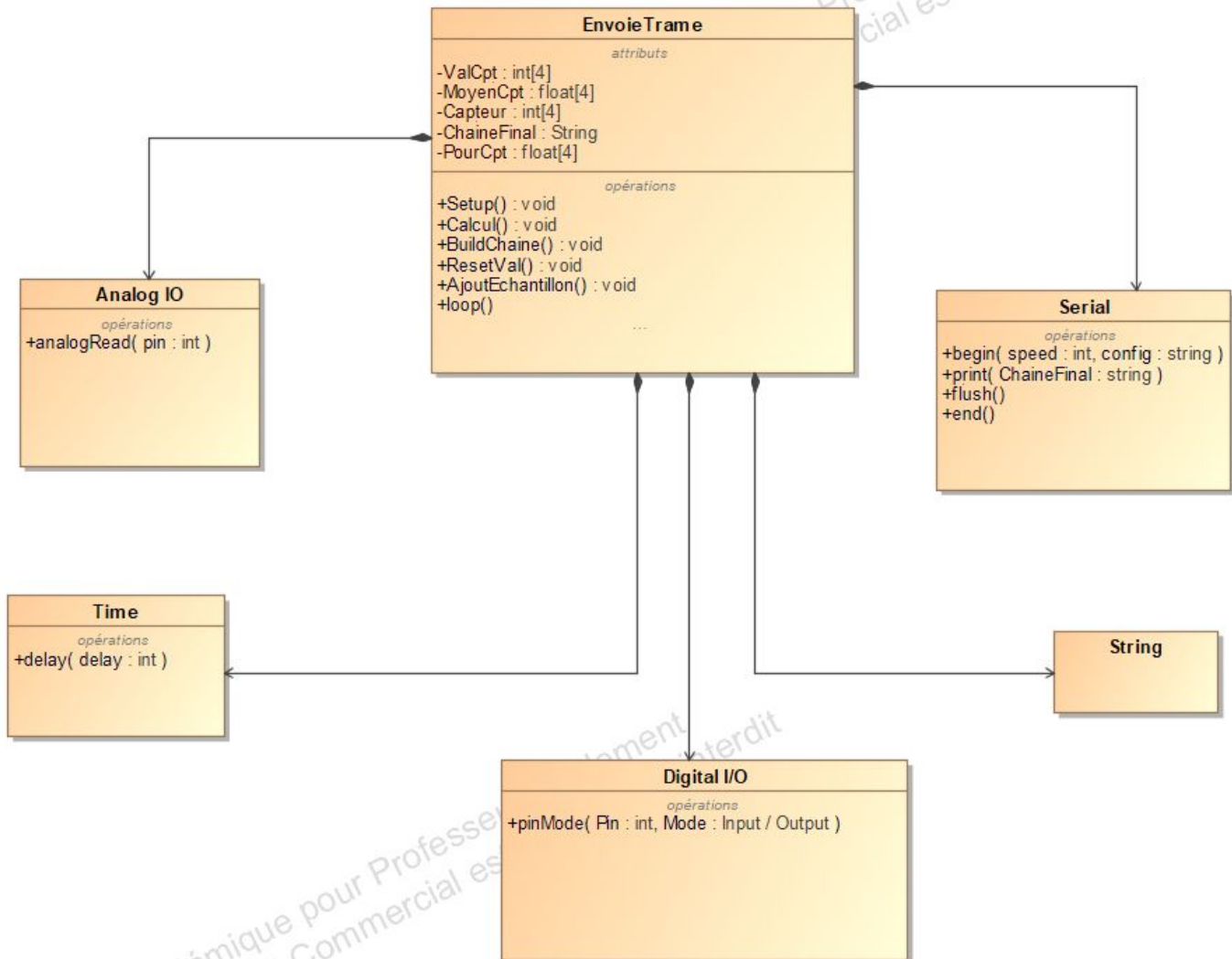
La détection signal communique avec le Nano Ordinateur Rover par le biais de la liaison USB.

Les données de la trame sera sous forme de : 75.21/79.21/40.56/45.12

Ces données sont les pourcentage de réception des capteurs.

Dudzinski Théo

Diagramme de classe (Envoie de données):



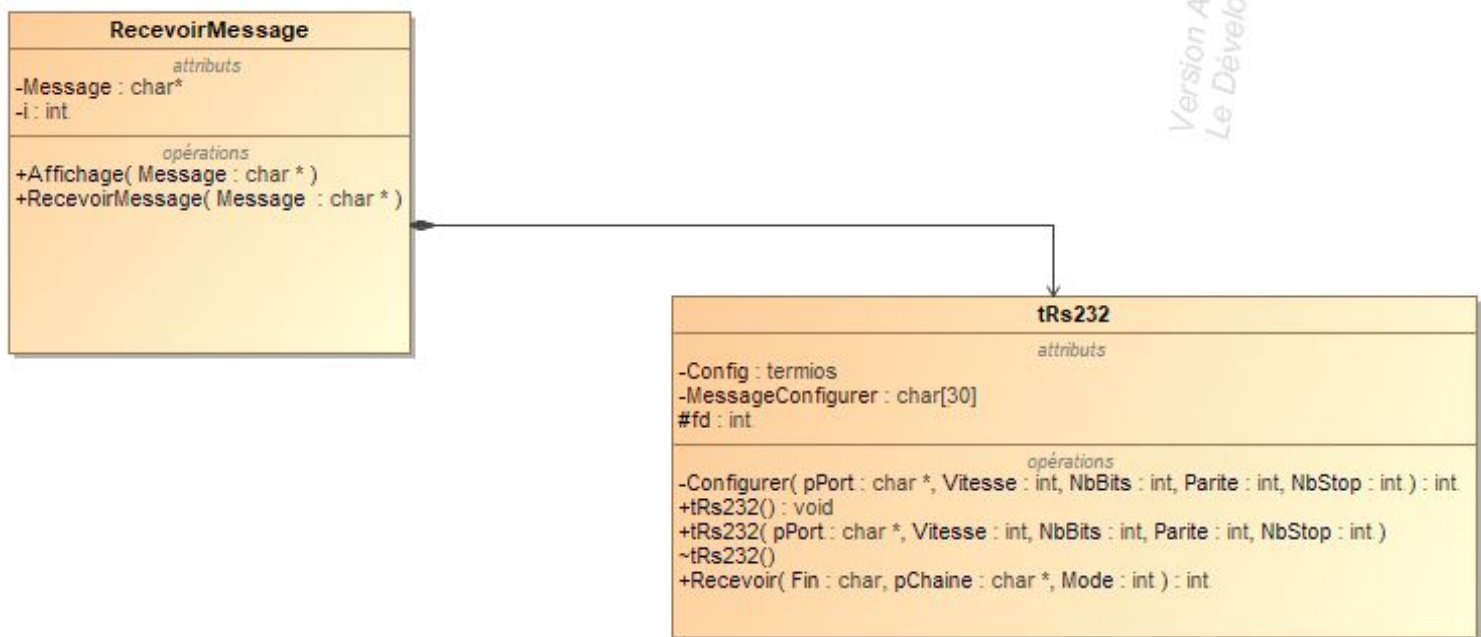
Le programme EnvoieTrame est composée de la classe Analog IO , Serial , Time, String et Digital I/O.Ce programme sert à envoyer une trame du module d'acquisition au Rover.

La classe Analog IO à comme méthode analogRead celle-ci permet de lire les valeurs sur les broches analogique de l'arduino.

La classe Serial permet de "contrôler" le port de l'arduino. Begin() sert à ouvrir le port.Print() sert à envoyer sur le port USB une trame. Flush() sert à supprimer les données en mémoire tampons et la méthode End() sert à fermer le port USB.

Dudzinski Théo

Diagramme de classe (Réception de données):



Le programme RevoirMessage est composée de la classe tRs232.

tRs232 à comme méthode Configurer(...), tRs232(), tRs232(...), ~tRs232() et de la méthode Recevoir(...)

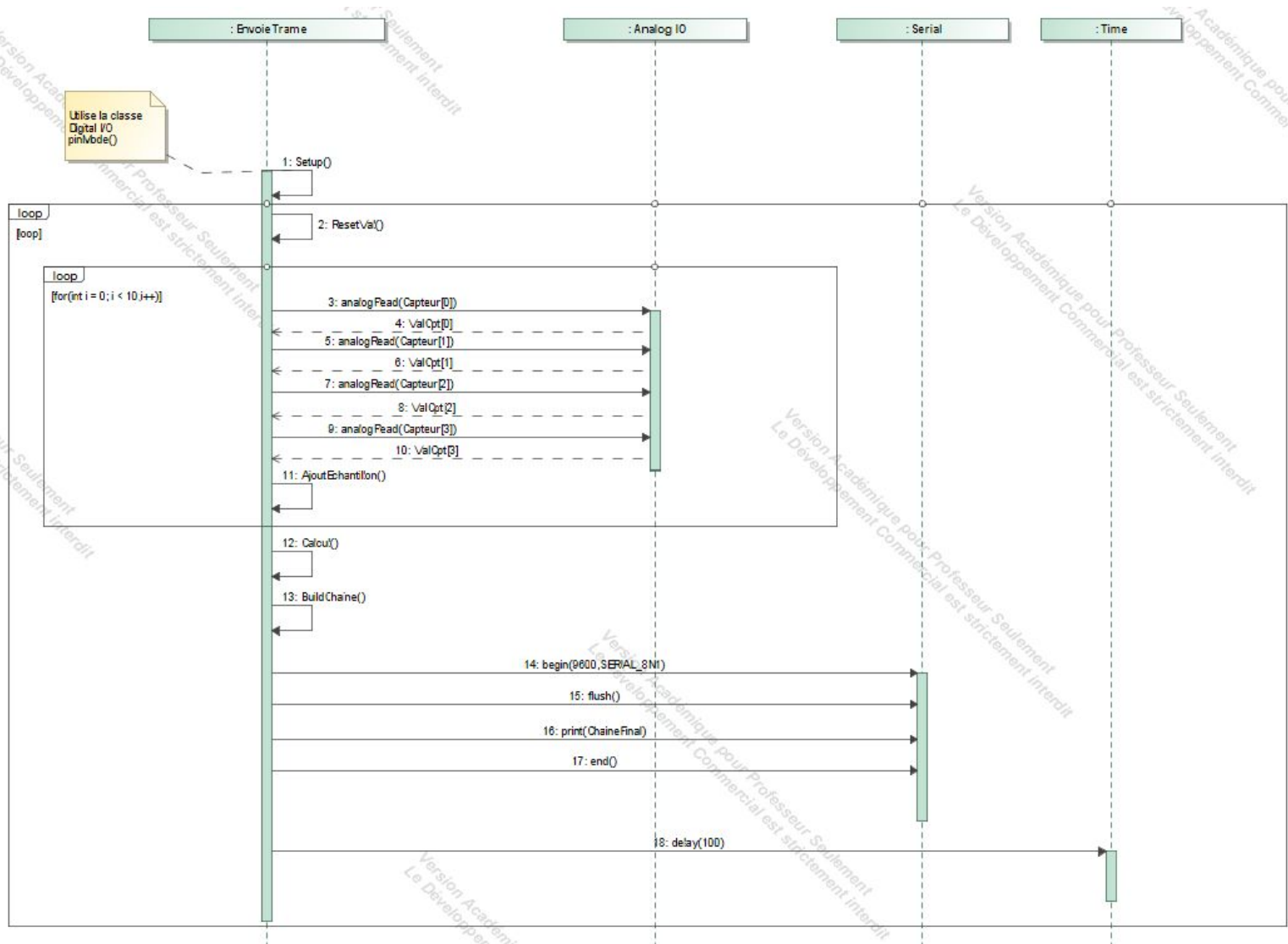
- Configurer(...) : Cela sert à configurer le port auquel on va recevoir les informations
- tRs232() est le constructeur sans surcharge
- tRs232(.....) est le constructeur pour lequel on peut préciser le nom du port, la vitesse, le nombre de bits de données , la parité et le nombre de bits de stop
- ~tRs232() est le destructeur
- Recevoir(...) cela sert à recevoir les informations on a comme surcharge le caractère de fin ensuite on a dans lequel on stocke les informations et pour finir le mode

RevoirMessage à deux méthodes :

- Affichage() cela sert à afficher les informations qui ont transité sur le port série.
- RecevoirMessage() est une méthode qui permet de récupérer les informations est de la stocker.

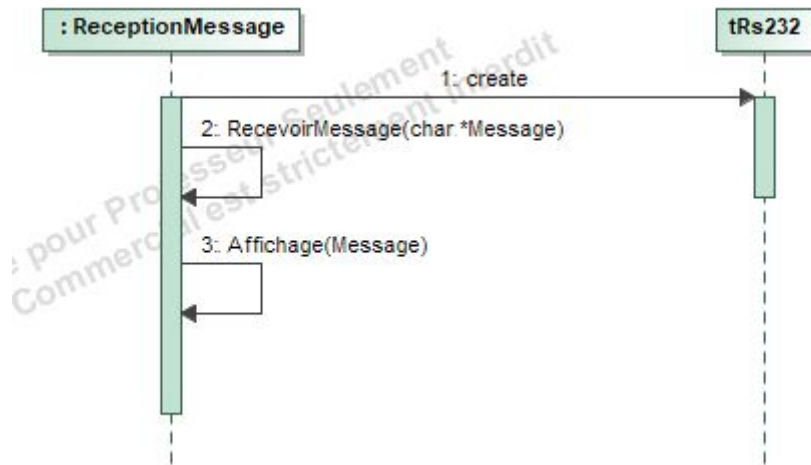
Dudzinski Théo

Diagramme de séquence du cas d'utilisation ‘Détecer le signal’ :



Au lancement, EnvoieTrame fait la méthode Setup() celle-ci sert à mettre en INPUT les broches analogiques. Ensuite on rentre dans une boucle infinie. On initialise/Réinitialise les valeurs grâce à ResetVal(). Ensuite on entre dans une boucle for pour récupérer 10 échantillons de chaque capteurs et on ajoute les échantillons grâce à AjoutEchantillon() pour on fait les calculs avec Calcul() puis on construit la chaîne avec BuildChaine(). On initialise le port avec begin(9600, SERIAL_8N1), on vide le port avec flush() puis on envoie la trame avec print(ChaineFinal) puis on ferme le port avec end() et on attend 100ms avec delay(100).

Dudzinski Théo



Ce programme se situe au niveau du Rover (Linux) Au lancement du programme, on crée un objet de type tRs232. Ensuite le programme exécute la méthode Recevoir Message(char *Message) celle-ci va entrer dans une boucle puis appelle la méthode recevoir de tRs232, on fait cela dans une boucle for pour ne pas prendre en compte les premières informations car celle-ci peuvent être incomplète donc on récupère le 4ème pour être sûr d'avoir l'information complète puis ensuite on appelle la méthode Affichage (Message) pour afficher les informations.

Dudzinski Théo

Caractéristiques principales :

Capteur
Basse fréquence
Directivité

La capteur doit pouvoir capter les basses fréquences
Le capteur doit recevoir des signaux qui viennent de face.

Nano ordinateur
4 entrée analogique
Interface de communication
Capacité calcul suffisante pour échantillonner

Le nano ordinateur doit avoir 4 entrées analogique minimum.
Une interface de communication en USB doit être présente.
La capacité de calcul doit être suffisante.

Émetteur
Basse fréquence
Taille compact
Puissance 4W minimum

L'émetteur doit pouvoir émettre les basses fréquences.
Il doit être compact et doit avoir une puissance minimum de 4W

Choix matériels :

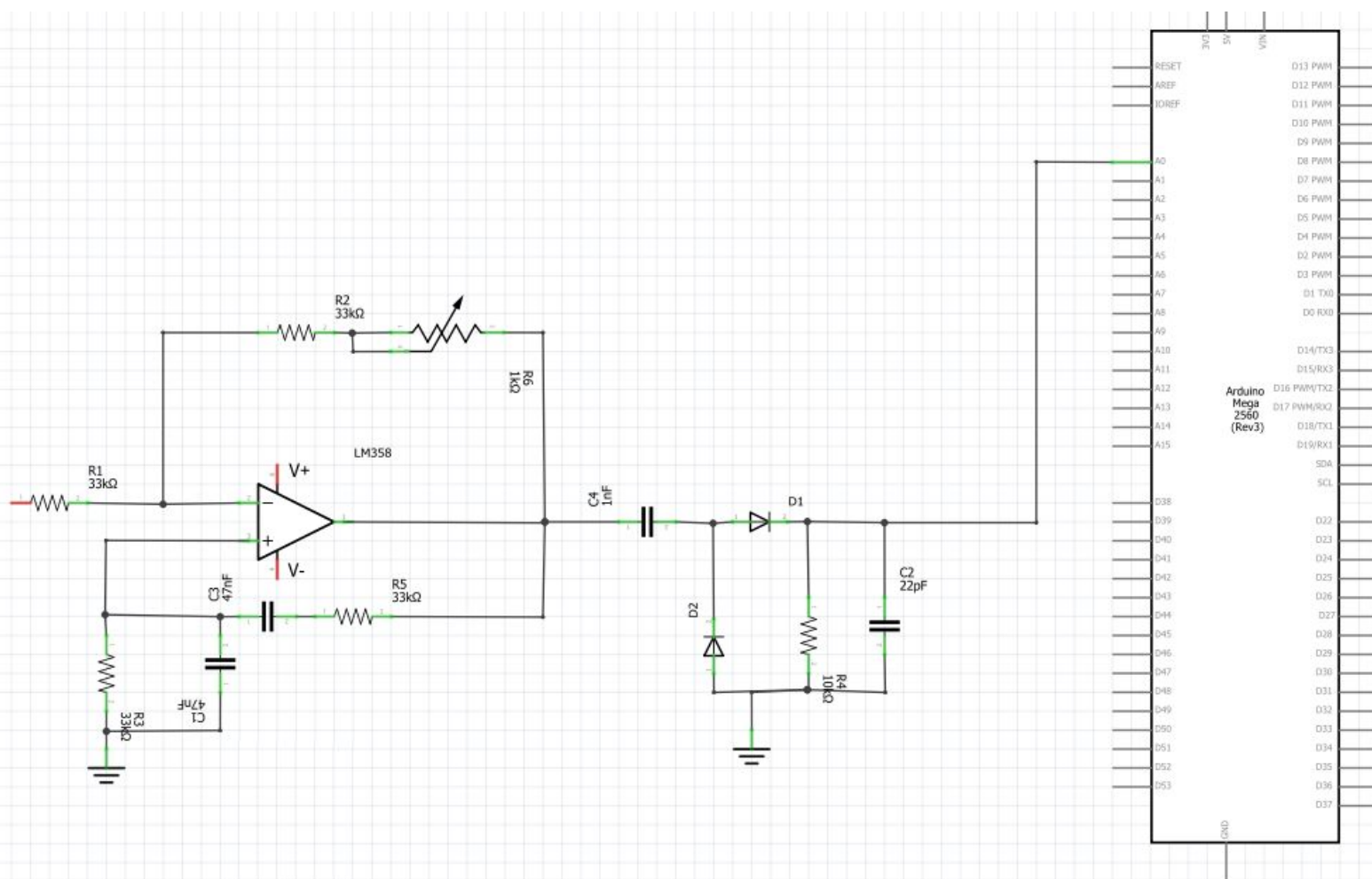
Pour les capteurs, j'ai retenue le capteur (microphone) T-bone SC 140 car celui-ci permet de capter les fréquences allant de 20Hz à 20kHz et sa directivité est cardioïde.

