Dossier Personnel

Jérémy RIBET

Sur la page de garde doivent apparaître :
- Segment Vol,
- Nom prénom de la personne auteure du dossier personnel
- Même page de garde pour TOUS...

Les captures à l'analyseur ne sont pas les meilleures choisies...

Table des matières

1 / Introduction générale :

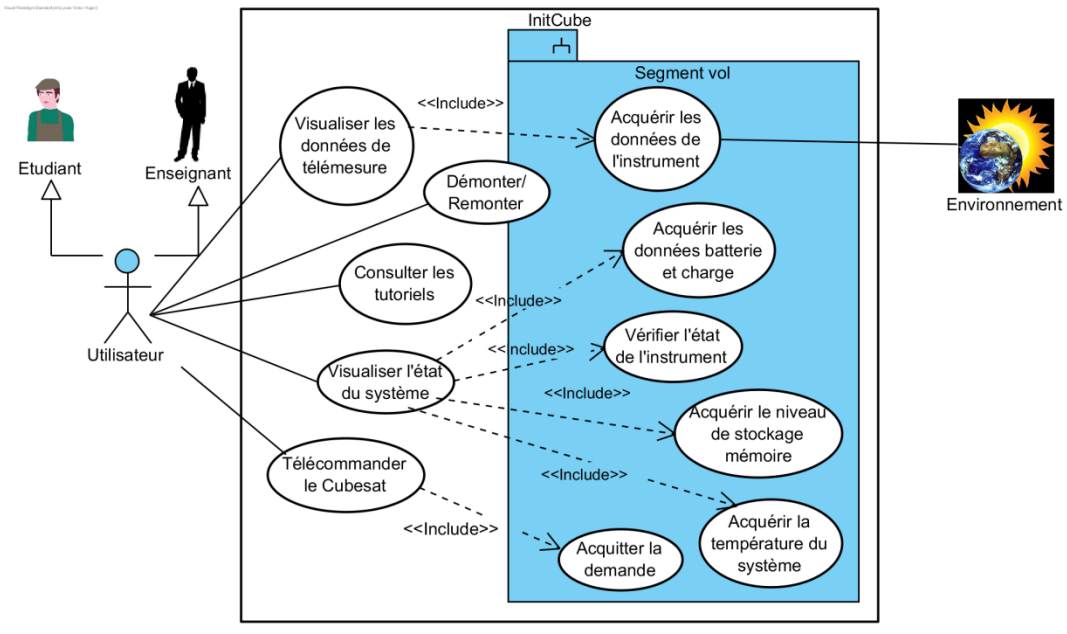
1.1 / Présentation du projet :

Jeunes en
Apprentissage pour la réalisation de
Nanosatellites au sein des
Universités et des écoles de l'enseignement
Supérieur

JANUS

Ce projet « J.A.N.U.S. » nous a été confié par le CNES, par l'intermédiaire de Mr Alain Gaboriaud. Il a pour objectif de créer un kit d'apprentissage pour des étudiants, afin de les former au concept de nano-satellites et ainsi les ouvrir au spatial. L'idée est de faire une réplique, qui simulera tous les comportements et les fonctionnalités d'un vrai CubeSat mais sur terre et avec les moyens d'une école ou d'un lycée.

Pour ce projet, deux groupes ont été formés. Un premier groupe s'est occupé du simulateur d'une station sol, appelée « Segment Sol » et qui récolte et met en forme les informations provenant du « Segment Vol ». Un second groupe s'est s'occupé de concevoir le simulateur du CubeSat en lui même, appelé ici « Segment Vol ».

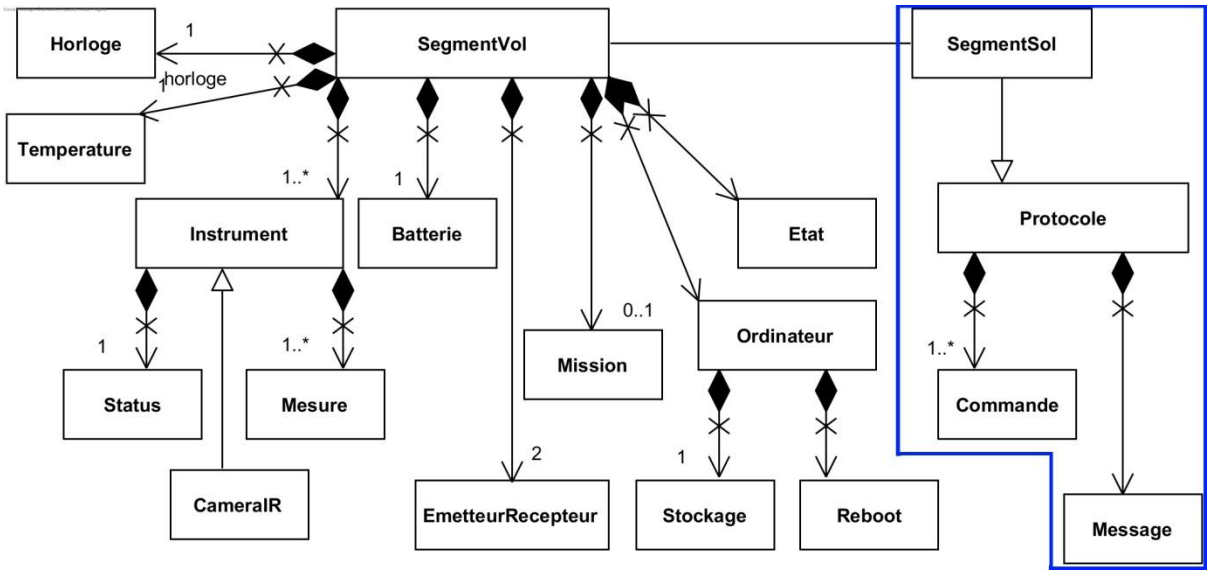


Pour ma part, je fais partie du “Segment Vol”. Mon groupe et moi avons réalisé un “InitCube” qui sera la réplique dans son principe d’un véritable CubeSat mais qui fonctionnera avec les contraintes terrestres et non spatiales. Nous avons divisé le travail en quatre parties. La première partie, étudiée par mon camarade Gwendal Loréal, concerne la charge utile choisie, c’est à dire une caméra Infrarouge qui représentera l’instrument de bord de cet incrément. La deuxième partie, étudiée par mon camarade Xavier Derozier, sera l’état et la gestion de la batterie (élément fondamental dans un système embarqué). La troisième partie, étudiée par mes camarades Lucas Marynus (Ordinateur de bord et température à bord) et Gwendal Loréal (État de l’instrument), concerne l’état du système et fournira toutes les valeurs en lien avec l’état de fonctionnement des éléments du système. La quatrième partie, la mienne, concerne donc la communication des données récoltées par les autres parties, au Segment Sol et la réception des télécommandes en provenance du Segment Sol.