Pour le nano ordinateur, j'ai retenue l'arduino Mega car il comporte assez d'entrée analogiques (16 entrées analogiques) pour le moyen de communication il est doté d'un port USB. Pour la capacité de calcul ce nano ordinateur est doté du processeur ATmega2560.

Pour le haut parleur, j'ai retenue le 2055-BFC-D40-22-1-002-ND. Il peut émettre des fréquences allant de 20Hz à 20kHz, il est aussi assez miniature car il fait 39.8mm de diamètre et sa puissance peut aller de 3.5W à 8W max.

Dudzinski Théo

Schéma électrique pour un capteur :

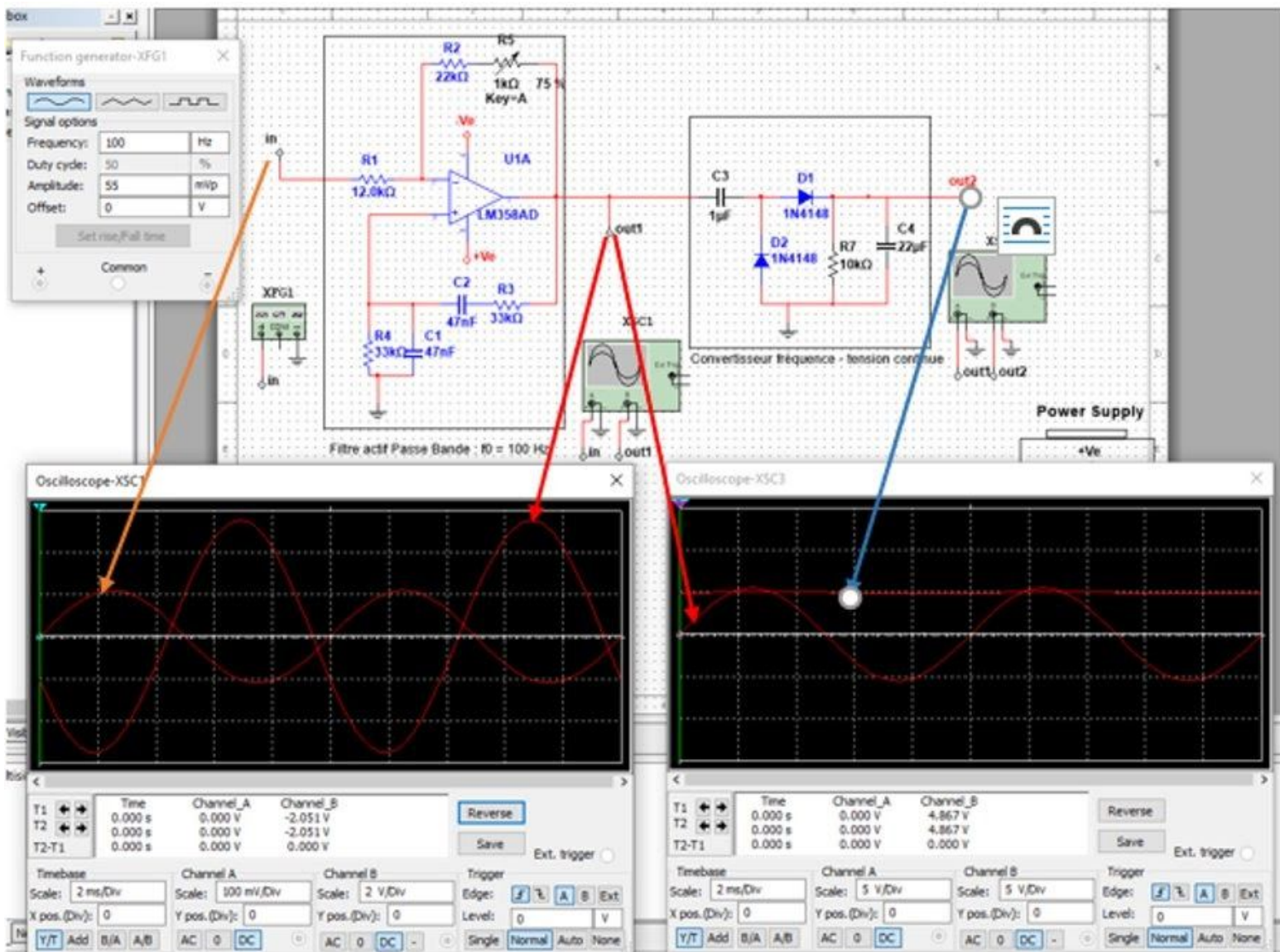


On peut voir ici le schéma électrique de la chaîne d'acquisition pour un capteur. Au début nous avons le capteur qui est positionné à gauche, il n'est pas représenté ici. Nous avons ensuite le filtre passe bande actif (filtre + amplificateur) au milieu ,nous avons le convertisseur fréquence / tension continue et ensuite à droite nous avons le nano ordinateur.

Dudzinski Théo

Simulation partie électronique :

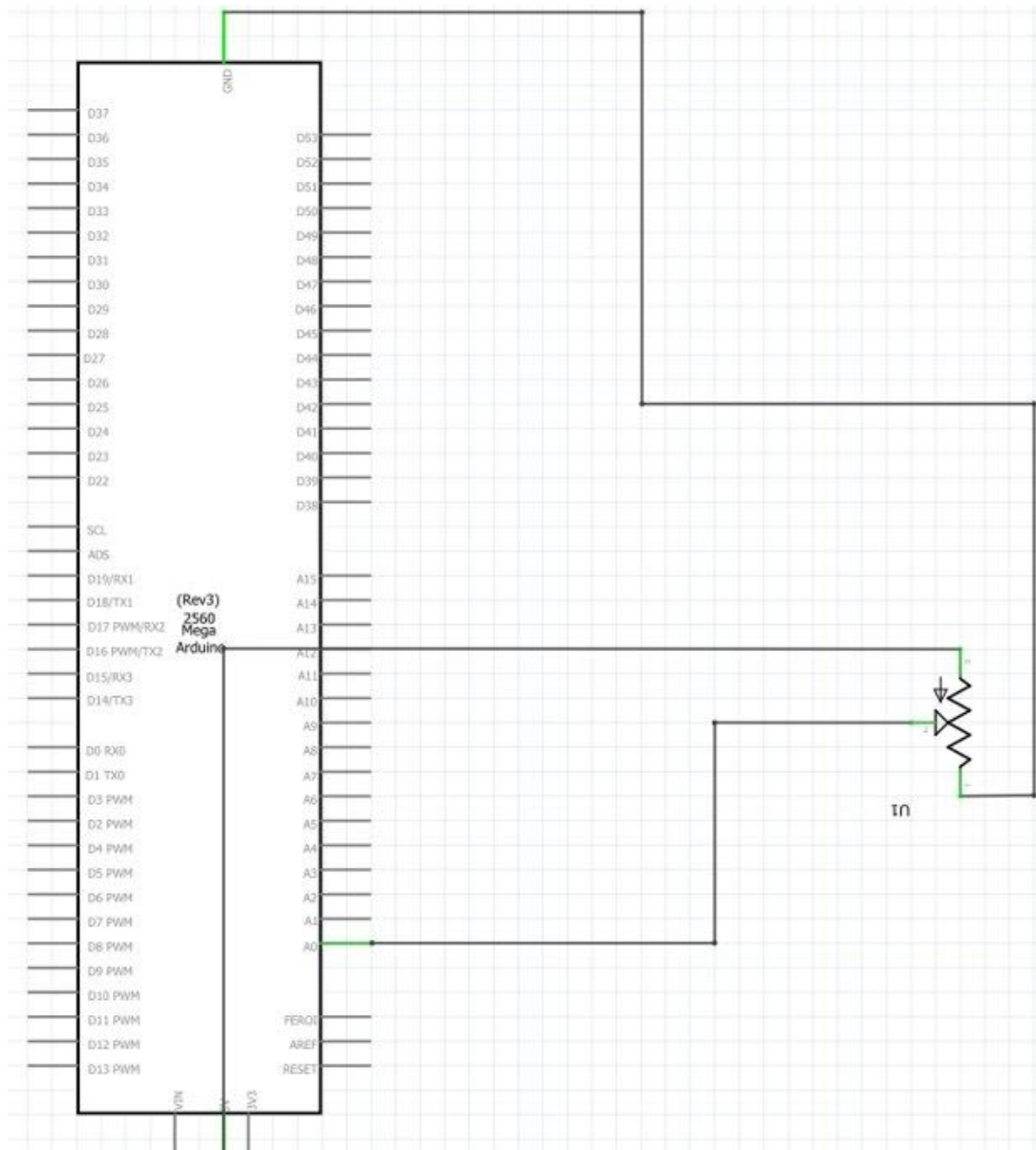
Simulation du filtre passe bande actif (Filtre + amplification) et du convertisseur fréquence / tension continue.



Le flèche orange montre le signal qui sort du capteur. La flèche rouge celle du signal après filtrage et amplification et pour finir la flèche bleu correspond au signal après avoir été converti par le convertisseur fréquence / tension continue. C'est cette courbe qui sera à l'entrée analogique de l'arduino.

Dudzinski Théo

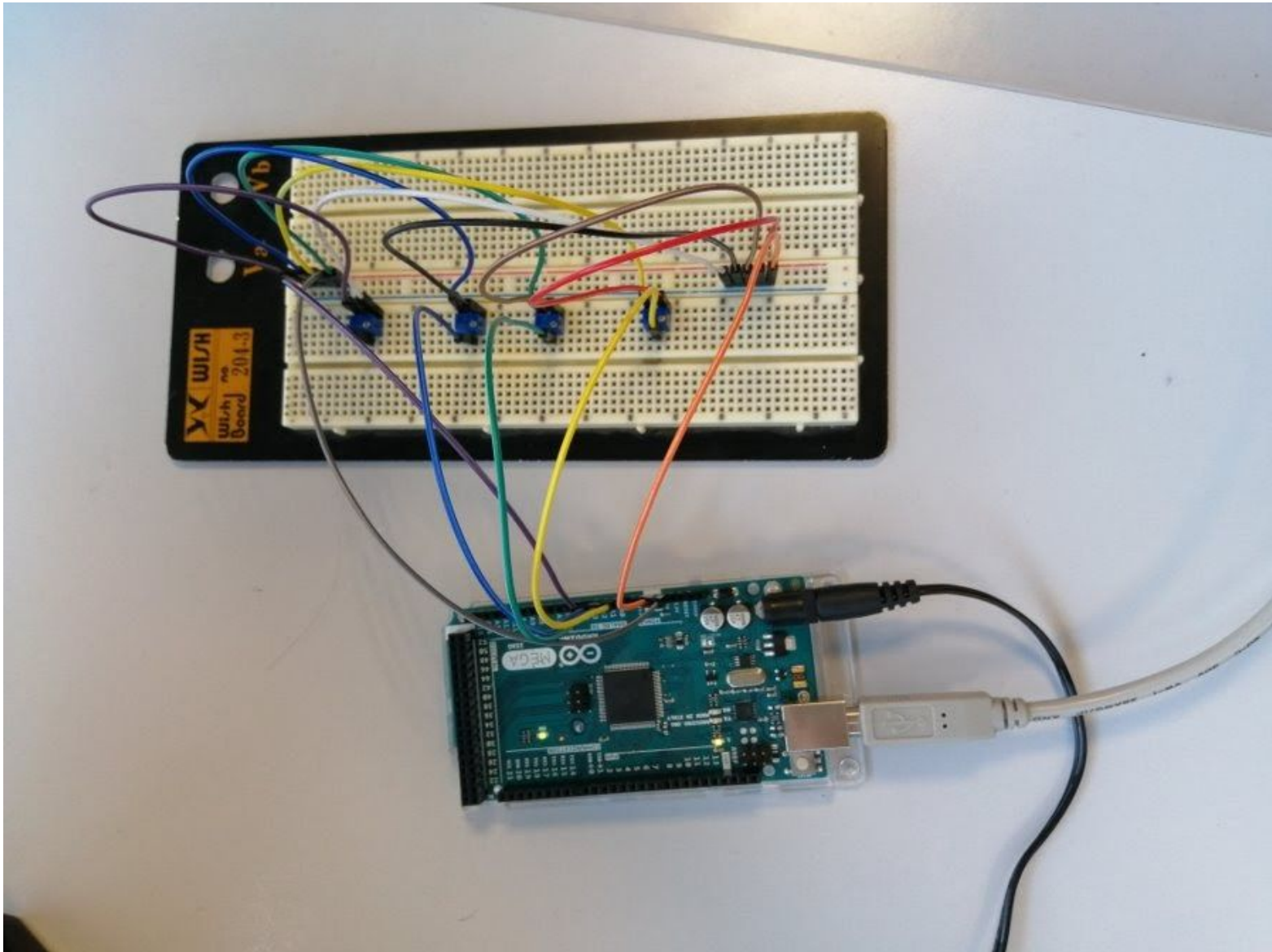
Prototypage :



Le potentiomètre représente la tension qui sera délivré par un capteur pour les simulations suivante. Pour les autres capteurs, se sera la même chose sauf qu'ils seront branchés sur les broches analogiques suivantes.

Dudzinski Théo

Prototypage :



Voici la photo du prototypage avec tous les potentiomètres connecté à l'arduino. Les potentiomètres simulent les capteurs.

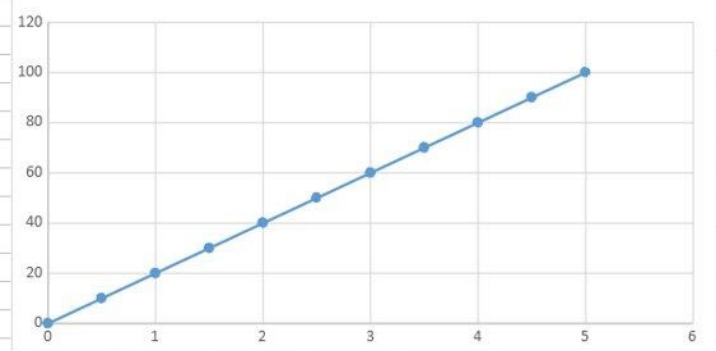
Dudzinski Théo

Tension théorique / réel des entrées analogiques

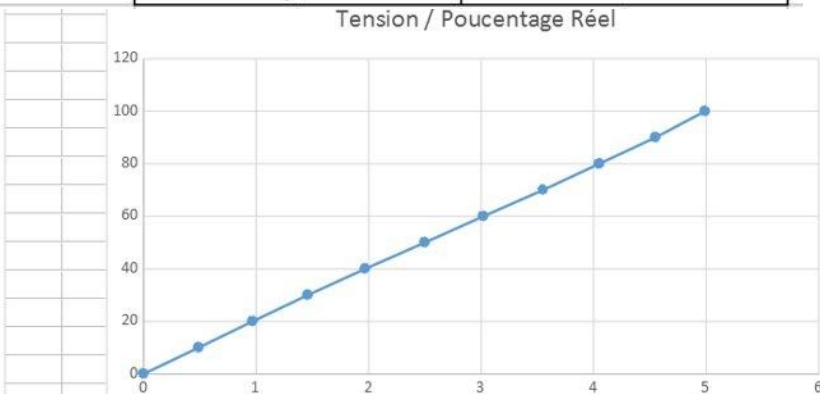
Arduino :

Théorique		Réal	
Tension (Volt)	Pourcentage	Tension(Volt)	Pourcentage
0	0	0	0
0,5	10	0,49	10
1	20	0,97	20
1,5	30	1,46	30
2	40	1,97	40
2,5	50	2,5	50
3	60	3,02	60
3,5	70	3,55	70
4	80	4,05	80
4,5	90	4,55	90
5	100	4,99	100

Tension / Pourcentage Théorique



Tension / Pourcentage Réel



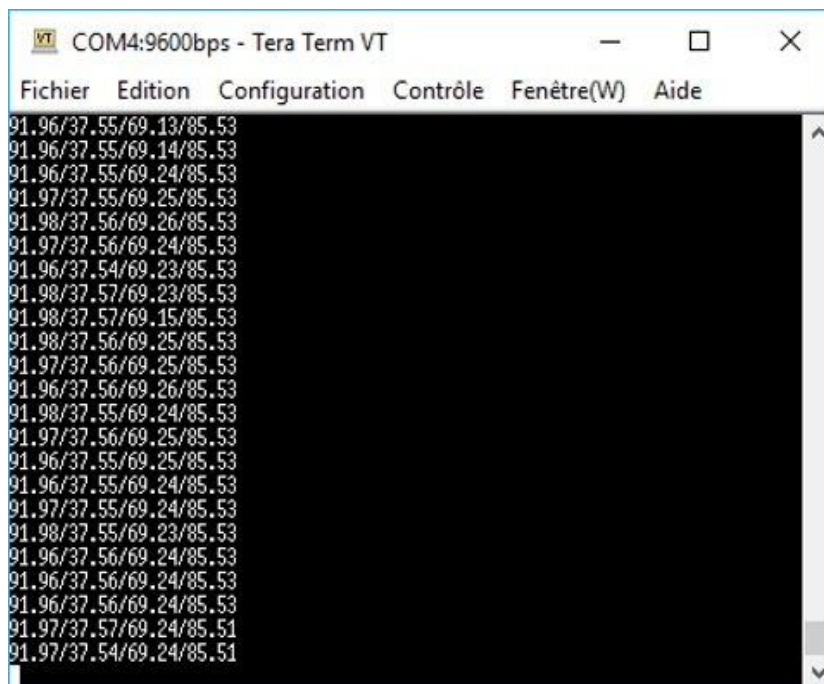
Le tableau à gauche sont les tensions et pourcentage théoriques.

Le tableau à droite sont les tensions et pourcentages réel.

On peut voir que les mesures sont relativement proche cela provient aussi du taux d'erreur du multimètre.

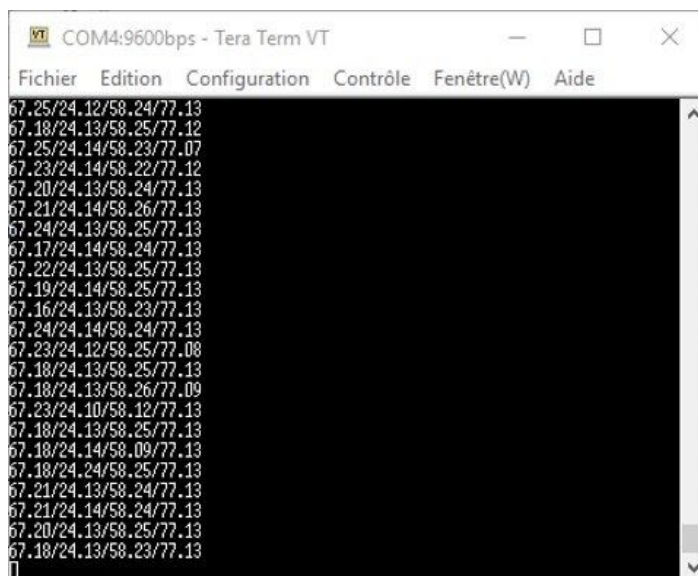
Dudzinski Théo

Test envoie des trames avec interface USB :



```

VT COM4:9600bps - Tera Term VT
Fichier Edition Configuration Contrôle Fenêtre(W) Aide
91.96/37.55/69.13/85.53
91.96/37.55/69.14/85.53
91.96/37.55/69.24/85.53
91.97/37.55/69.25/85.53
91.98/37.56/69.26/85.53
91.97/37.56/69.24/85.53
91.96/37.54/69.23/85.53
91.98/37.57/69.23/85.53
91.98/37.57/69.15/85.53
91.98/37.56/69.25/85.53
91.97/37.56/69.25/85.53
91.96/37.56/69.26/85.53
91.98/37.55/69.24/85.53
91.97/37.56/69.25/85.53
91.96/37.55/69.25/85.53
91.96/37.55/69.24/85.53
91.97/37.55/69.24/85.53
91.98/37.55/69.23/85.53
91.96/37.56/69.24/85.53
91.96/37.56/69.24/85.53
91.96/37.56/69.24/85.53
91.97/37.57/69.24/85.51
91.97/37.54/69.24/85.51
  
```



```

VT COM4:9600bps - Tera Term VT
Fichier Edition Configuration Contrôle Fenêtre(W) Aide
67.25/24.12/58.24/77.13
67.18/24.13/58.25/77.12
67.25/24.14/58.23/77.07
67.23/24.14/58.22/77.12
67.20/24.13/58.24/77.13
67.21/24.14/58.26/77.13
67.24/24.13/58.25/77.13
67.17/24.14/58.24/77.13
67.22/24.13/58.25/77.13
67.19/24.14/58.25/77.13
67.16/24.13/58.23/77.13
67.24/24.14/58.24/77.13
67.23/24.12/58.25/77.08
67.18/24.13/58.25/77.13
67.18/24.13/58.26/77.09
67.23/24.10/58.12/77.13
67.18/24.13/58.25/77.13
67.18/24.14/58.09/77.13
67.18/24.24/58.25/77.13
67.21/24.13/58.24/77.13
67.21/24.14/58.24/77.13
67.20/24.13/58.25/77.13
67.18/24.13/58.23/77.13
  
```

On peut voir que l'arduino envoie bien les données sur l'interface de communication

USB

Dudzinski Théo

Test reception USB sur linux :



```
Ouverture du port reussie.  
45.24/40.93/60.86/51.75
```



```
Ouverture du port reussie.  
62.05/31.41/52.76/78.49
```

On peut voir que linux réceptionne bien les données envoyées via le port de communications USB.

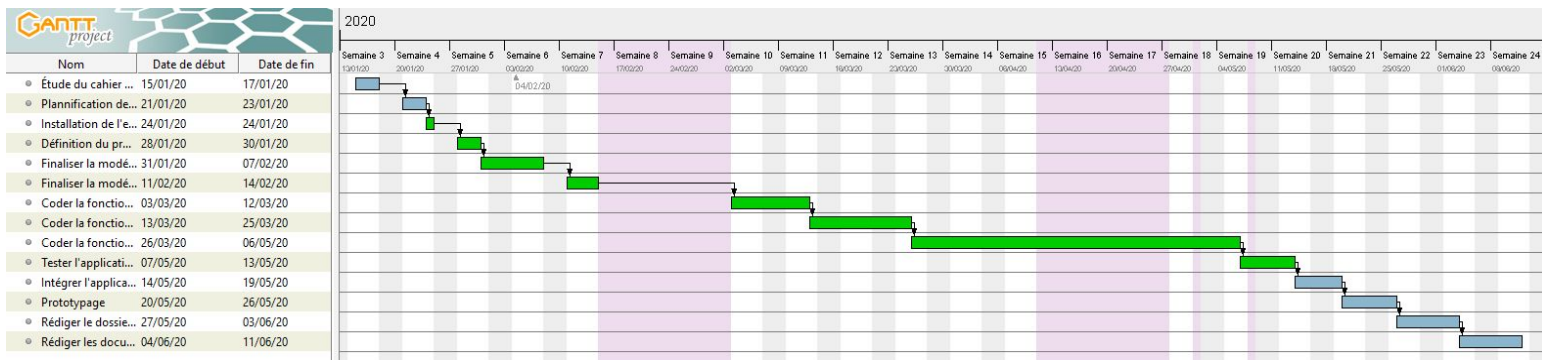
Dudzinski Théo

Partie

IHM Technicien

Dufour Jordan

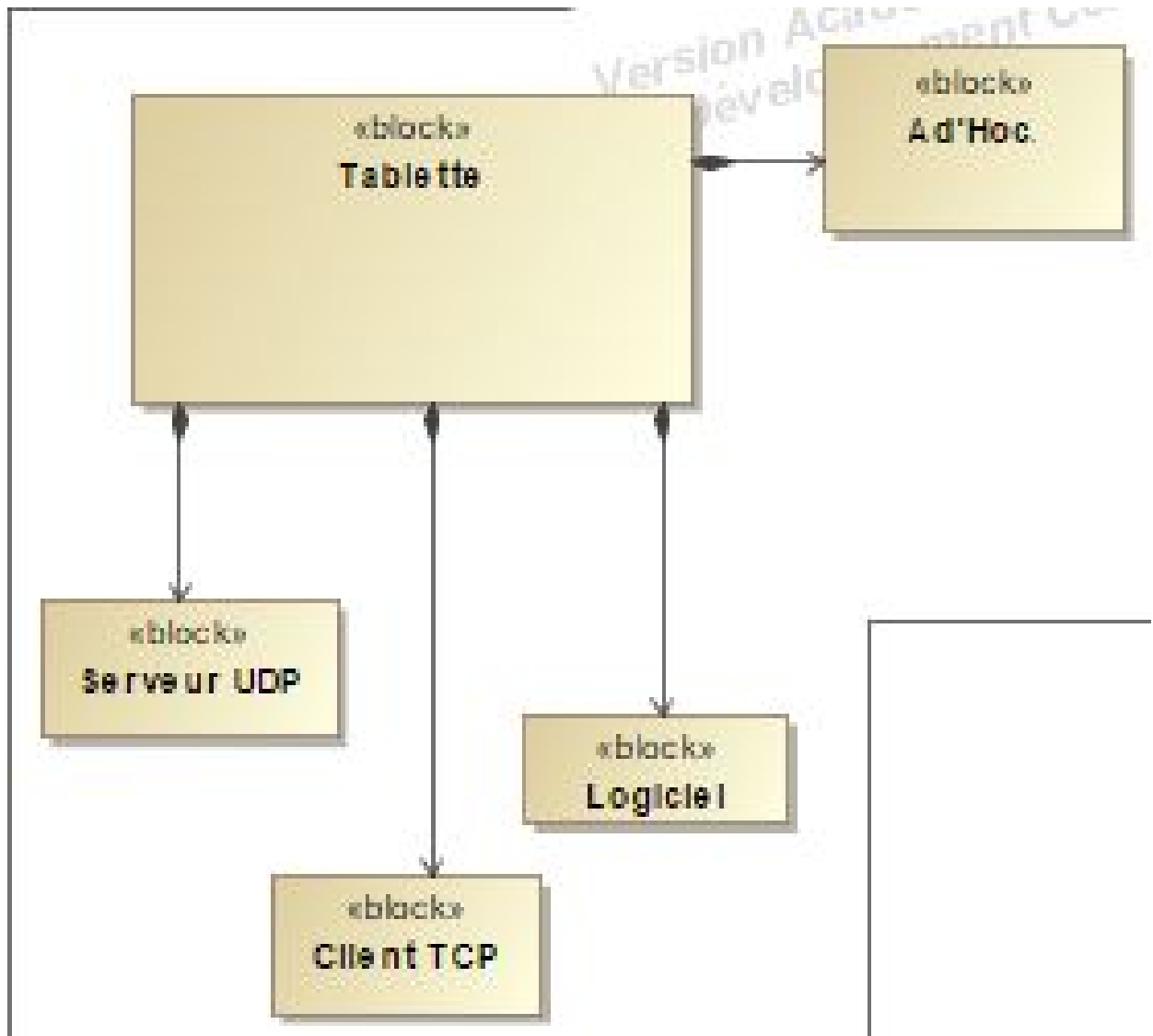
Planning prévisionnel :



Notre projet débute en début Janvier avec l'étude du cahier des charges et celui-ci finit en début Juin avec la rédaction du dossier technique. Notre projet devrait durer 5 mois. Ce planning comprend l'étude, le choix des matériels, la programmation et le prototypage et la rédaction du dossier technique.

Dufour Jordan

Diagramme de définition de bloc :

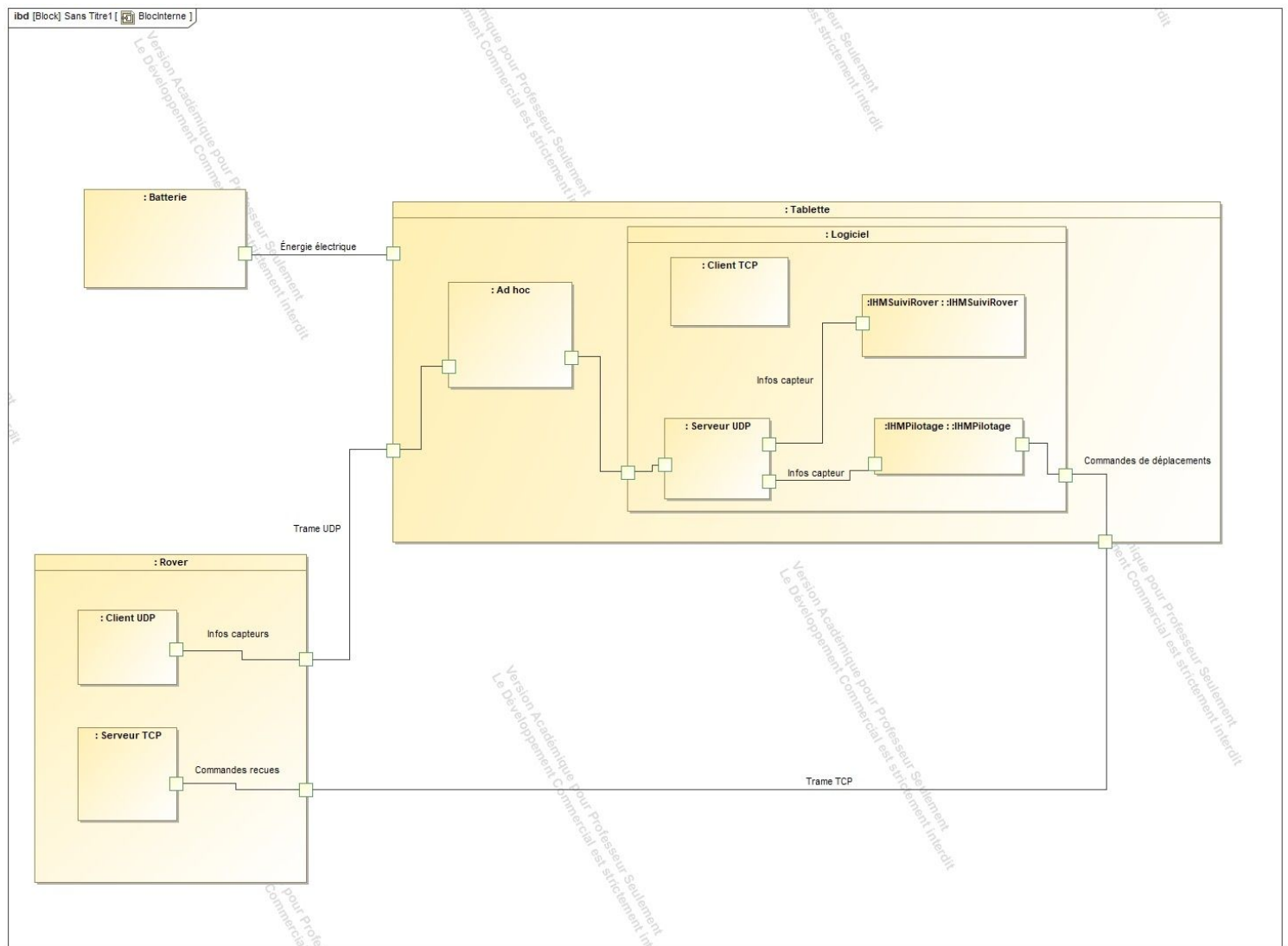


Tablette :

La tablette est composée d'un système Ad Hoc, d'un serveur UDP, d'un client TCP et du logiciel pour contrôler le rover.

Dufour Jordan

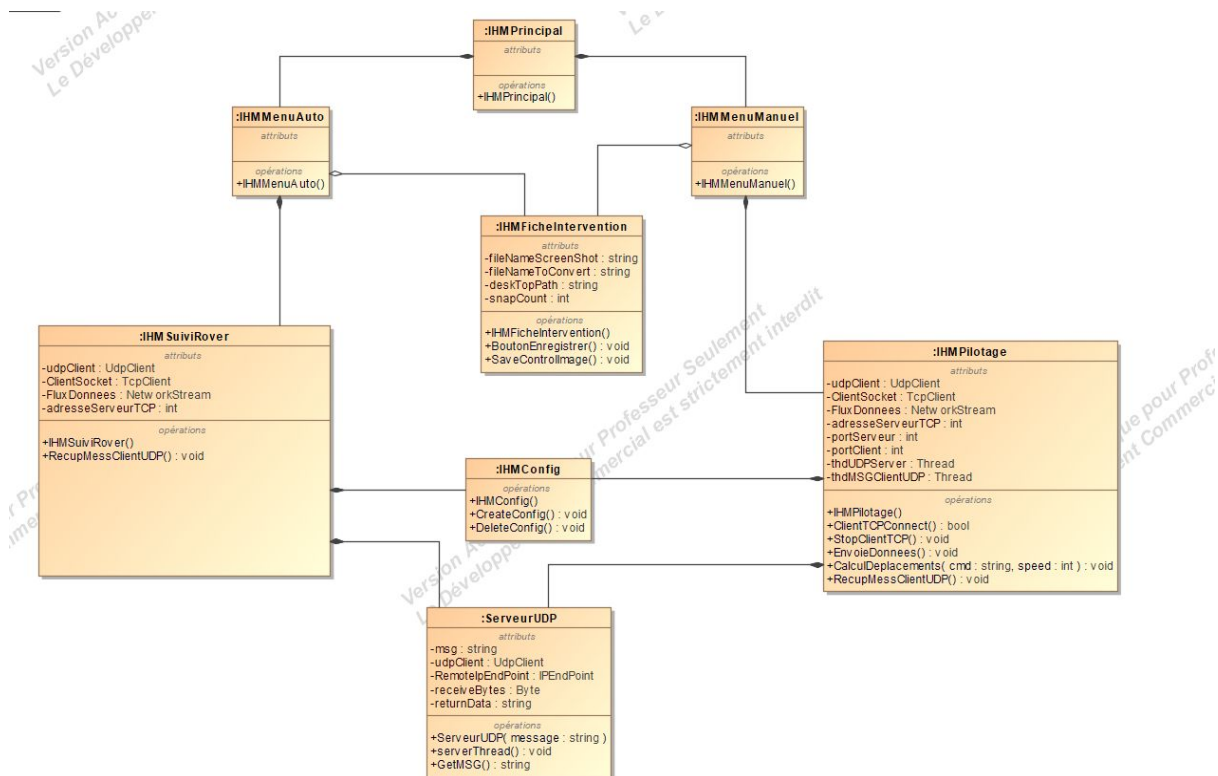
Diagramme de bloc interne :



La tablette étant sur batterie, elle récupère en permanence des trames UDP via le mode Ad Hoc par le rover, qui sont les informations des capteurs. Elle est aussi équipée du logiciel permettant de choisir le mode de contrôle, que ça soit manuel ou automatique et il permet aussi de contrôler le rover tout en ayant les informations perçues par les capteurs.

Dufour Jordan

Diagramme de classes :



L'IHM principal permet de choisir le mode de contrôle, soit manuel, soit automatique.