1.2 / Présentation de la partie communication :

Le but de la partie communication va être d’assurer la communication et le bon fonctionnement des « télécommandes » en provenance du SegmentSol et des retours (télémesures et acquittements) du SegmentVol. J’ai travaillé sur les éléments de protocoles des deux cotés de cette communication.

Le diagramme des classes simplifié ci-dessous me situe par rapport au travail de mes camarades :



La classe **SegmentSol** a ici la particularité de représenter, sur le Segment Vol, la station au sol distante. Il ne s’agit pas du segment sol en lui même mais de la classe de gestion de la liaison série qui permet la communication avec les modules radios.

2 / Description du travail :

2.1 / Protocole

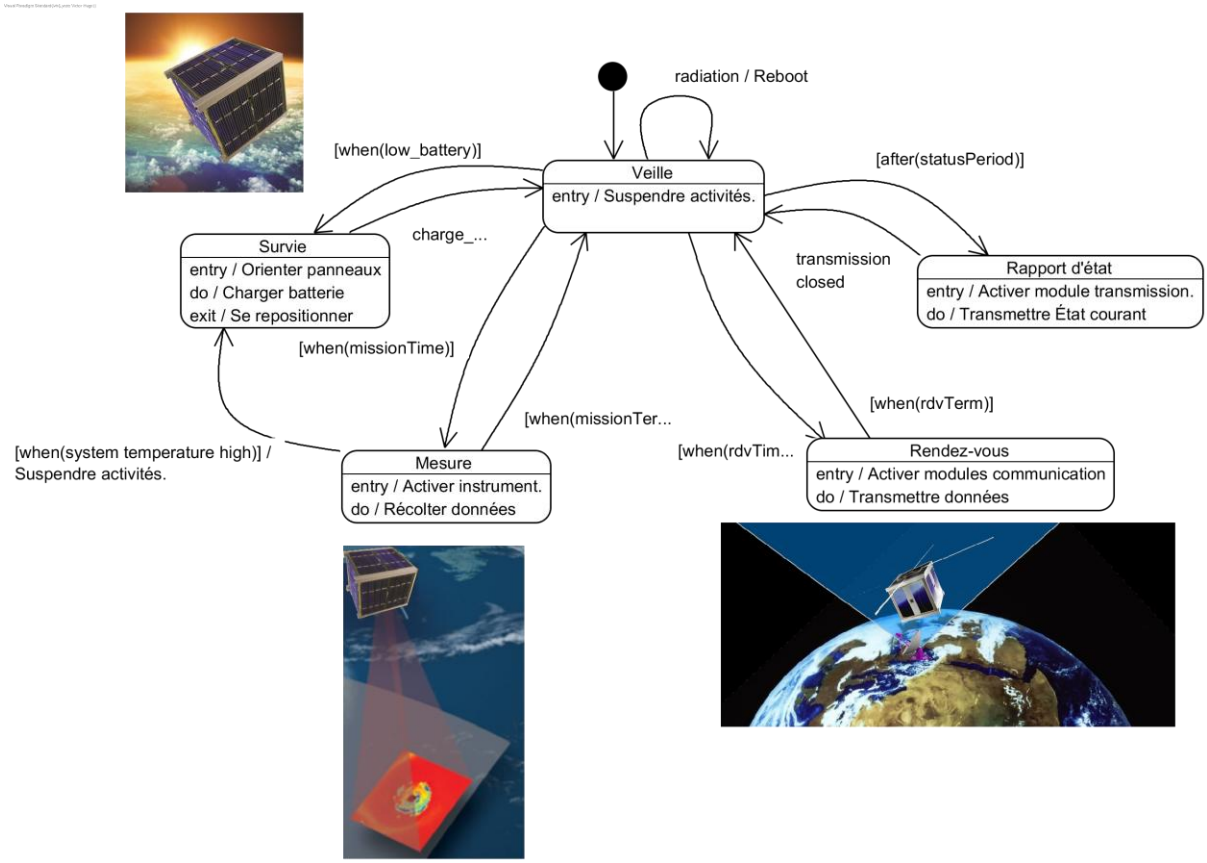
2.1.1 / Objectif :

L'objectif est de créer un protocole avec un format de trame pour réaliser une communication et un échange entre le Segment Sol et Vol. Le but de créer un protocole est de pouvoir se mettre d'accord entre les deux segments pour pouvoir exploiter toutes les informations qui seront récoltés et dont chacun aura besoin.

La première contrainte était une contrainte pédagogique. Les professeurs souhaitaient que la communication se fasse en ASCII afin que les messages soient plus lisibles dans un terminal.

La seconde contrainte était de respecter au mieux le type de télécommandes que l'on peut trouver sur un Cubesat.

Je me suis donc inspiré des états différents présentés par M. Gaboriaud au cours de la réunion du 14 février.



On voit sur ce diagramme d'états/transitions que le segment sol doit pouvoir :

1. programmer le segment vol afin qu'il exécute une mission pendant une période donnée. Cette mission consistera à faire des mesures périodiquement à partir d'une date de mission.

2. programmer une date de rendez-vous pour que le segment vol délivre l'ensemble de ses mesures de mission. Dans la réalité, le rendez-vous est automatique.
3. demander au segment vol de passer en mode survie si une surchauffe ou un niveau batterie faible est constaté,
4. demander l'état du segment vol,
5. demander que le segment vol effectue une mesure.