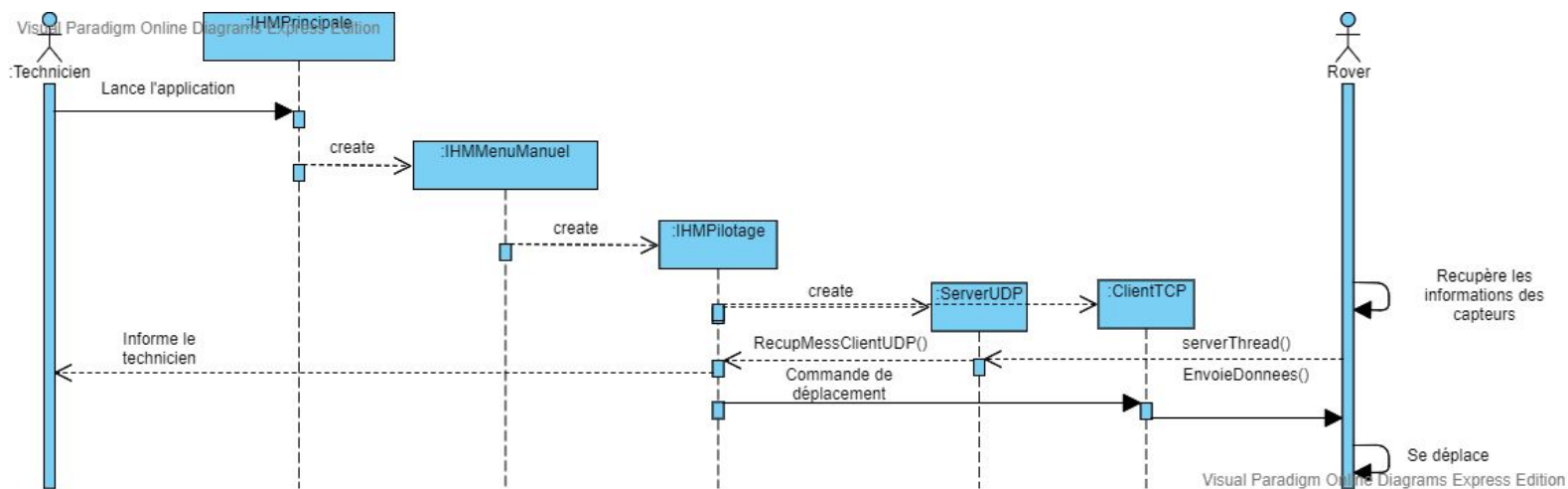
L'IHM du menu automatique permet d'accéder ensuite au suivi du rover sans le contrôler et donne aussi l'accès à la partie pour rédiger un compte rendu.

L'IHM du menu manuel permet d'accéder ensuite au pilotage du rover afin de pouvoir contrôler ce dernier à distance et permet aussi de rédiger à ça guise un compte rendu.

Les IHM de suivi du rover et du pilotage du rover permettent, quand à eux, de pouvoir configurer l'application selon les paramètres du rover (adresse IP, port de communication, etc...)

Dufour Jordan

Diagramme de séquence du cas d'utilisation “Piloter le Rover” :



Lorsque l'utilisateur lance l'application, il choisi le mode manuel et ensuite il choisi de piloter le rover.

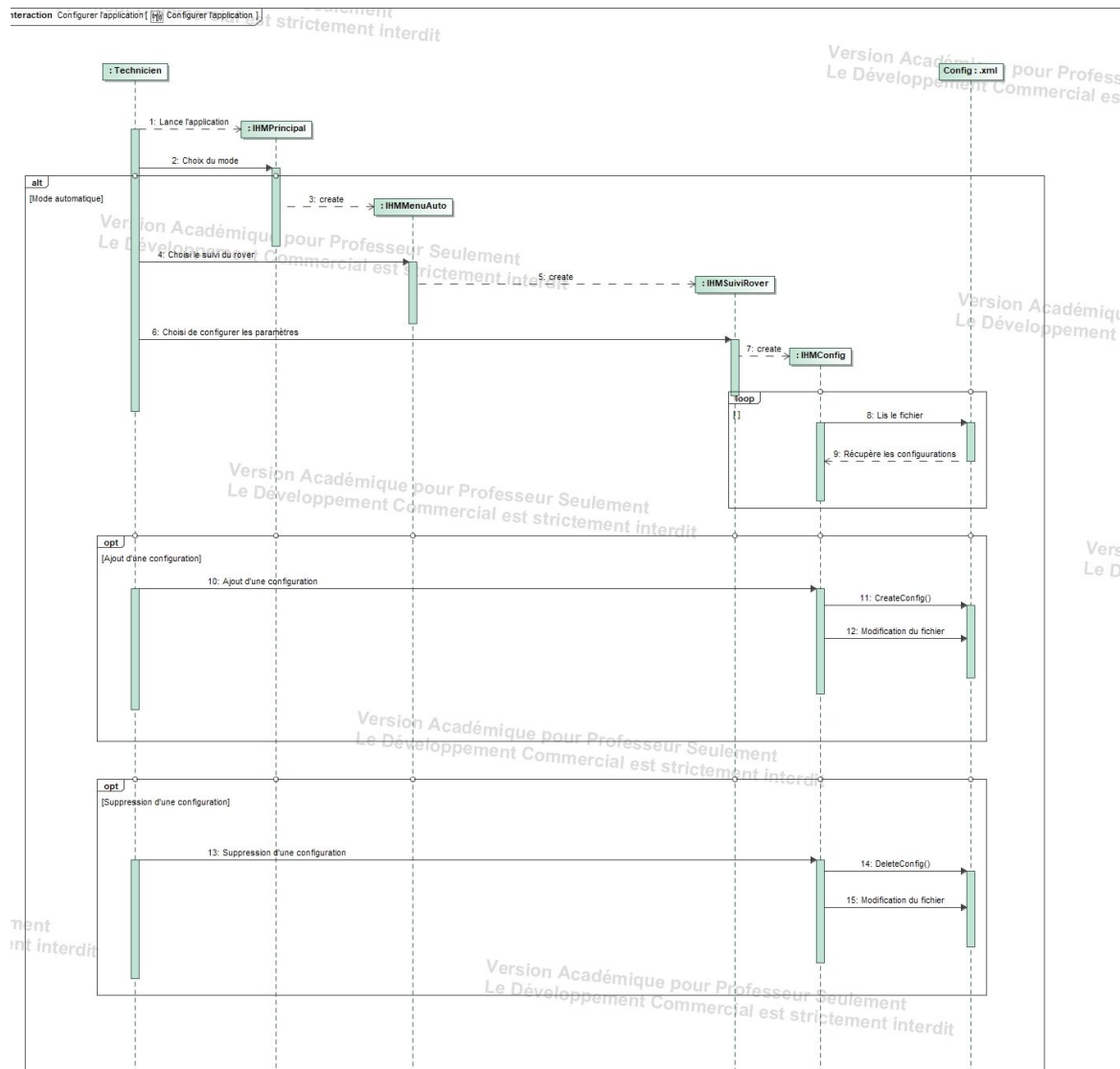
Une fois sur la fenêtre du pilotage (IHMPilotage), automatiquement en arrière plan, un serveur UDP est créé et reste allumé tant que l'utilisateur reste sur la fenêtre, de ce fait, ce serveur lui, permettra de recevoir les informations des capteurs en permanence que le rover lui envoi.

De même que le serveur UDP est créé, un client TCP est aussi créé en parallèle pour pouvoir envoyer des données liés aux commandes de déplacements lorsque l'utilisateur appuis sur un des boutons de déplacements.

Dufour Jordan

Diagramme de séquence du cas d'utilisation

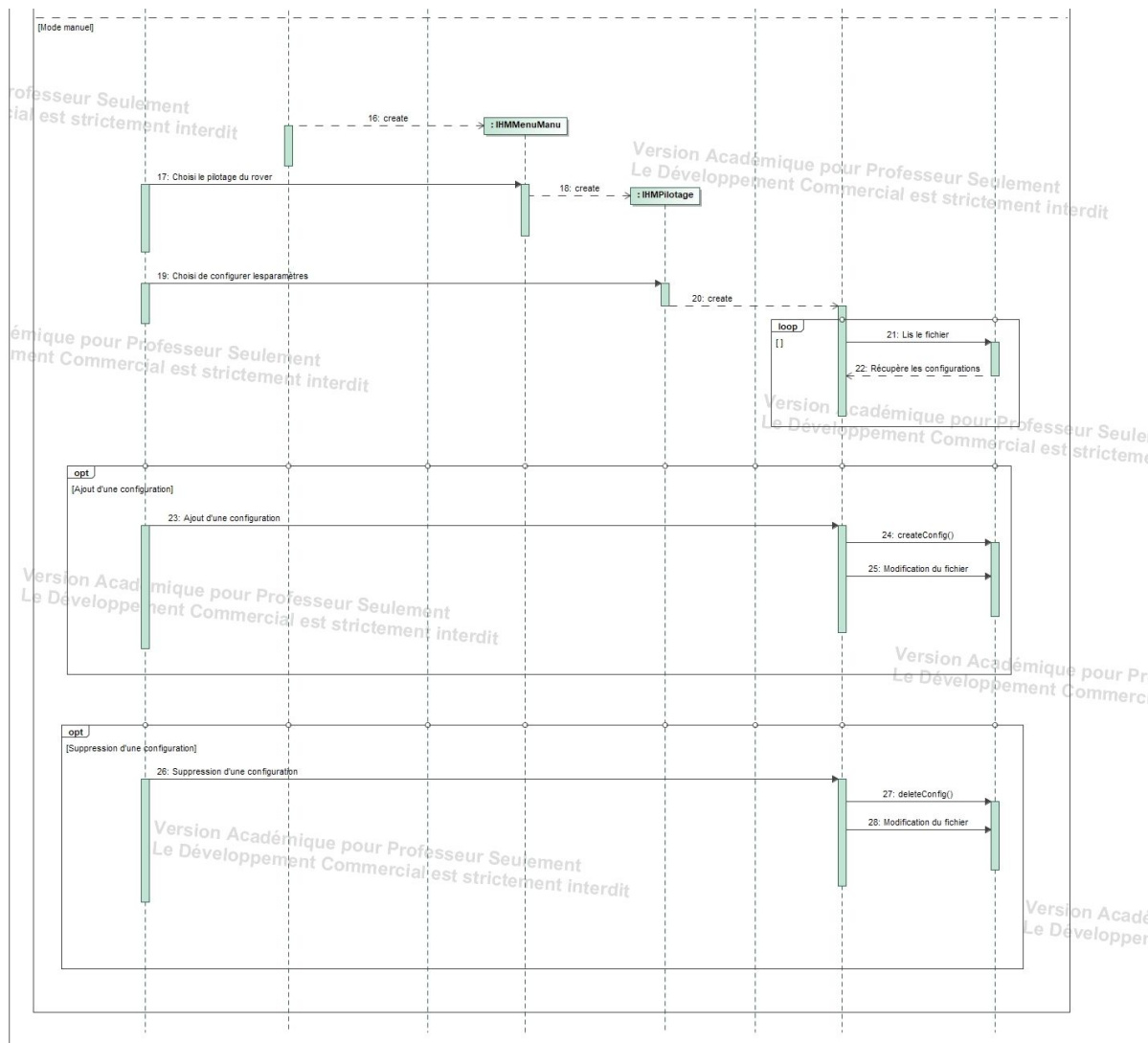
“Configurer l’application” :



Lorsque l'utilisateur lance l'application, il choisi le mode automatique et ensuite il choisi de suivre le rover qui se déplacera sans l'intervention du technicien. A partir de la fenêtre de suivi du rover, le technicien peut alors modifier les paramètres réseaux, de choisir une des configurations déjà existantes ou d'en créer une.

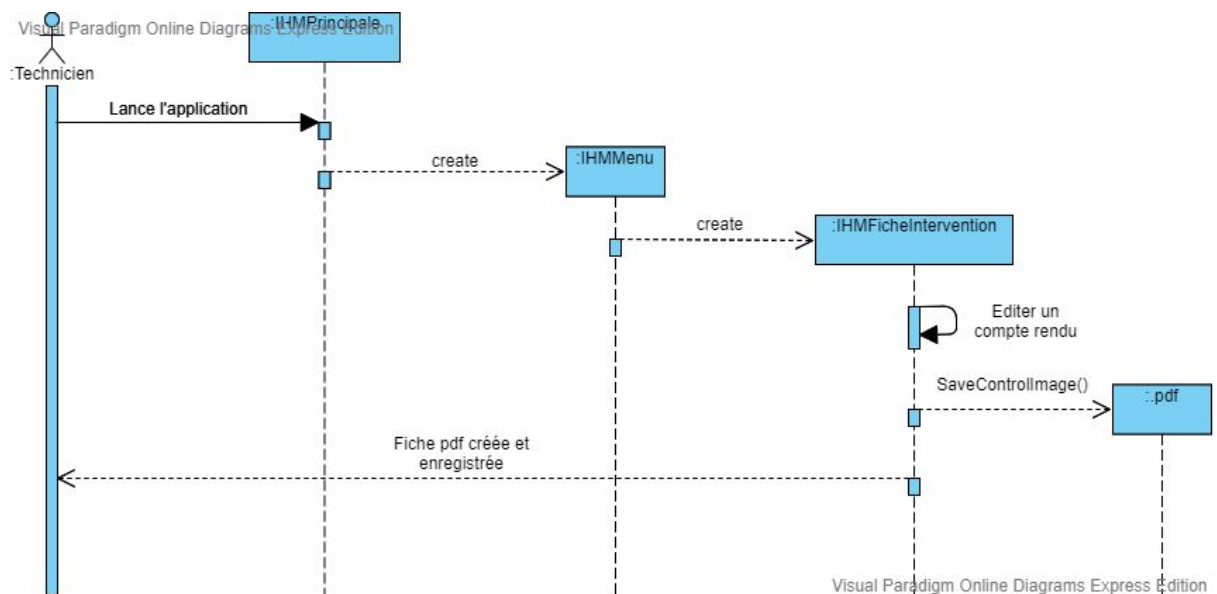
S'il veut créer une configuration, le technicien doit aller sur "Config...", dans l'onglet "Ajouter" et entrez les informations dans les champs respectives et appuyer sur le bouton **Ajouter**

Dufour Jordan



Dufour Jordan

Diagramme de séquence du cas d'utilisation “Editer un compte rendu” :



Dufour Jordan

Langage de codage :



Le langage qui est utilisé pour créer l'application est le C# (C sharp) .NET, plus précisément une application Windows Forms qui sera faite à l'aide de Visual Studio Community 2019.

Matériel mise à disposition :

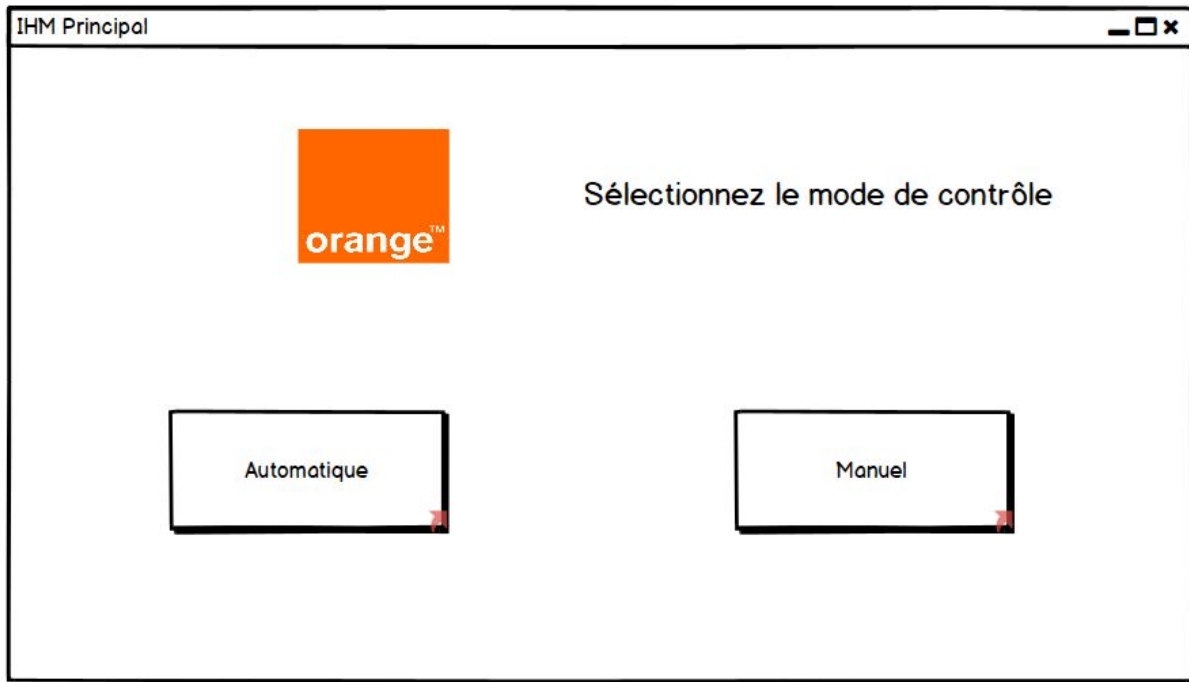


Une tablette Surface Pro 2, fonctionnant sous Windows 10 à été mise à disposition afin d'avoir la capacité de contrôler le rover à distance.

Au démarrage de celle-ci, il y a mise en route du mode ad hoc afin de pouvoir se connecter au rover et communiquer avec celui-ci.

Dufour Jordan

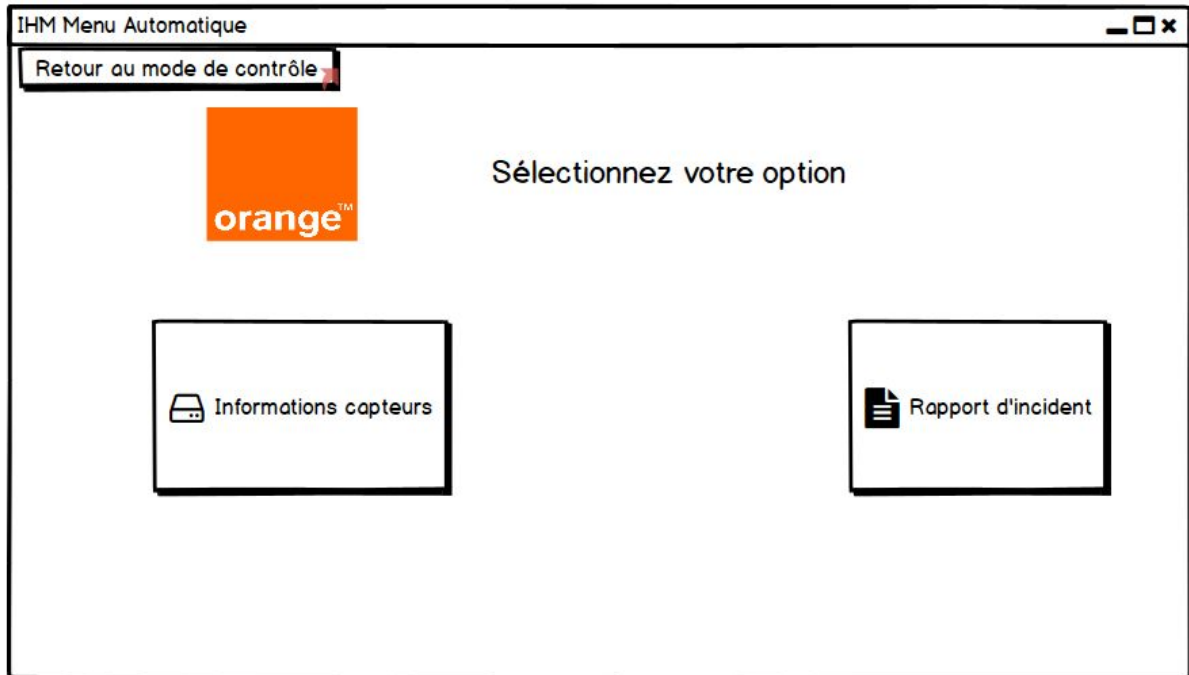
La maquette de l'IHM Principale :



L'IHM Principale est la première fenêtre que l'utilisateur verra lorsqu'il démarrera l'application.