De son coté, le segment vol pourra transmettre périodiquement et automatiquement sont état au segment sol.

2.1.2 / Formats des trames :

Trames de télécommande :

Segment Sol <--------> Sement Vol

ID	Nbre d'octets suivant avec CMD incluse mais **hors checksum**	CMD	Payload ou données utiles	Checksum	\n
2 octets	1 octet	Code commande sur plusieurs octets	max 90 octets	2 octets	

Cette trame est le format général d'une trame de télécommande et de télémesure avec CMD pour le code de commande qui déterminera le type de commande demandée.

Codes Commande (CMD) :

MISSION	Configuration de mission
MEASURE	Demande ou transmission de mesure
STATUS	Demande ou transmission d'état du système
DEPLOY	Demande de déploiement antennes
SURVIVAL	Demande d'orientation du Cubesat pour recharge
EMPTY	Commande vidage mémoire
SAVE	Ordre d'enregistrement
MEETING	Configuration date de rendez-vous
DATE	Mise à l'heure de l'ordinateur de bord

Trame d'Acquittement :

Segment Vol ---------> Segment Sol

A la réception d'une télécommande, un acquittement est retourné. Il aura le format suivant :

ID	REPONSE	Checksum
2 octets	code Réponse (Cf Tableau 3) sur plusieurs octets	2 octets

Codes réponse :

MESSAGE sur état de la commande	Description
OK	La commande a été exécutée avec succès
FAIL	Échec de l'exécution de la commande
BUSY	Le dispositif est occupé, réessayer plus tard
ERROR	Erreur interne

Trame Mission :

Segment Sol --------> Segment Vol

~1	46	MISSION	- P 1 0	- D 1 1 0	- D T 2 0 1 9 / 0 2 / 1 3 ␣ 1 2 : 1 2 : 3 6	- T C	-SAVE	Checksum
2 octets	Nbre octets sur 1 octet Hors checksum	7 octets	Périodicité (10 mn) Ex : 4 octets (variable)	Durée (ici 110 mn)	Date Début mission au format AAAA/MM/JJ hh :mm :ss	Type mesure et unité	5 octets	2 octets

␣ : Espace (0x20)

Cette trame mission va transmettre la date de la trame et le type de mesure demandée.

Code type d'information :

-TC	Température en °C
-PIX	Attente : Pixels de la caméra IR
-P+ 2 caractères	périodicité des relevés en minutes
-D+ 3 caractères	durée pendant laquelle des relevés seront faits en minutes
-DT+ 19 caractères	Date de début de mission ou du relevé au format : AAAA/MM/JJ hh:mm:ss

Trame Meeting :

Segment Sol --------> Sement Vol

~1		MEETING	-	D	T	2	0	1	9	/	0	2	/	1	3		1	2	:	1	2	:	3	6	-	D	1	1	0	-SAVE	Checksum
2 octets	Nbre octets sur 1 octet Hors checksum	7 octets	Date Début mission au format AAAA/MM/JJ hh :mm :ss																						Durée du rendez-vous (ici 110 mn)					5 octets	2 octets

Cette trame permet de programmer la date de rendez-vous pour la restitution de la mission effectuée. .

Trame État du système :

Segment Sol --------> Sement Vol

~1		STATUS	-	B	O	R	D		-	I	N	S	T	-	P	1	0	Checksum
2 octets	Nbre d'octets	7 octets	Options facultatives. Si pas d'option sélectionnées, tout sera transmis en 2 paquets.											Périodicité des envois (10 mn)				2 octets

Cette trame permet au segment sol de demander l'état de certains appareils au segment vol. Dans le cas de notre incrément, l'état de tous les appareils est transmis.

Trames de transmission de données (télémesures) :

Segment Vol ---------> Segment Sol

~1	Nbre d'octets suivants mais **hors checksum**	Information	Payload ou données utiles	Checksum
2 octets	1 octet	Code information cf Tableau 1	max 90 octets	2 octets

Transmission de l'état du système :

ID	Nbre octets	STATUS	nbre Paquets	num paquet	code appareil	Données état appareil	code appareil	Données état appareil 2	Checksum
2 octets	sur 1 octet	7 octets			cf Tableau 9	cf Tableau 9	cf Tableau 9	cf Tableau 9	2 octets

Cette trame envoie l'état du système et organise les données grâce aux codes ci dessous :

-BORD	Ordinateur de bord	
	-SMo : Stockage SD libre en Mo	
	-S% : Stockage SD utilisé en pourcent	
	-RMo : mémoire RAM libre en Mo	
	-R% : occupation mémoire RAM en %	
	-DT : Date/heure à bord	
	-T : température du processeur en °C (*)	
-INST	Instrument	
	État de marche	-ON : allumé
		-OFF : Éteint
	-E : erreur	
	mode d'exploitation	-SLEEP : mode sleep
		-STBY : mode stand-by suivi de la durée (10 s ou 60 s)
		-NORM : normal
	-T : température de la caméra (*)	
-BATT	Batterie	
	-C : Niveau de charge en pourcentage	
	-V : Tension batterie en V	
	-A : Courant en mA	
	-LOAD : en charge	
	-T : température batterie (*)	
-REBOOT	Données de redémarrages dus aux radiations (particules énergétiques)	
	-DT : Date du dernier redémarrage	
	-N : Nombre de redémarrage	
-CUBE	-T : température dans le cube en °C (*)	

(*) **(Attention : les mesures de température sont précédées et suivies d'un espace)**

2.1.3 / Difficultés rencontrées

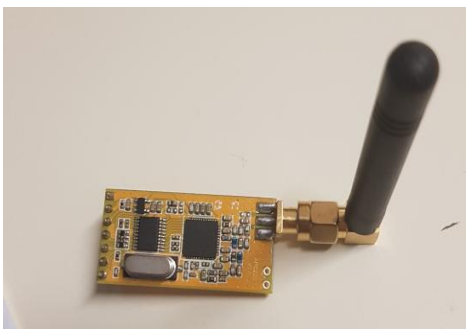
Il y a eu plusieurs difficultés rencontrées. Il a fallu définir le format, donc le nombre d'octets. Donc un long moment de réflexion a été nécessaire à la réalisation des formats.