Elle permet de choisir le mode de contrôle, soit entre le mode automatique ou le mode manuel.

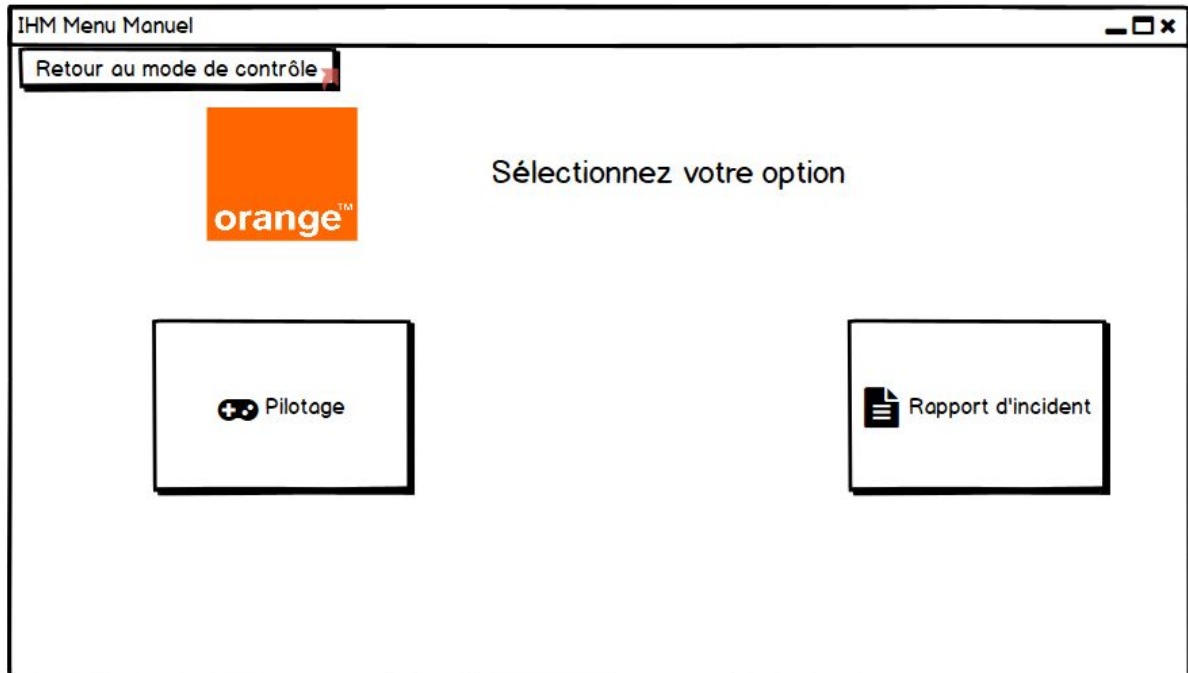
La maquette de l'IHM Menu Automatique :



L'IHM Menu Automatique permet le choix entre l'édition d'un compte rendu ou bien encore suivre les déplacements du rover qui, ce dernier se déplacera de façon autonome.

Dufour Jordan

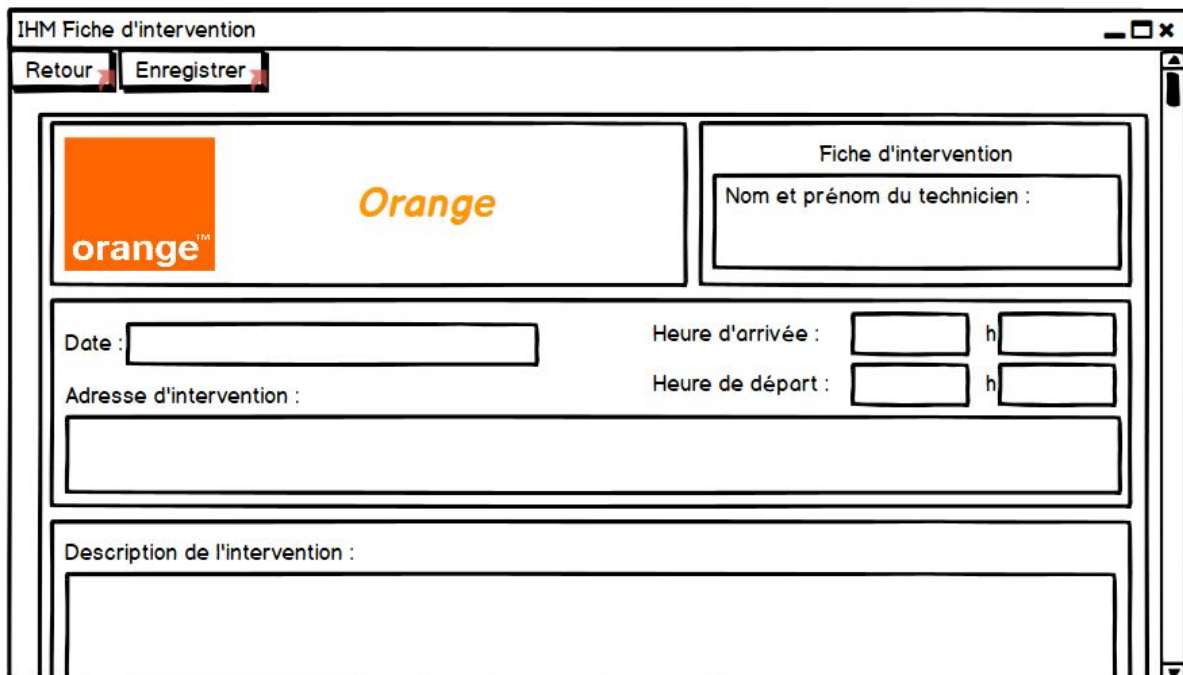
La maquette de l'IHM Menu Manuel :



L'IHM Menu Manuel permet le choix entre l'édition d'un compte rendu ou bien encore contrôler manuellement les déplacements du rover tout en ayant les informations des capteurs sous l'oeil.

Dufour Jordan

La maquette de l'IHM Fiche d'intervention :

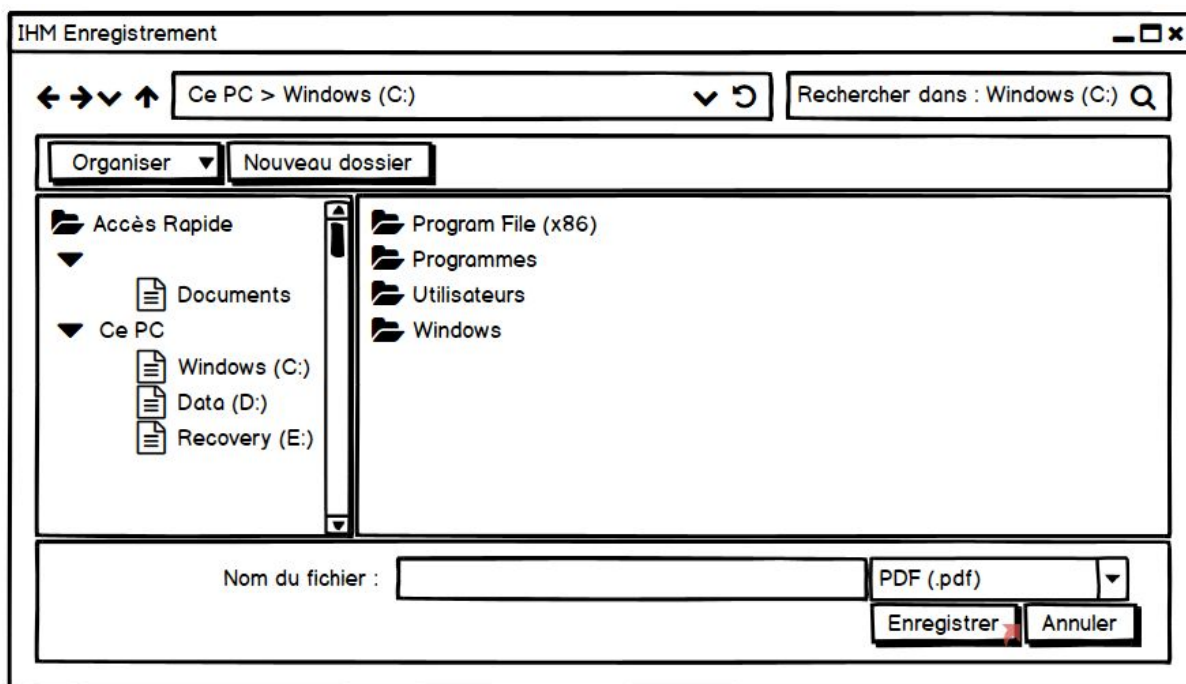


The image shows a wireframe of a web application window titled "IHM Fiche d'intervention". At the top left, there are two buttons: "Retour" and "Enregistrer". The main content area is divided into several sections. On the left, there is a large box containing the Orange logo (an orange square with "orange" text) and the word "Orange" in a large, stylized font. To the right of this, there is a section titled "Fiche d'intervention" with a sub-label "Nom et prénom du technicien :" followed by a text input field. Below these, there are two rows of form fields. The first row has "Date :" followed by a date input field, and "Heure d'arrivée :" followed by two input fields for hours and minutes. The second row has "Adresse d'intervention :" followed by a large text area, and "Heure de départ :" followed by two input fields for hours and minutes. At the bottom, there is a section titled "Description de l'intervention :" followed by a large text area. The window has standard OS controls (minimize, maximize, close) in the top right corner.

L'IHM Fiche d'intervention permet l'édition d'un compte rendu, comprenant tous les détails, voire les plus importants lors de l'intervention du technicien, il y permet aussi d'y mentionner la date et l'adresse de l'intervention tout en sachant quel technicien s'y occupait.

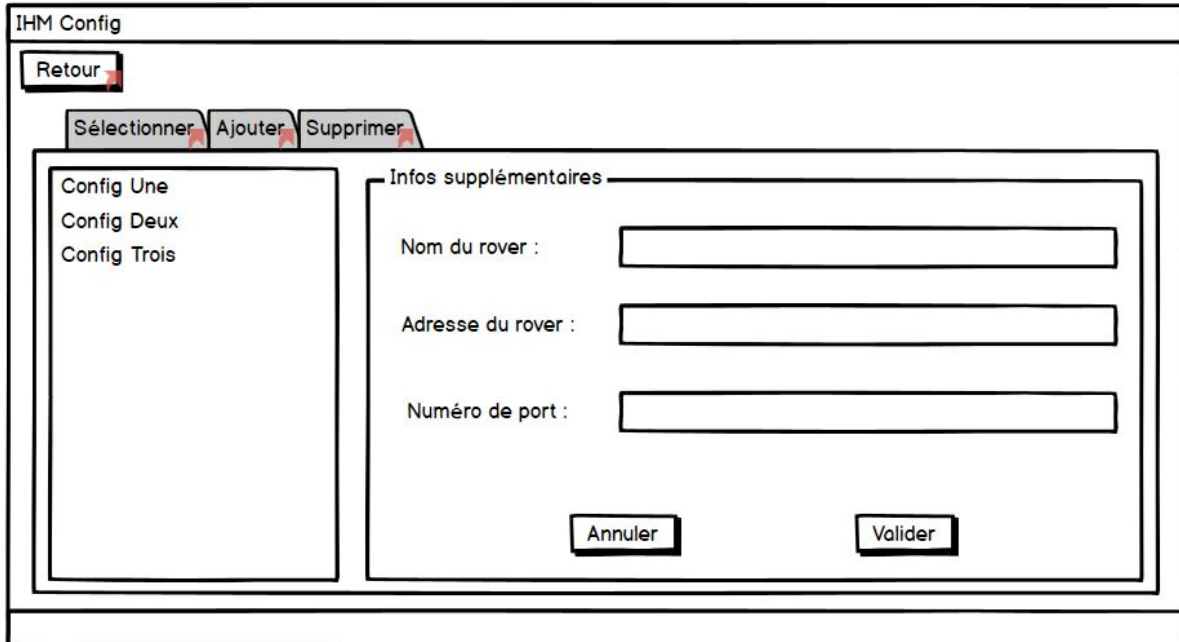
De ce fait, une fois la fiche remplie, le technicien peut l'enregistrer sous format pdf et d'y avoir un aperçu de ce pdf pour y vérifier si toutes les informations mentionnées sont présentes. Cet enregistrement ouvre une fenêtre d'enregistrement (voir page 52).

La fenêtre d'enregistrement :



Cette fenêtre permet d'enregistrer la fiche en cours de complétion là où l'on veut qu'elle soit, que cela soit directement sur le bureau de la tablette, ou dans un dossier, voire même dans une clé USB.

La maquette de l'IHM Config (section Sélectionner) :



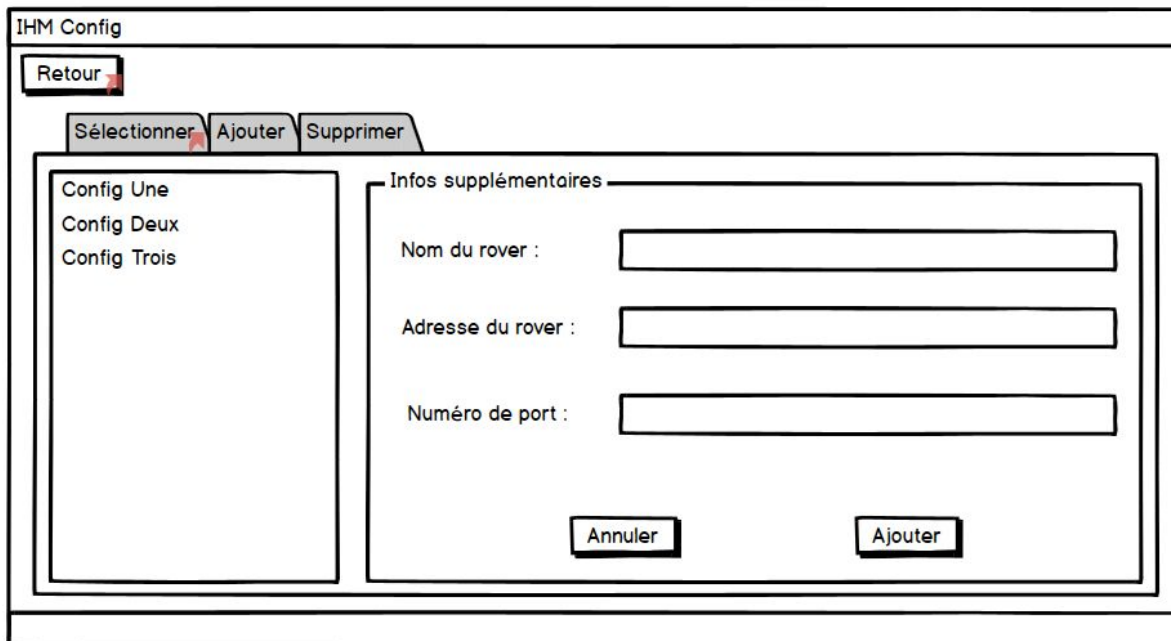
The image shows a wireframe of a software interface titled "IHM Config". At the top left is a "Retour" button with a red arrow. Below it are three buttons: "Sélectionner", "Ajouter", and "Supprimer", each with a red arrow. The main area is divided into two panels. The left panel contains a list of three items: "Config Une", "Config Deux", and "Config Trois". The right panel is titled "Infos supplémentaires" and contains three input fields labeled "Nom du rover :", "Adresse du rover :", and "Numéro de port :". At the bottom of the right panel are two buttons: "Annuler" and "Valider".

Cette fenêtre permet d'y sélectionner une configuration de rover déjà existante.

Veuillez noter que toutes les configurations qui sont déjà présentes dans un fichier des configurations seront affichés dans une liste.

Dufour Jordan

La maquette de l'IHM Config (section Ajouter) :



Maquette de l'IHM Config (section Ajouter) :

Le formulaire est divisé en deux sections principales :

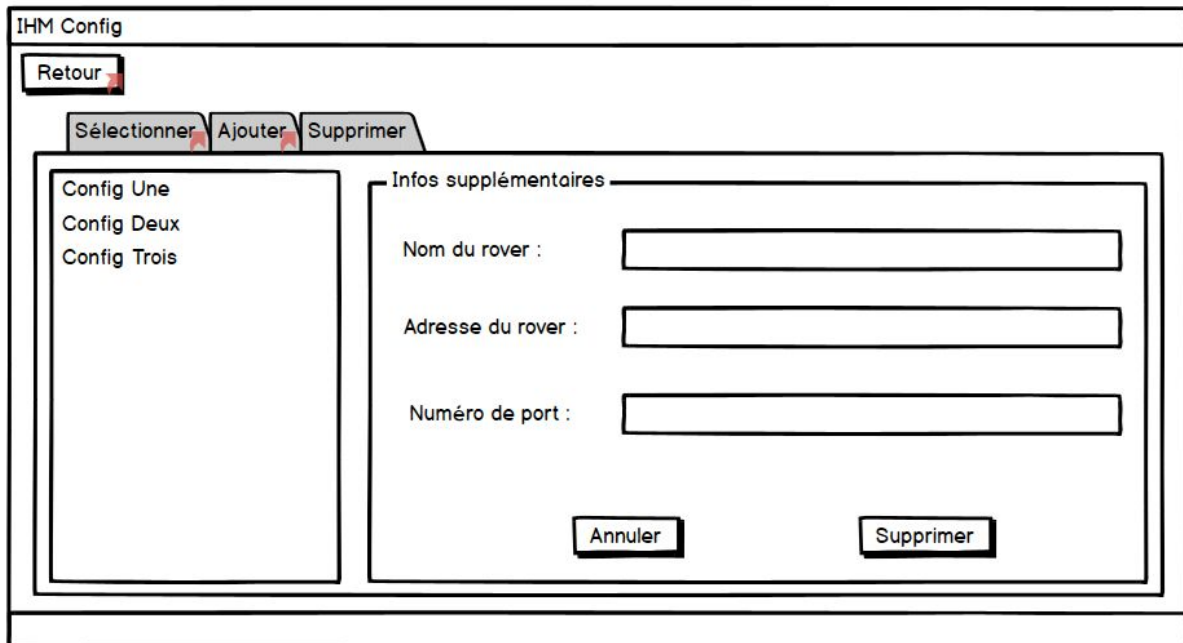
- Section gauche (Liste des configurations) :**
 - Un bouton "Retour" en haut à gauche.
 - Une barre de navigation avec trois onglets : "Sélectionner", "Ajouter" (actif), et "Supprimer".
 - Une liste de configurations : "Config Une", "Config Deux", et "Config Trois".
- Section droite (Infos supplémentaires) :**
 - Un titre "Infos supplémentaires" suivi d'une ligne de séparation.
 - Deux champs de saisie : "Nom du rover :" et "Adresse du rover :".
 - Un champ de saisie : "Numéro de port :".
 - Deux boutons : "Annuler" et "Ajouter".