2.2 / Architecture de la communication

Afin de pouvoir communiquer avec l'autre segment, il a fallu évidemment choisir un module de communication. Nous avions le choix entre deux modules :

Le module XBee

Module 433MHz



Ce comparatif a été fait a partir de quatre caractéristiques :

- La vitesse de transmission
- La tension utilisée
- Le type de signal utilisé
- La robustesse

	XBee 2.4GHz	Module 433MHz
Vitesse de transmission	250Kbps	De 1.2Kbps à 115.2kbps
Tension utilisée	3.3V	5V
Type de signal utilisé	TTL	TTL
Robustesse	Bon	Moyen

La comparaison faite, nous avons décidé avec un commun accord avec le Segment Sol d'utiliser les deux modules.

En effet, dans le cadre de nos recherches, nous avons constaté que dans le milieu de la météorologie de types de communications étaient utilisés.

Voici un extrait de l'article « https://www.spaceacademy.net.au/spacelink/radiospace.htm » :

- UHF Band
  - 399.9 - 403 MHz
    This band includes navigation, positioning, time and frequency standard, mobile communication, and meteorological satellites. Around 400 MHz is a companion band for satellites transmitting on 150 MHz.
  - 432 - 438 MHz
    This range includes a popular amateur satellite band as well as a few Earth resources satellites.
  - 460 - 470 MHz
    Meteorological and environmental satellites, includes uplink frequencies for remote environmental data sensors.
- S Band
  - 2.025 - 2.3 GHz
    Space operations and research, including 'deep space' links from beyond Earth orbit. This encompasses the Unified S-band (USB) plan which is used by many spacecraft, and which was also used by the Apollo lunar missions. It also includes military space links including the US Defense Meteorological

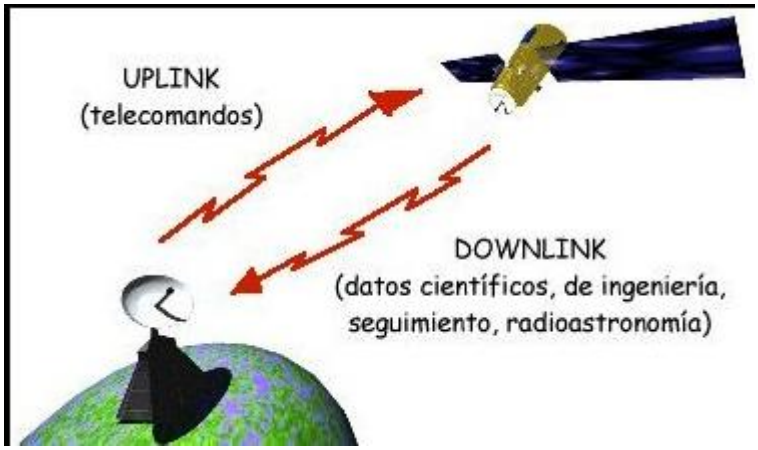
Satellite Program (DMSP). **Many Earth resources (remote sensing) satellites downlink in this band.**

- 2.5 - 2.67 GHz
  Fixed (point-to-point) communication and broadcast satellites, although the broadcast allocation is only used in some Asian and Middle-eastern countries.

Or, nous avons choisi de nous placer dans le contexte météorologique et de l'étude des points chaud terrestres (Camera IR de mon camarade Gwendal Loréal).

Le module 433MHz sera utilisé pour les transmissions montantes (du Segment Sol jusqu'au segment Vol) puisque qu'une trame de télécommande ne sera pas aussi conséquente qu'une trame de télémesures, qui peut contenir des images et d'autres grosses données. C'est pour cela que nous allons utiliser le module XBee qui supportera bien mieux les trames plus conséquentes grâce à ces 2.4 GHz de fréquence.

Le module XBee est un module de terrain, c'est à dire qu'il n'y aura jamais de perturbation sur sa fréquence, tandis que le module 433 MHz est sur une fréquence qui est beaucoup utilisée.



Il y aura une transmission Rx et Tx sur les deux segments. Le 433 Mhz en Tx du Segment Sol et en Rx sur le Segment Vol.

2.3 Choix de bibliothèques

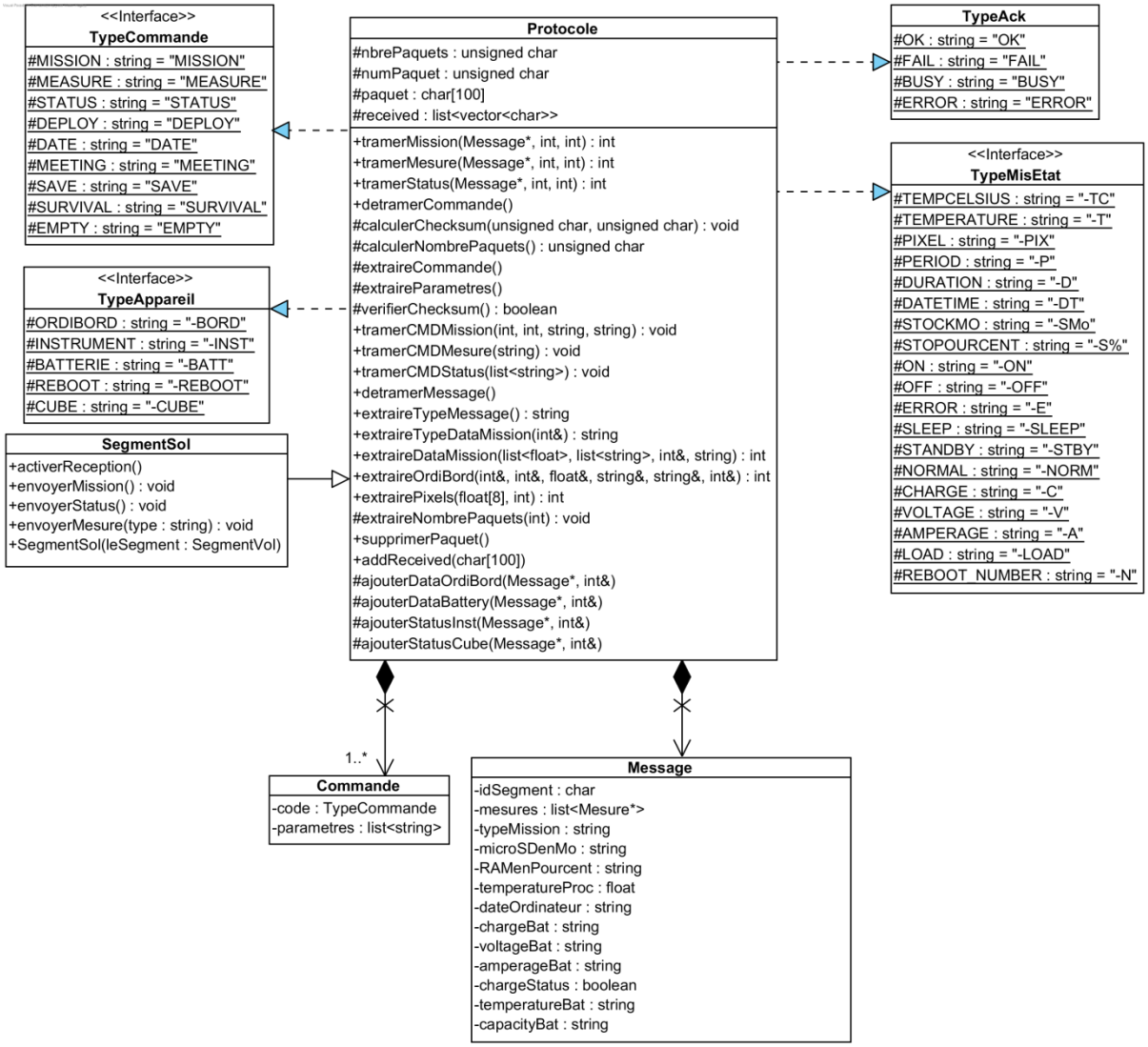
On m'a par ailleurs demandé de choisir une bibliothèque de communication avec la liaison série puis avec les fichiers de configuration que l'on a choisi en langage xml.

- serialib : Pour la communication série, j'ai choisie cette bibliothèque simple et légère. Il n'est pas nécessaire de l'installer.

- plugixml : pour la communication avec les fichiers de configuration xml j'ai choisi cette librairie qui me semblait être facile de mise en œuvre avec les exemples d'internet. Je n'ai pas eu le temps de l'utiliser dans le cadre du projet.

2.4 Étude des classes

2.4.1 / Détail des classes à ma charge



La partie communication représente une grande partie du projet. J'ai développé les classes :

- SegmentSol
- Commande
- Message
- EmetteurRecepteur
- TypeCommande
- TypeAppareil
- TypeAck
- TypeMisEtat

2.4.2 / Présentation des classes :

SegmentSol :

Comme dit précédemment, cette classe est spéciale. Elle représente le segment sol sur le segment vol et gère la liaison série de communication avec la station sol. Grâce a ses méthodes “ envoyer…..()”, cette classe ouvrira des canaux et permettra l’envoie des trames et leur acheminement jusqu'au segment sol.

Message :

Cette classe est une classe stockage provisoire des données récoltées par la classe SegmentVol à partir des différents capteurs. Elle est essentiellement constituée d’attributs, de setteurs (Modificateurs) et getteurs (Accesseurs). Les setteurs mettront également en forme certaines données avant transmission.

Protocole :

Cette classe est une classe clé pour la réalisation du protocole et l'élaboration de la trame. Grâce notamment aux méthodes “ Tramer…..()”, cette classe va permettre d’organiser les données et de les mettre dans un tableau. Une organisation en fonction des formats de trame de chacun des commandes et messages demandés.

Cette classe comporte également les méthodes d’extraction à destination du segment sol (Celui de l’autre équipe). Cela leur permettra de relever les données qu’ils pourront afficher dans leur IHM.

TypeCommande :

Cette classe est une interface de déclaration. Elle va permettre d'avoir des "codes" qui vont permettre de différencier et d'identifier les commandes qui seront demandées. Par exemple, si une trame comporte le nom de code " MEASURE", nous saurons que la mission sera de faire une mesure (température pour cet incrément). Ces codes sont des constantes en string.

TypeAppareil :

Cette classe va permettre d'avoir des "codes" qui vont permettre d'identifier l'appareil dont on donne les mesures. Par exemple pour le nom de code "-Batt", nous auront en suivant les mesures correspondant à la batterie de l'appareil, et cela jusqu'au prochain nom de code. Une trame d’état peut contenir et contiendra plusieurs appareils. Ces codes sont des constantes en string.