Cette fenêtre permet d'y ajouter une configuration de rover que l'utilisateur souhaite.

Veuillez noter que toutes les configurations qui sont ajoutés seront affichés avec celles déjà présentes dans la liste des configurations à sélectionner.

Dufour Jordan

La maquette de l'IHM Config (section Supprimer) :



Maquette de l'IHM Config (section Supprimer) :

Le formulaire est intitulé "IHM Config".

Il contient une barre de navigation avec les boutons : Retour, Sélectionner, Ajouter, Supprimer.

La section "Infos supplémentaires" est divisée en deux parties :

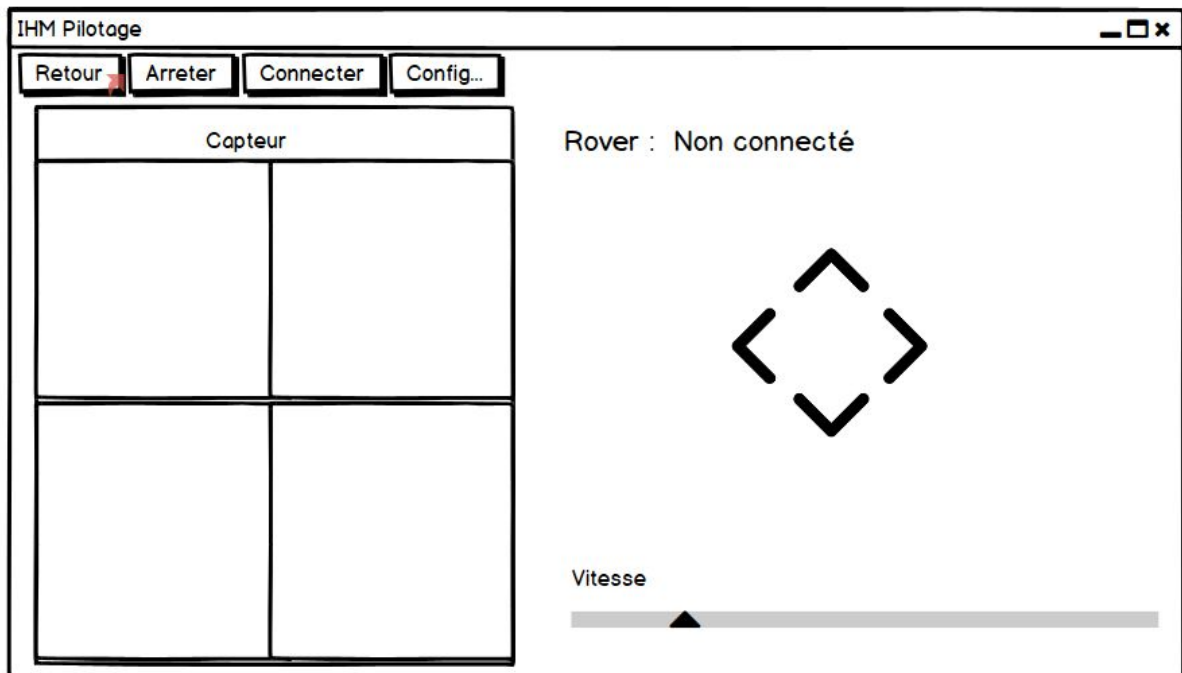
- À gauche, une liste de configurations à sélectionner : Config Une, Config Deux, Config Trois.
- À droite, des champs de saisie pour les informations supplémentaires : Nom du rover, Adresse du rover, Numéro de port.

En bas à droite, il y a deux boutons : Annuler et Supprimer.

Cette fenêtre permet d'y supprimer une configuration de rover que l'utilisateur souhaite.

Veuillez noter que toutes les configurations qui sont supprimés ne seront plus affichés avec celles déjà présentes dans la liste des configurations à sélectionner.

La maquette de l'IHM Pilotage :



Cette fenêtre permet d'y piloter le rover comme l'utilisateur souhaite.

Veuillez noter que l'utilisateur doit sélectionner une configuration avant de se connecter au rover via le bouton **Connecter**.

Cette fenêtre permet aussi d'accéder aux configurations via le bouton **Config...**

Dufour Jordan

Fonction de connexion au Rover grâce à la tablette :

```
1 référence
private bool ClientTCPConnect()
{
    bool connected = false;
    try
    {
        ClientSocket = new TcpClient();
        connected = true;
        ClientSocket.Connect(adresseServeurTCP, portClient);

        return connected;
    }
    catch (Exception e)
    {
        connected = false;
        return connected;
    }
}
```

Cette fonction permet au technicien de se connecter au rover en appuyant sur le bouton **Connecter**, soit dans la fenêtre de suivi du rover en mode automatique, soit avant de vouloir piloter le rover en mode manuel.

Cette fonction crée donc un client qui se connectera au serveur, c'est-à-dire le rover. Si l'opération réussie, alors on le fait savoir au technicien via une barre de statut qui lui indiquera qu'il est bel et bien connecté au rover.

Dans le cas échéant, la barre de statut lui indiquera que le client n'a pas réussi à se connecter au rover. Dans ce cas là, il faut revoir les paramètres sélectionnés dans **Config...**

Dufour Jordan

Fonction de récupération des informations reçues par les capteurs :

```
public class ServerUDP
{
    string msg;
    1 référence
    public ServerUDP(string messageR)
    {
        msg = messageR;
    }
    1 référence
    public void serverThread()
    {
        try {
            UdpClient udpClient = new UdpClient(53000);

            while (true)
            {
                IPEndPoint RemoteIpEndPoint = new IPEndPoint(IPAddress.Any, 0);

                Byte[] receiveBytes = udpClient.Receive(ref RemoteIpEndPoint);

                string returnData = Encoding.ASCII.GetString(receiveBytes);

                msg = returnData.ToString();
                //MessageBox.Show(msg);
            }
        } catch (Exception e)
        { }
    }

    1 référence
    public string GetMSG
    {
        get { return msg; }
        set { msg = value; }
    }
}
```

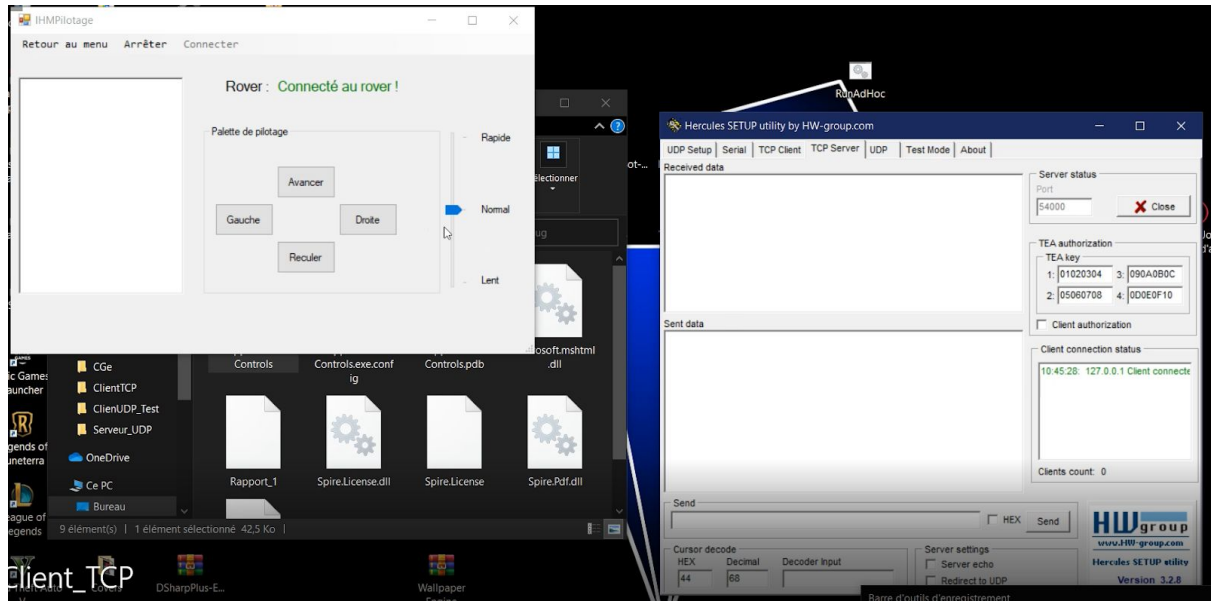
Cette fonction permet au technicien de recevoir et percevoir les informations que les capteurs reçoivent.

On crée d'abord un point de fin afin de créer le serveur et on y lis toutes les informations qu'il va recevoir et on les convertit en ASCII afin de les affichés sur l'application.

Dufour Jordan

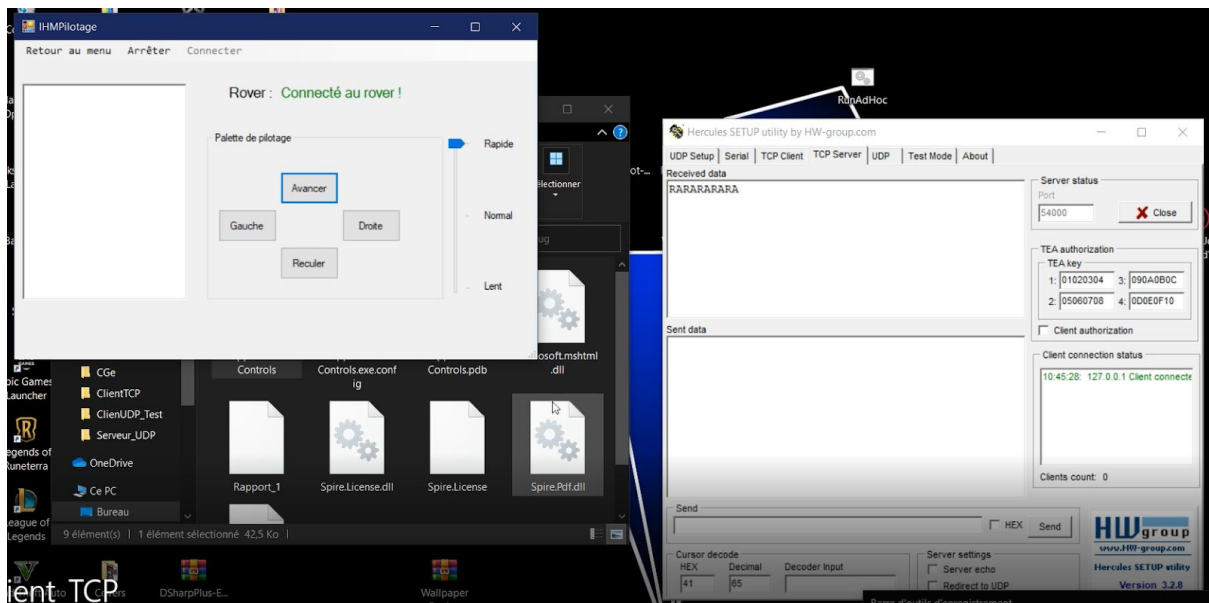
Test unitaire du client TCP (en local) :

Pour une meilleure visualisation et compréhension de ce test, il est conseillé de lire la vidéo fournie avec ce dossier du nom de "Client_TCP.mp4" dans le répertoire "annexes"



Pour ce test, nous utilisons un logiciel permettant de simuler un serveur TCP (Hercules) afin que l'on puisse tenter une connexion TCP via la tablette.

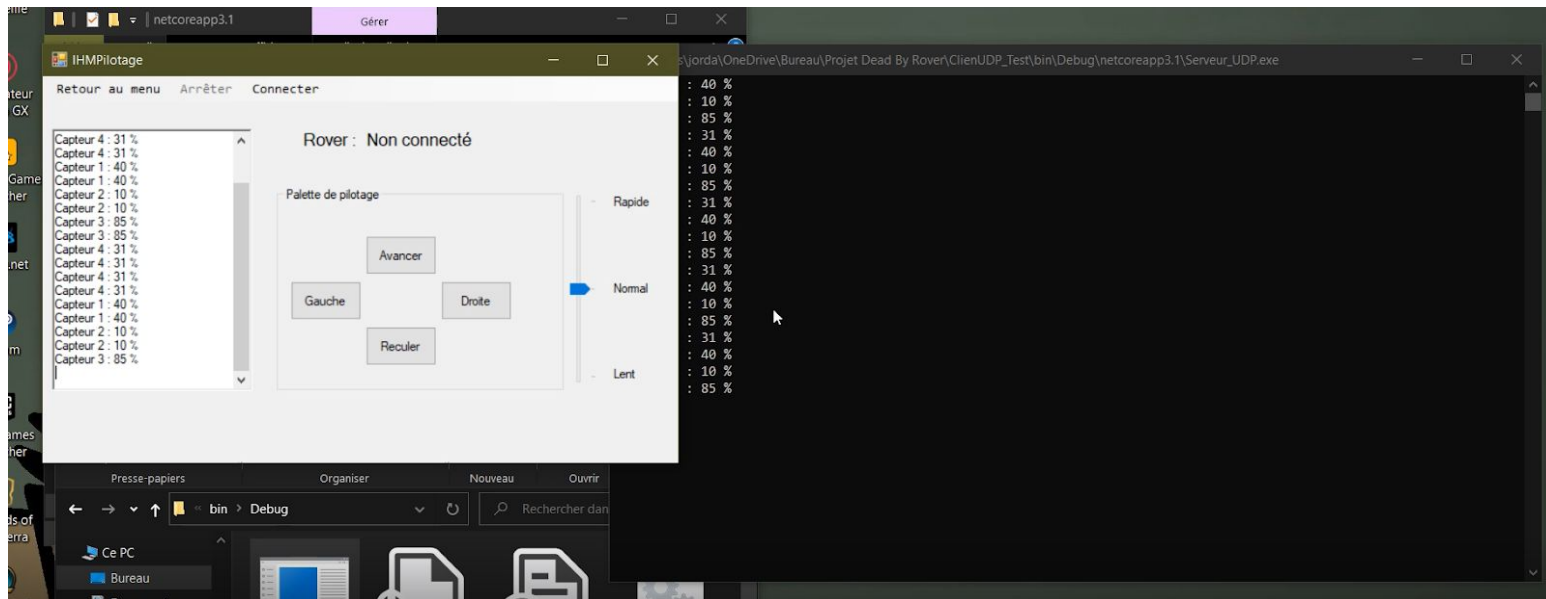
Comme vous pouvez le voir, la connexion est réussie et l'on essaie d'envoyer des commandes de déplacement (image ci-dessous)



Dufour Jordan

Test unitaire du serveur UDP (en local) :

Pour une meilleure visualisation et compréhension de ce test, il est conseillé de lire la vidéo fournie avec ce dossier du nom de “Serveur_UDP.mp4” dans le répertoire “annexes”



Ici pour simuler un client UDP qui envoie des informations en permanence, nous avons conçu une petite application console qui envoie ces informations au port du serveur de la tablette.

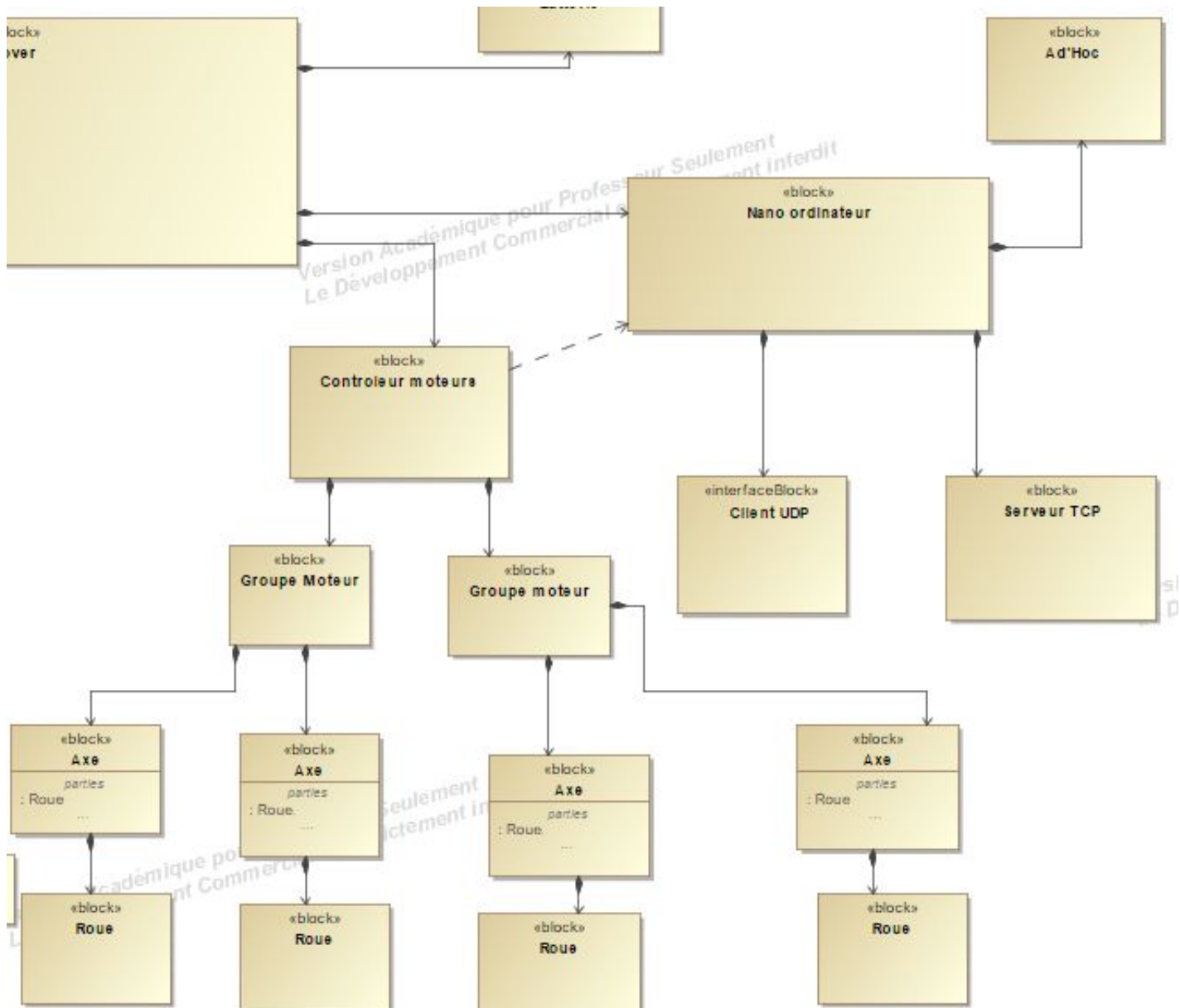
Comme nous pouvons le constater ici, la tablette récupère bel et bien les informations envoyés par le client UDP.