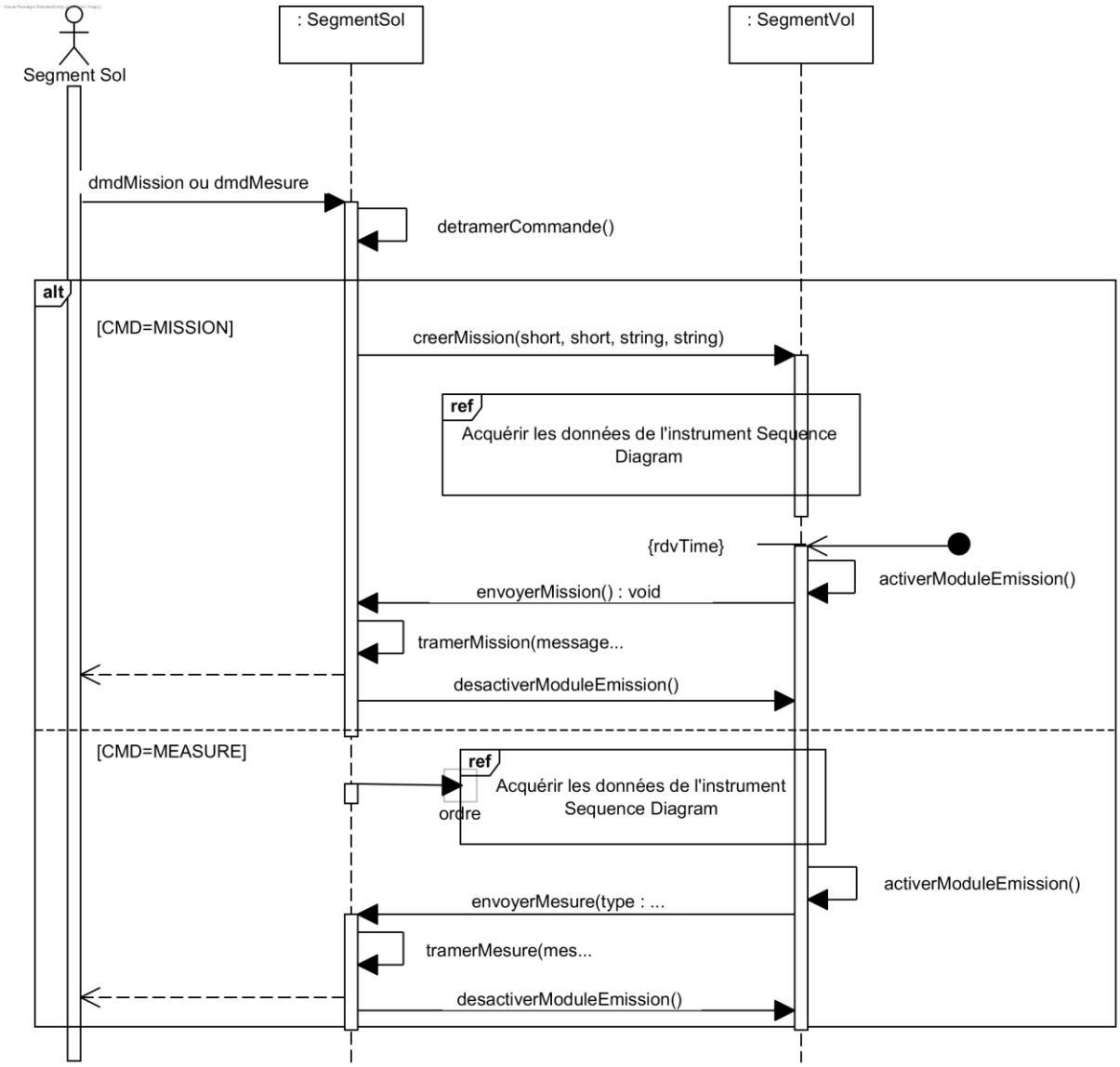
TypeAck :

Cette classe va permettre d'avoir des "codes" qui vont permettre d'identifier les différents type de réponse de l'appareil pour confirmer la bonne réception ou pas d'un message. Par exemple, pour le nom de code "OK" cela veut dire que l'on a une bonne réception du message et que l'initcube sera opérationnel pour réaliser la mission demandée. Ces codes sont des constantes en string.

TypeMisEtat :

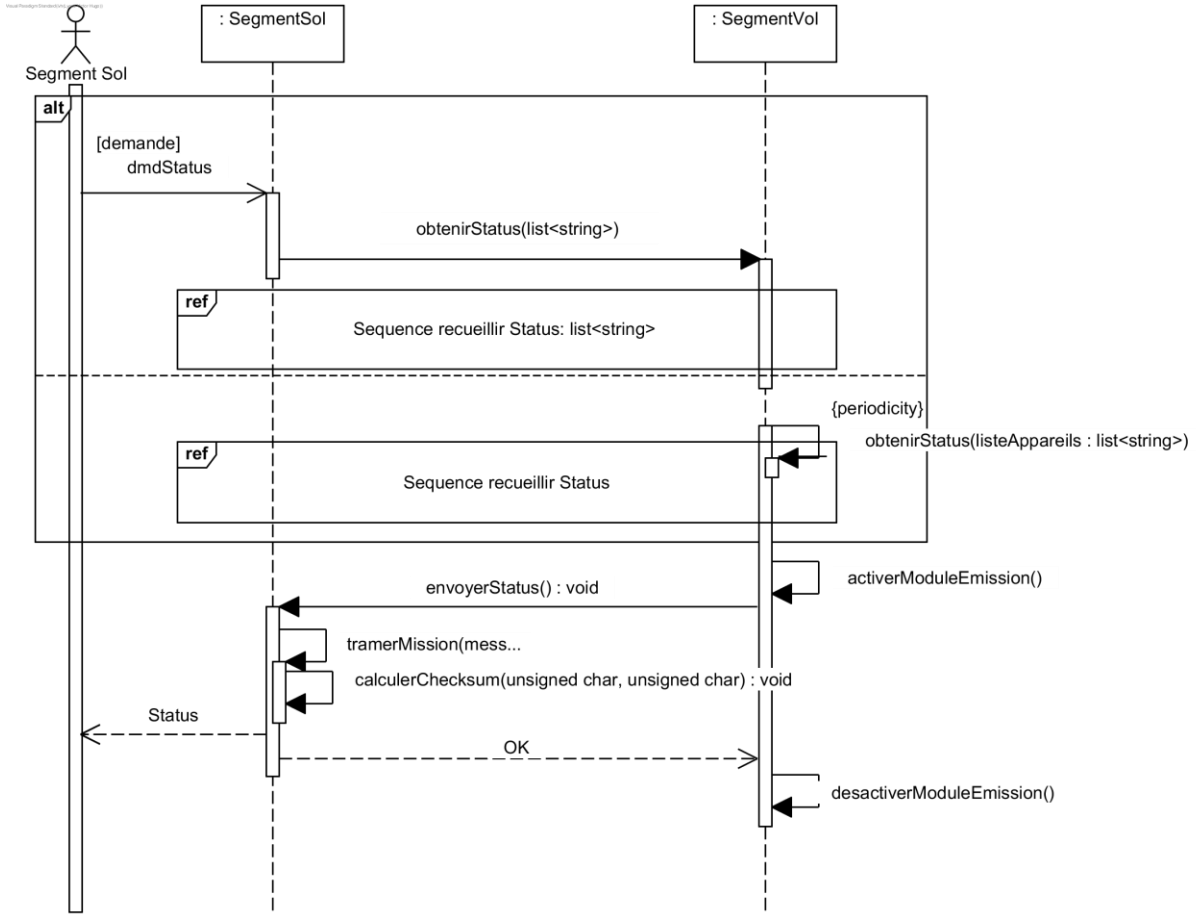
Cette classe va permettre d'avoir des "codes" qui vont permettre d'identifier les types de mesures dans une trame mais également quelques informations sur l'état du système. Par exemple le nom de code "-T" va permettre de voir que la valeur qu'il y a après est une température de l'appareil nommé avant grâce à la classe TypeAppareil, le nom de code “STBY” annonce que le cube est en mode “ standby” , cela va donc permettre d’avoir des information sur le cube . Ces codes sont des constantes en string.