Dufour Jordan

Partie Déplacements

Hadoux Benjamin

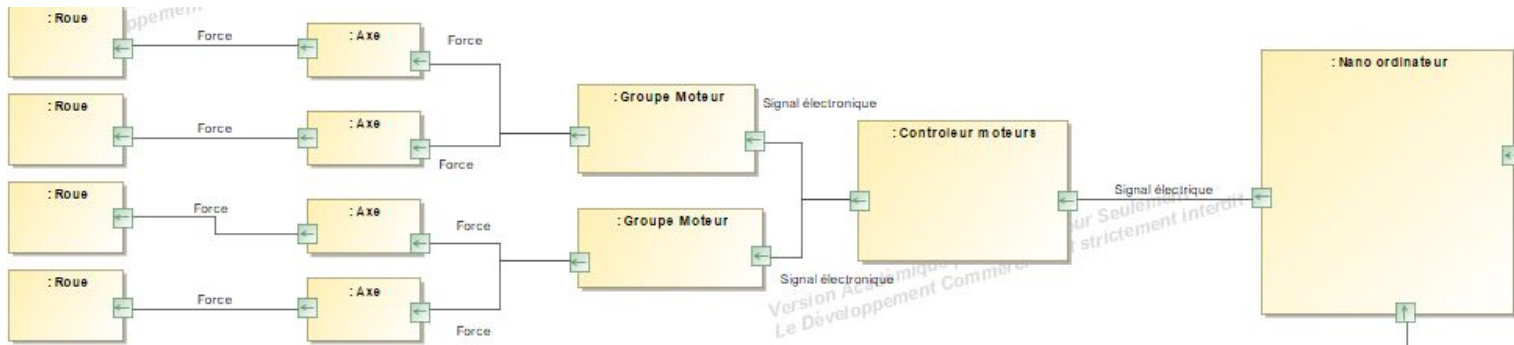
Diagramme de définition de bloc :



Le contrôleur moteur est composé des deux groupe moteur qui permette de contrôler les axes des roues, le nano ordinateur est composé d'un client UDP pour l'envoyer à la tablette

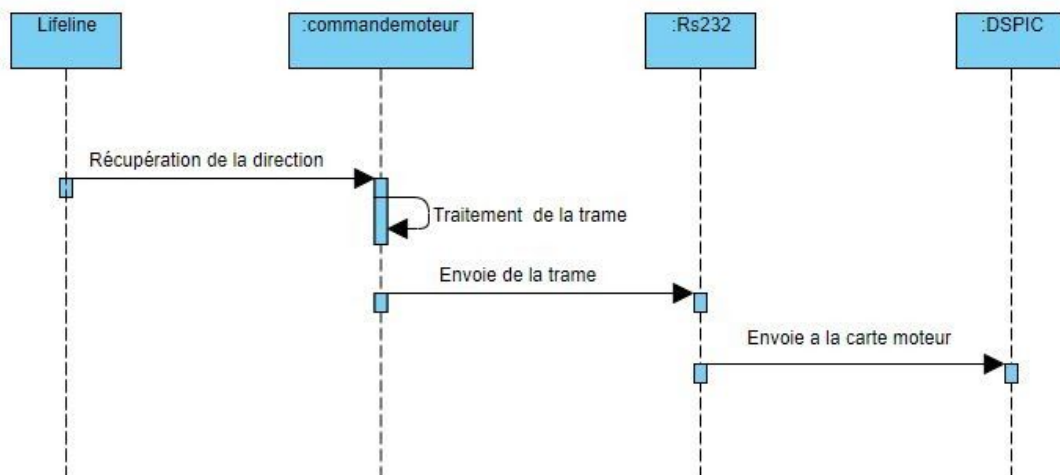
Hadoux Benjamin

Diagramme de définition de bloc interne :



Le nano ordinateur permet l'envoi des trames au contrôleur moteur qui permet de traduire la trame pour les groupes moteurs

Diagramme de séquence :



Je récupère la direction inscrit dans le mutex pour ensuite la convertir en trame que j'envoie via la liaison RS232 à la carte moteur pour effectuer le mouvement.

Hadoux Benjamin

Conception détaillée : Etude du protocole

Manquant d'informations dans la documentation technique du système, j'ai via Wireshark sniffer une trame envoyée au rover afin d'en comprendre le protocole de communication permettant de contrôler le Rover.

*Ethernet

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

ip.addr==192.168.1.106 && tcp.dstport==15020

No.	Time	Source	Destination	Protocol	Length	Info
28	27.252602	192.168.1.1	239.255.255.250	SSDP	354	NOTIFY * HTTP/1.1
29	27.356772	192.168.1.1	239.255.255.250	SSDP	386	NOTIFY * HTTP/1.1
30	27.460774	192.168.1.1	239.255.255.250	SSDP	315	NOTIFY * HTTP/1.1
31	27.564604	192.168.1.1	239.255.255.250	SSDP	374	NOTIFY * HTTP/1.1
32	27.668734	192.168.1.1	239.255.255.250	SSDP	368	NOTIFY * HTTP/1.1
33	27.772787	192.168.1.1	239.255.255.250	SSDP	315	NOTIFY * HTTP/1.1
34	27.876611	192.168.1.1	239.255.255.250	SSDP	370	NOTIFY * HTTP/1.1
35	27.980722	192.168.1.1	239.255.255.250	SSDP	380	NOTIFY * HTTP/1.1
36	29.147812	192.168.1.155	192.168.1.106	TCP	63	49873 → 15020 [PSH, ACK] Seq=1 Ack=1 Win=65536 Len=9
37	29.149391	192.168.1.106	192.168.1.155	TCP	60	15020 → 49873 [ACK] Seq=1 Ack=10 Win=29312 Len=0
38	29.150158	192.168.1.106	192.168.1.155	TCP	75	15020 → 49873 [PSH, ACK] Seq=1 Ack=10 Win=29312 Len=21
39	29.190381	192.168.1.155	192.168.1.106	TCP	54	49873 → 15020 [ACK] Seq=10 Ack=22 Win=65536 Len=0
40	31.137395	192.168.1.106	192.168.1.155	TCP	60	15020 → 49873 [FIN, ACK] Seq=22 Ack=10 Win=29312 Len=0
41	31.137480	192.168.1.155	192.168.1.106	TCP	54	49873 → 15020 [ACK] Seq=10 Ack=23 Win=65536 Len=0
42	31.181798	192.168.1.155	192.168.1.106	TCP	54	49873 → 15020 [FIN, ACK] Seq=10 Ack=23 Win=65536 Len=0
43	31.183025	192.168.1.106	192.168.1.155	TCP	60	15020 → 49873 [ACK] Seq=23 Ack=11 Win=29312 Len=0

> Frame 36: 63 bytes on wire (504 bits), 63 bytes captured (504 bits) on interface 0

> Ethernet II, Src: Giga-Byt_b5:04:55 (fc:aa:14:b5:04:55), Dst: Sparklan_4e:20:9c (00:0e:8e:4e:20:9c)

> Internet Protocol Version 4, Src: 192.168.1.155, Dst: 192.168.1.106

> Transmission Control Protocol, Src Port: 49873, Dst Port: 15020, Seq: 1, Ack: 1, Len: 9

▼ Data (9 bytes)

Data: ff07100110015b803d

[Length: 9]

```

0000  00 0e 8e 4e 20 9c fc aa 14 b5 04 55 08 00 45 00  ...N...U..E.
0010  00 31 59 36 40 00 80 06 1d 3b c0 a8 01 9b c0 a8  ..1Y6@...;....
0020  01 6a c2 d1 3a ac 14 cd bb 5c 06 c1 0b 20 50 18  .j... \...P.
0030  01 00 93 5a 00 00 ff 07 10 01 10 01 5b 80 3d    ...Z.... [=
  
```

Hadoux Benjamin

	<u>Lent</u>	<u>Normale</u>	<u>Rapide</u>
<u>Avancer</u>	LA	NA	RA
<u>Droit</u>	LA	ND	RD
<u>Gauche</u>	LG	NG	RG
<u>Reculer</u>	LR	NR	RR

J'ai conçu ce tableau pour faciliter la communication au sien du système

Tests unitaires:

Deplacements

Commande :

LR

ff 07 a0 00 a0 00 ab 4c 54

LA

ff 07 a0 00 a0 00 5b c0 6c

LD

ff 07 a0 00 96 00 5b 20 4a

LG

ff 07 96 00 a0 00 5b 08 40

Deplacements

Commande :

NR

ff 07 50 01 50 01 ab d8 30

NA

ff 07 50 01 50 01 5b 80 26

ND

ff 07 50 01 60 01 5b 47 48

NG

ff 07 60 01 50 01 5b 4f 4a

Deplacements

Commande :

RR

ff 07 10 01 10 01 ab 80 79

RA

ff 07 10 01 10 01 5b 80 3d

RD

ff 07 10 01 20 01 5b 98 7a

RG

ff 07 20 01 10 01 5b 50 7e

J'ai récupéré la commande défini par le tableau de commandes pour ensuite envoyer la trame correspondante

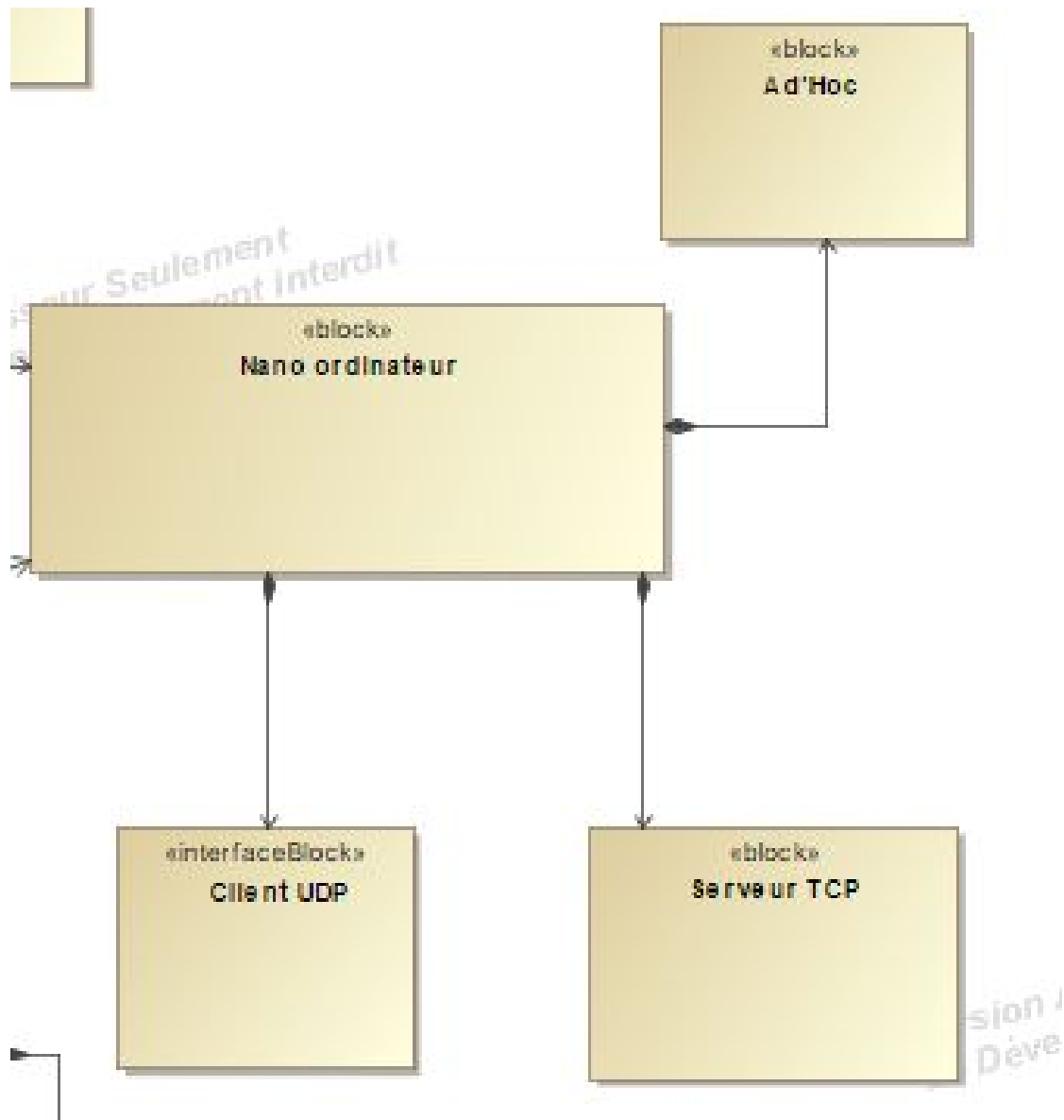
Hadoux Benjamin

Partie

Gestion du Rover

Gerzynski Gaëtan

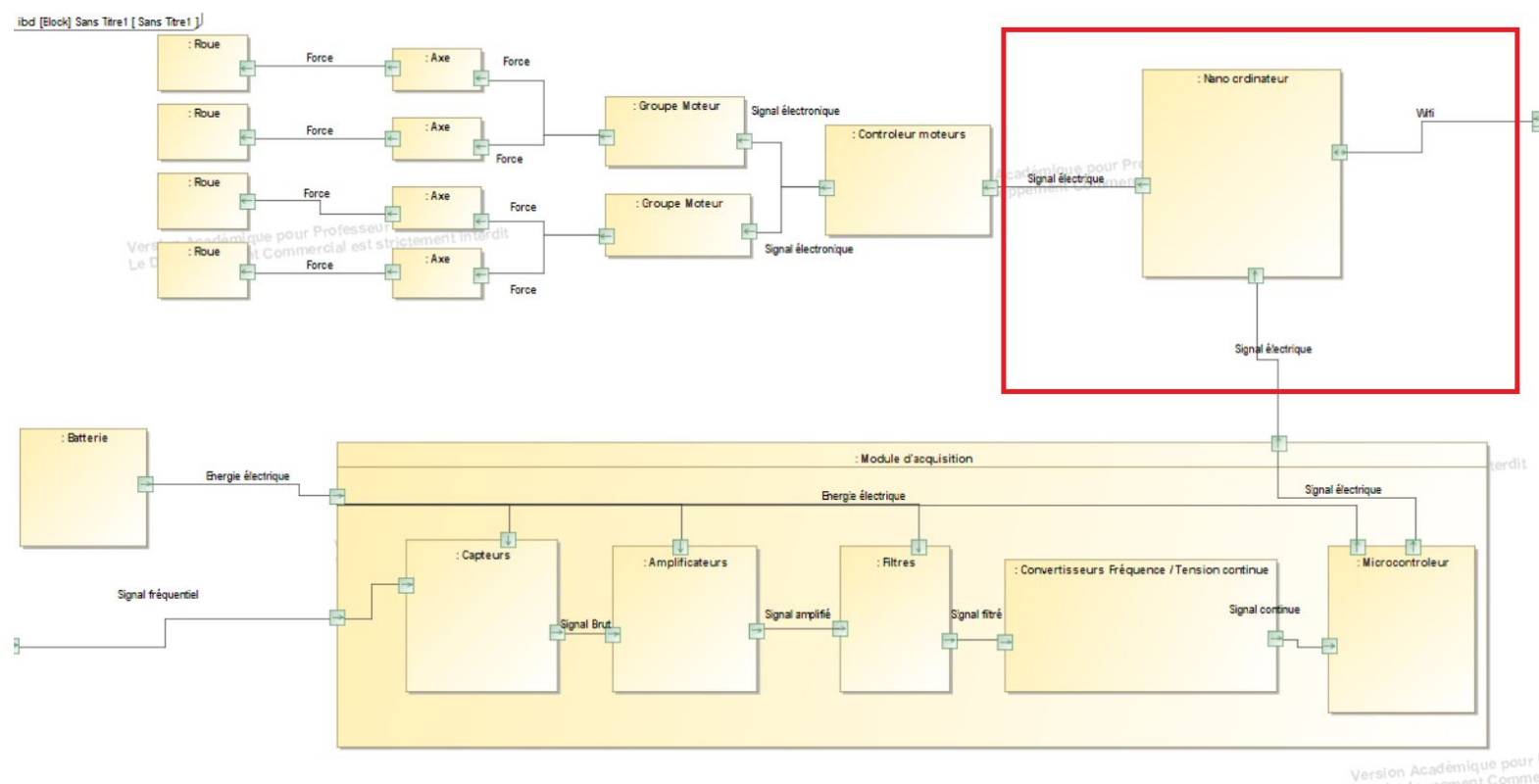
Diagramme de définition de bloc :



Le nano ordinateur sera composé d'un Serveur TCP afin de récupérer les informations de déplacement envoyer par la tablette. La communication se fera par le biais d'un mode ad hoc.