2.4.3 Diagramme de séquence d’acquisition des mesures/transmission des télémesures d’instrument



2.4.4 Diagramme de séquence d'acquisition/transmission du status

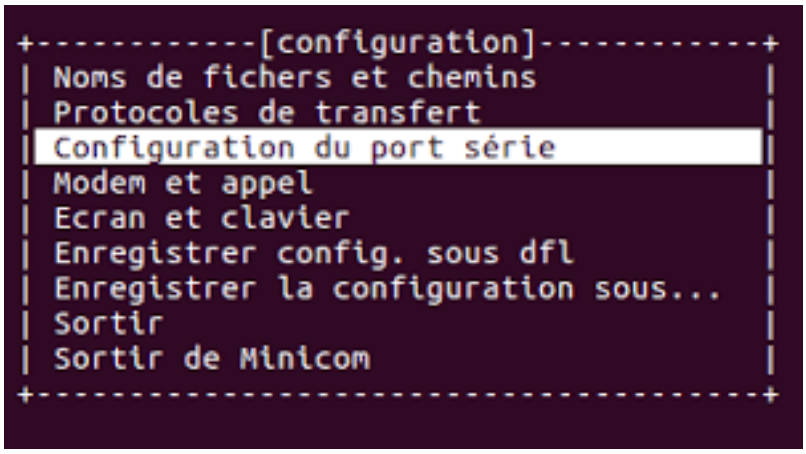
Ce diagramme de séquence présente une commande " STATUS" et montre tout le processus qu'il y a entre le Segment Sol et le Segment Vol :



2.5 / Les résultats

Une fois le format défini, des tests ont été effectués à l'aide de minicom avec le port série. Le but était d'afficher les données et afficher la trame pour voir si le code faisait bien le travail demandé.

Cette capture représente une trame " MISSION " qui a été tramée et envoyée sur le port série et observé sur le PC avec minicom :



Nous pouvons voir sur la capture que le format de la trame a bien été respecté. Les valeurs sont bien classées. Certaines valeurs s'affichent sous forme de symboles. Cela signifie que cela n'est pas de l'ASCII. Par conséquent, elle ne peut pas s'afficher.

Extrait du code :

```
void SegmentSol::envoyerMission(){
    serialib LS;
    char  Ret;
    unsigned char idSegment=leSegment->getIdentifiant();
    message->setIdSegment(idSegment);
    CameraIR* camera=leSegment->getCameraIR();
    list<Mesure*> mesures=camera->getMesures();
    message->setMesures(mesures);
    Mission * laMission = leSegment->getMission();
    string leTypeMission = laMission ->getMeasureType();
    message->setTypeMission(leTypeMission);
    int nbrePaquets = this->calculerNombrePaquets(message);
    for (int i=0;i<nbrePaquets;i++)
    {
        Ret=LS.Open(DEVICE_PORT,9600);

        tramerMission(message, nbrePaquets, i+1);

        Ret=LS.Write(tableau,tableau[2]+6);
        LS.Close();
    }
    camera->clearMesures();
    mesures=camera->getMesures();

}
```

Récuppération des informations de segment vol.

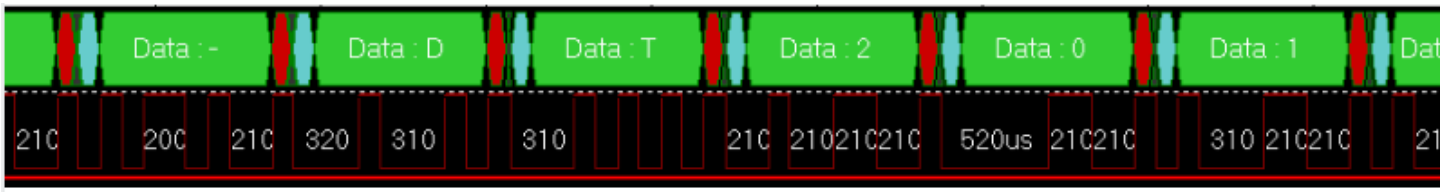
Gestion des paquets

ouverture du port avec serialib

écriture du tableau tramé

La trame de mission (ensemble de données datées), trop grande pour un seul paquet, doit être divisée sur plusieurs paquets. Ces paquets sont séparés par un retour à la ligne.

Pour vérifier que toutes les informations sont présentes, il a fallu analyser la trame avec un analyseur logique.



Grâce à cette analyse, nous pouvons voir l'organisation de la trame et les détails des informations qui ne se sont pas affichées sur le minicom.

3 / Journal de bord

Semaine	Activité principale
1	Découverte du projet
2	Choix du module de communication grâce à des Tests et des analyses.
3	Test et choix des librairies serial et pugiXML
4	Analyse UML
5	Définition du protocole
6	Semaine de revue
7	Début de codage
8	Codage de Protocole
9	Codage de Protocole
10	Codage de Protocole
11	Codage de Segment Sol
12	Semaine de revue et de rédaction du dossier
13	Codage Segment Sol
14	Test integration
15	Préparation Revue

4 / Environnements utilisés

Fedora

Fedora est un système d'exploitation issu de Linux , c'est celui ci que j'ai utilisé pour sa facilité d'utilisation

NetBeans

Ce logiciel est un Environnement de Développement Intégré , il m'a permis de développer le code plus facilement , en organisant tous mes codes dans un projet .