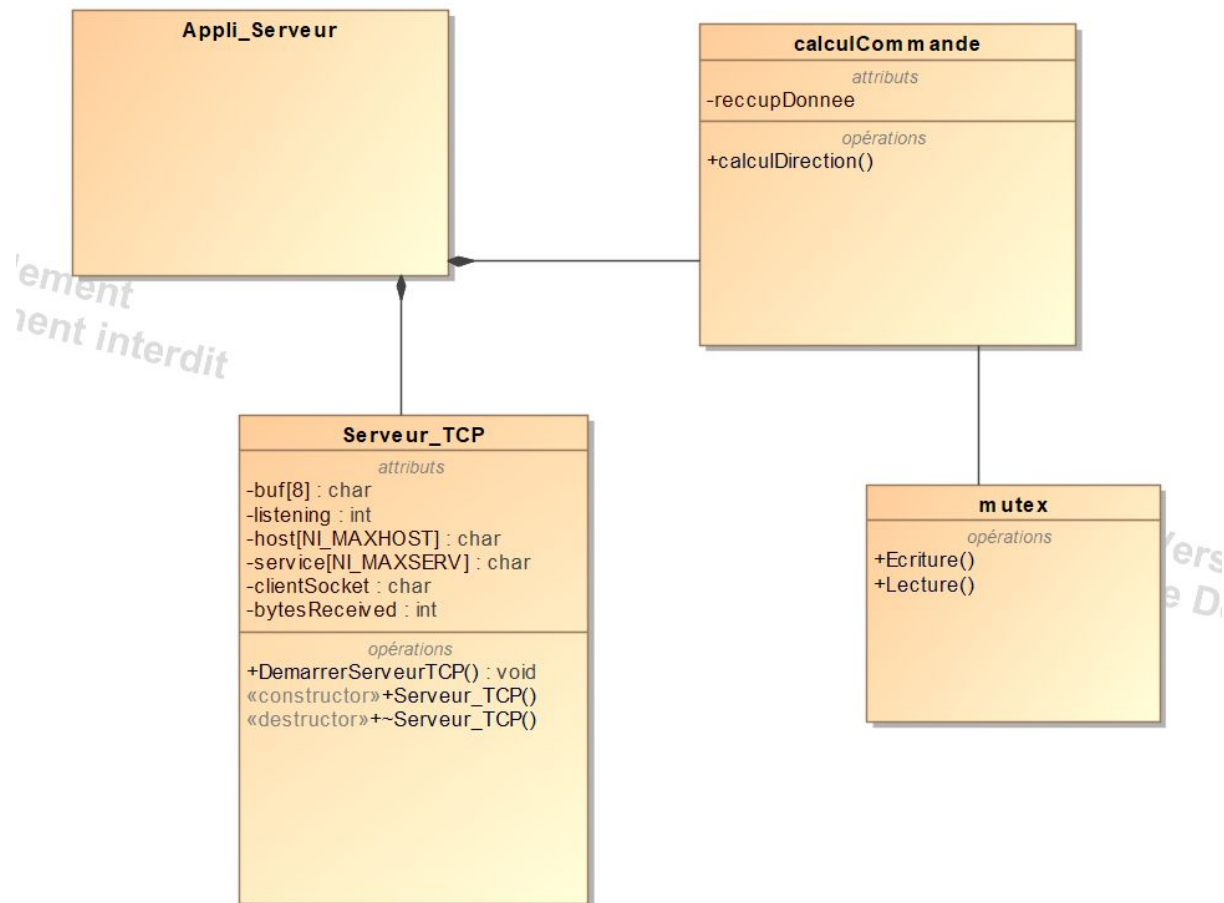
GerzynskiGaëtan

Diagramme de définition de bloc interne :



GerzynskiGaëtan

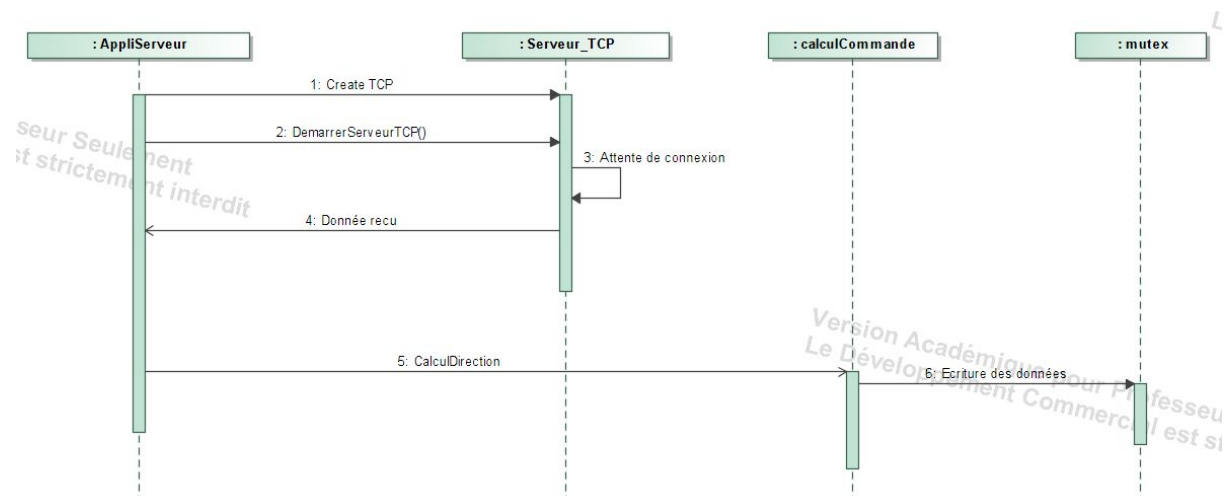
Diagramme de classes:



Le serveur Tcp se lance, attend une connexion. Lorsque la connexion est établi il reçoit les information de la tablette.

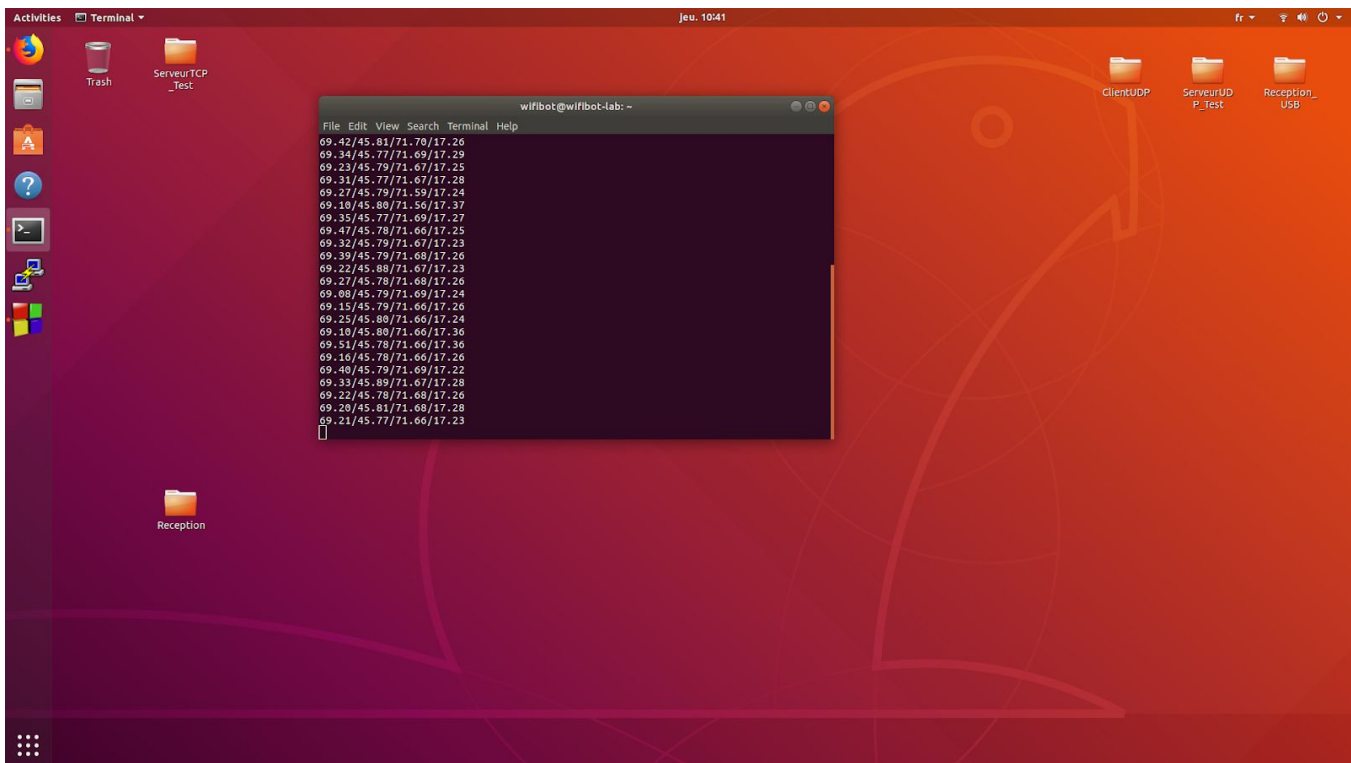
La classe calcul Commande sert à déterminer la direction que le rover doit prendre en fonction des information de capteur et du mode automatique ou manuelle. Il vas alors par la suite stocker ces information dans un fichier partager.

Diagramme de séquences:



L'application embarquée sur le Rover servira à créer un serveur, une fois démarré il attend une connexion. Après la connexion avec le client il reçoit les données envoyer par la tablette ensuite ces données serviront à calculer les déplacement du rover en mode automatique et manuel si le mode manuel et activer alors les déplacement se feront selon les donnée reçu de la part de la tablette et seront envoyer dans un fichier partagé de même pour le mode automatique mise a part la phase de calcul qui sera fait au préalable.

Test récupération des trames avec interface USB :

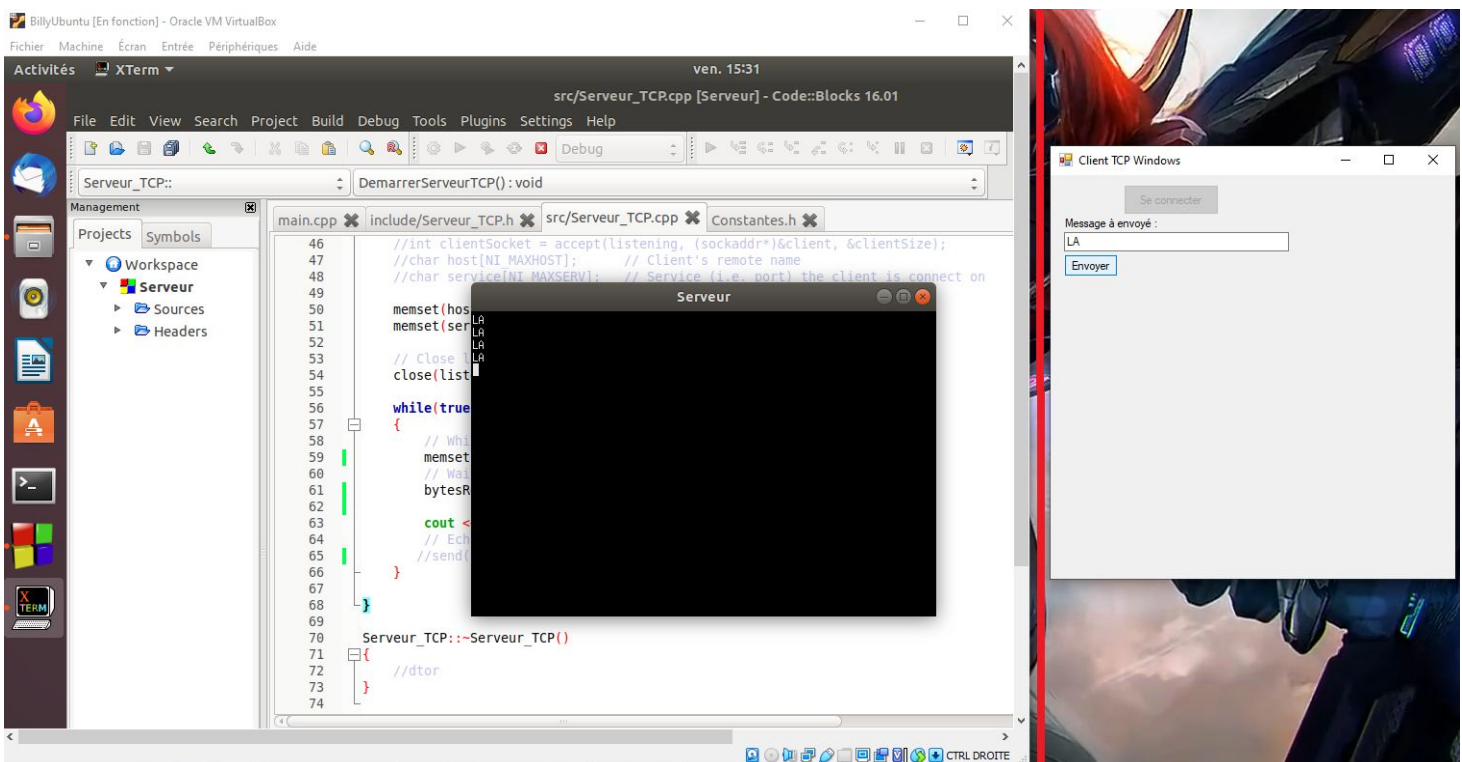
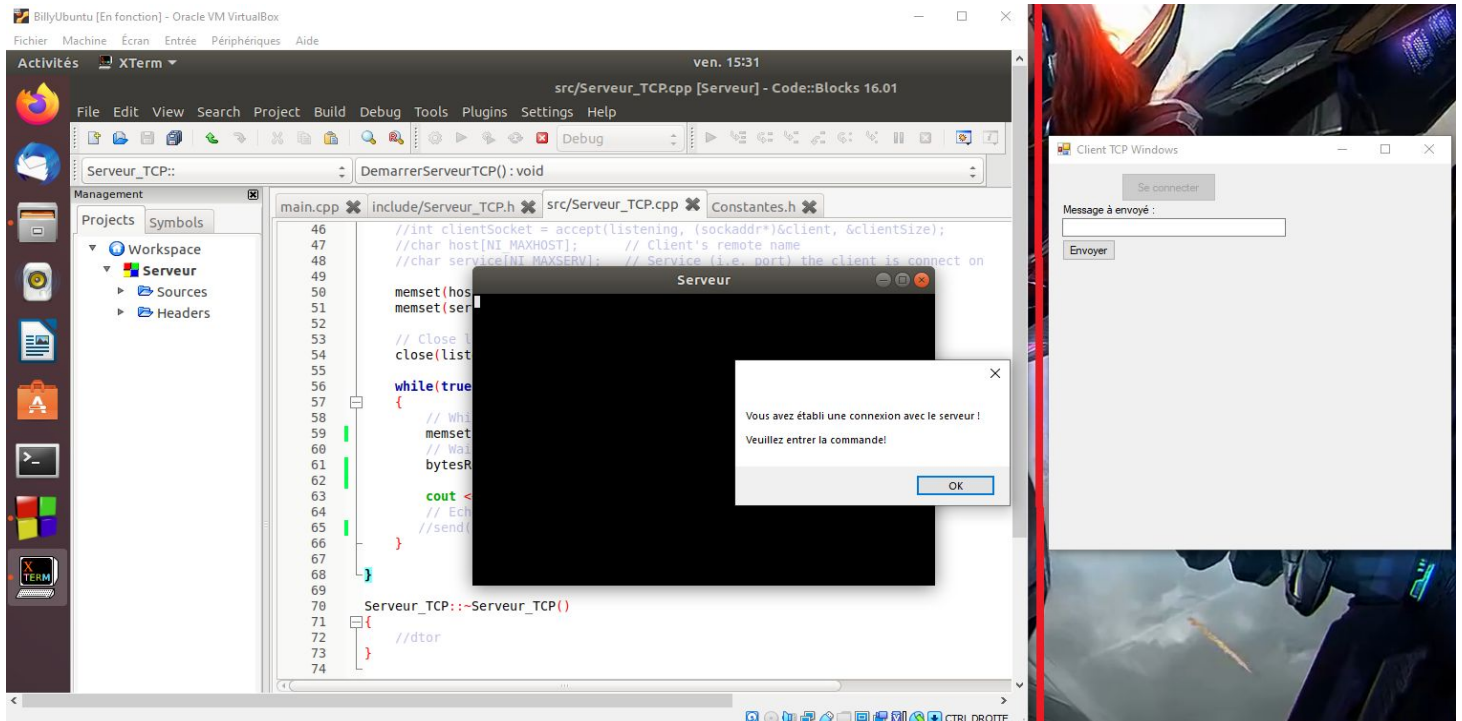


The screenshot shows a Linux desktop with a red background. A terminal window titled 'wifibot@wifibot-lab: ~' is open, displaying a list of network traffic data. The data consists of IP addresses and ports, such as '69.42/45.81/71.78/17.26'. The desktop has a sidebar with icons for 'Trash', 'ServeurTCP_Test', and 'Reception'. The top bar shows 'jeu. 10:41' and 'fr'.

Ici je récupère les données envoyer par les capteurs de la carte arduino grâce à minicom il faut configurer ne connaissant la port sur lequel il est branché. Comme on peut le voir on récupère bien les pourcentage des capteur sur le système d'exploitation linux.

GerzynskiGaëtan

Test récupération de données avec un serveur TCP:

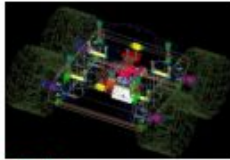


Gerzynski Gaëtan

Ici on test l'envoi des données du client TCP sur la tablette au Serveur TCP sur le Rover sur le client TCP de la tablette on envoie "LA" qui correspond à la commande avancé lentement et comme on peut le constater on la retrouve bien sur le Serveur TCP.

Gerzynski Gaëtan

Annexes



WifiBOT

Lab V4

Default Specifications

Motor sensor :	4 hall effect coders 336 ties / wheel turn
Speed control :	4 x PID DSPIC Microchip 33E coded in C RS232 Boot loader ICD2/3 (option)
Motors :	4x 12v Brushless motors 26:1 planetary gear 156 rpm
Dimensions:	L : 32 cm W : 37 cm H : 15 cm W : 3.8Kg
Power Batteries:	12.8V LIFEPO4 10AH Power supply 18V / 220V Path Power Management Charger inside the robot You can use the robot during charging
Control bus :	RS232/USB Simple protocol C/C++ API, (ROS, MatLab, RTMAPS, possible)
Distant Protocol :	Sockets TCP/UDP via WIFI or RJ45
CPU :	Intel Celeron J1900 quad core SBC 1.8Ghz 4G Ram / 60G SSD HD Nvidia Jetson Nano or Xavier NX Raspberry 3 or 4
Sensors :	4 Infrared 1 web cam wide angle CSI Camera for Nvidia
Software:	C++ control API 1 HMI Embedded Camera Web Server



**WIFI AP
(included)**



**DC 18V POWER
(included)**