Raspberry pi3

C'est un micro-ordinateur qui nous as permis à chacun de notre côté avec notre Raspberry de travailler dans notre "coin" . Elle servira également de carte principale dans notre projet.

Visual Paradigm

Visual paradigm est un logiciel qui m'a permis de réaliser mes diagrammes. Il est habilité à faire tous les diagrammes UML.

TimePerformance

Cet outil nous a permis de nous organiser notre travail et d'utiliser la méthode scrum. Le projet a été sur cinq sprints.

Google Drive

Google drive nous a permis d'organiser tous les documents que nous avons, et de se les partager mutuellement.

Analyseur Logique

Il m'a permis de prendre des captures de mes trames circulant sur le port auquel je lui aurais demander.

5 / Tests Unitaires

Test unitaire

Projet	Nom de l'élément testé	Type de l'élément testé	Version	Incrément
InitCube	Protocole	Classe	1	

Description :	Cette classe permet de tramer les données et de les préparer à l'envoie .

Description du test

Scénarios concernés	Ce test unitaire va permettre de savoir si les données se trient correctement .
Description	En lancant le code , les données vont se mettre dans un tableau
Environnement nécessaire	Système opperationnel
Situation initiale	Raspberry et Poste
Classes de tests nécessaires	SegmentSol , TypeCommande , TypeAck
Nom du script	TU_SegmentSol , TU_TypeCommande , TU_TypeAck

Auteur du test

Nom du testeur	Date	Conclusions	Validation
Ribet Jérémy	28/05/19 DocInformation:Created	La trame est bel et bien formé	Oui

Description du script

	Traitement	Paramètres en entrée	Résultats attendus
1	tramerMission	message , nbrePaquets , numPaquets, Id	Les données Missions doivent être triés dans un tableau .
2	tramerStatus	message , nbrePaquets , numPaquets, Id	Les données Status doivent être triés dans un tableau .
3	tramerMesure	message , nbrePaquets , numPaquets, Id	Les données Mesure doivent être triés dans un tableau .
4	extraireTypeMessage	pos	Le type de message doit être stocker dans une variable
5	extraireNombrePaquets	pos	Le Nombre de paquets doit être stocker dans une variable
6	extraireTypeDataMission	pos	Le type de donnée doit être stocker dans une variable
7	extraireDataMission	list<float> &datas , list<string> &dateHours,int &pos, string typeMission	Les données doivent être stcker dans une variable
8			
9			

Problèmes identifies

Traitement	Résultats obtenus	Gravité de l'erreur

Test unitaire

Projet	Nom de l'élément testé	Type de l'élément testé	Version	Incrément
InitCube	Segment Vol	Classe	1	

Description :	Cette classe permet d'envoyer les trames

Description du test

Scénarios concernés	Ce test va permettre de voir si l'envoie des données se réalise bien
Description	En lancant le code , les données vont s'envoyer .
Environnement nécessaire	Système opperationnel
Situation initiale	Raspberry et Poste
Classes de tests nécessaires	Protocole , TypeCommande , TypeAck
Nom du script	TU_Protocole , TU_TypeCommande , TU_TypeAck

Auteur du test

Nom du testeur	**Date**	**Conclusions**	**Validation**
Ribet Jérémy	28/05/19 DocInformation:Created	La trame est bel et bien envoyé	Oui

Description du script

	Traitement	Paramètres en entrée	Résultats attendus
1	envoyerMission		Les données mission doivent être envoyer dans la disposition du format de tramerMission
2	envoyerStatus		Les données Status doivent être envoyer dans la disposition du format de tramerStatus
3	envoyerMesure		Les données de Mesures doivent etre envoyer dans le format de tramerMeasure
4			
5			
6			
7			
8			
9			

Problèmes identifies

Traitement	Résultats obtenus	Gravité de l'erreur

Test unitaire

Projet	Nom de l'élément testé	Type de l'élément testé	Version	Incrément
InitCube	Message	Classe	1	

Description :	Cette classe permet d'initialiser les valeurs et de les convertir

Description du test

Scénarios concernés	Ce test va permettre de voir si les valeurs sont bien initialiser et converties
Description	En lancant le code , les données vont se convertir en string
Environnement nécessaire	Système opperationnel
Situation initiale	Raspberry et Poste
Classes de tests nécessaires	
Nom du script	

Auteur du test

Nom du testeur	Date	Conclusions	Validation
Ribet Jérémy	28/05/19 DocInformation:Created	Les données sont converties	Oui

Description du script

	Traitement	Paramètres en entrée	Résultats attendus
1	setTemperatureProc	atemperatureProc	Les valeurs sont converties de float en string
2	setChargeBat	achargeBat	Les valeurs sont converties de char en string
3	setAmperageBat	aamperageBat	Les valeurs sont converties de short en string
4	setTemperatureBat	atemperatureBat	Les valeurs sont converties de float en string
5	setVoltageBat	avoltageBat	Les valeurs sont converties de float en string
6	setCapacityBat	acapacityBat	Les valeurs sont converties de short en string
7	setTemperatureCube	atemperatureCube	Les valeurs sont converties de float en string
8			
9			

Problèmes identifies

	Traitement	Résultats obtenus	Gravité de l'erreur

6 / Bilan

Ce projet est très intéressant, d'abord parce que le client est prestigieux. C'est une grande entreprise et les attentes sont présentes. Le sujet est également très intéressant, nous travaillons dans le domaine très fermé du spatial.

Je pense que le projet doit être réalisé par des étudiants, l'aspect apprentissage est bien maîtrisé par des étudiants.

Ce projet m'a confirmé ce que j'ai appris en entreprise au niveau de l'exigence, qui est assez grande mais aussi par rapport au cahier des charges, qui doit être respecté.

Enfin cela m'a aidé à apprendre plus de choses dans le domaine de l'informatique et du développement, très présents dans ma partie.