

Projet InitCube Vol

Sur la page de garde doivent apparaître :

- Segment Vol,
- Nom prénom de la personne auteure du dossier personnel
- Même page de garde pour TOUS...
- Penser à préciser SegmentVol dans le nom du fichier



Sommaire

1. Introduction.....	3
1.1 Notre travail.....	3
1.2 Répartition du travail.....	3
2. Ma mission dans ce projet.....	4
2.1 Explication de mon travail.....	4
2.2 Répartition dans le temps.....	5
3. Logiciel utilisés.....	8
4. Travail réalisé.....	9
4.1 Diagramme de séquence :.....	9
4.2 Diagramme de classe.....	10
4.3 Récupération mémoire :.....	11
4.4 Qu'est ce que popen() ?.....	12
4.5 Capteur de température du système.....	13
4.5.1 Choix du capteur.....	13
4.5.2 Comment trouver l'adresse du capteur ?.....	14
4.5.3 Conversion du résultat obtenu.....	15
5. Test Unitaire.....	18
5.1 Classe Ordinateur.....	18
5.1.1 Description du test.....	18
5.1.2 Auteur du test.....	18
5.1.3 Description du script.....	19
5.1.4 Problèmes rencontrés.....	19
5.2 Classe Stockage.....	20
5.2.1 Description du test.....	20
5.2.2 Auteur du test.....	20
5.2.3 Description du script.....	21
5.2.4 Problèmes identifiés.....	21
5.3 Classe Temperature.....	22
5.3.1 Description du test.....	22
5.3.2 Auteur du test.....	22
5.3.3 Description du script.....	23
5.3.4 Problèmes identifiés.....	23
6. Test Intégration.....	24
6.1 Classe Ordinateur/SegmentVol.....	24
6.1.1 Description du test.....	24
6.1.2 Auteur du test.....	24
6.1.3 Description du script.....	25
6.1.4 Problèmes rencontrés.....	25
6.2 Classe Ordinateur/Stockage.....	26
6.2.1 Description du test.....	26
6.2.2 Auteur du test.....	26
6.2.3 Description du script.....	27
6.2.4 Problèmes rencontrés.....	27
7- Bilan.....	28

1. Introduction

1.1 Notre travail

Le but du projet proposé par le CNES est de créer un prototype de CubeSat (Nano-Satellite) pour initier les étudiants à l'Espace.

Le projet proposé par le client est donc séparé en deux. Un groupe qui traite le segment vol et l'autre le segment sol.

Nous nous attarderons que sur la partie Vol puisque c'est cette partie que j'ai traité.

1.2 Répartition du travail

Notre équipe est composée de quatre étudiants de BTS :

LOREAL Gwendal : Il s'est occupé de l'instrument à bord du nano-satellite, la caméra infra-rouge.

DEROZIER Xavier : Il s'est chargé de la récupération de toutes les informations de la Pi-Juice et de la batterie avec le codage de ses classes puis la classe SegmentVol.

RIBET Jeremy : Il s'est concentré sur la communication entre la partie Vol et la partie Sol des télémessures et des télécommandes en créant un protocole de communication et en codant les classes qu'il fallait.

MARYNUS Lucas : Je me suis occupé de l'état de l'ordinateur de bord qui est une raspberry. J'ai codé la classe Ordinateur qui permet de récupérer la température du processeur. Je me suis chargé du codage de la classe Stockage qui permet de récupérer la mémoire libre de la carte SD, le pourcentage de mémoire utilisée, la mémoire RAM libre et l'occupation de la mémoire RAM.

J'ai aussi récupéré la température du système à l'aide d'un capteur de température positionné entre la batterie et la Pi Juice pour savoir l'état du système.

Grâce à toutes ces informations, j'ai codé la classe Reboot qui permet de passer l'InitCube en mode survie lorsqu'il y a une mémoire RAM trop saturée, ou une température soit du processeur soit du système trop élevée ou trop faible. J'ai déterminé ces seuils sans trop de précision puisque nous n'avons pas les réelles conditions spatiales.

2. Ma mission dans ce projet

2.1 Explication de mon travail

Mon travail a été de visualiser l'état du système ou plus précisément, de l'ordinateur de bord. Pour se faire, j'ai codé plusieurs classes qui m'ont permis de traiter mon objectif.

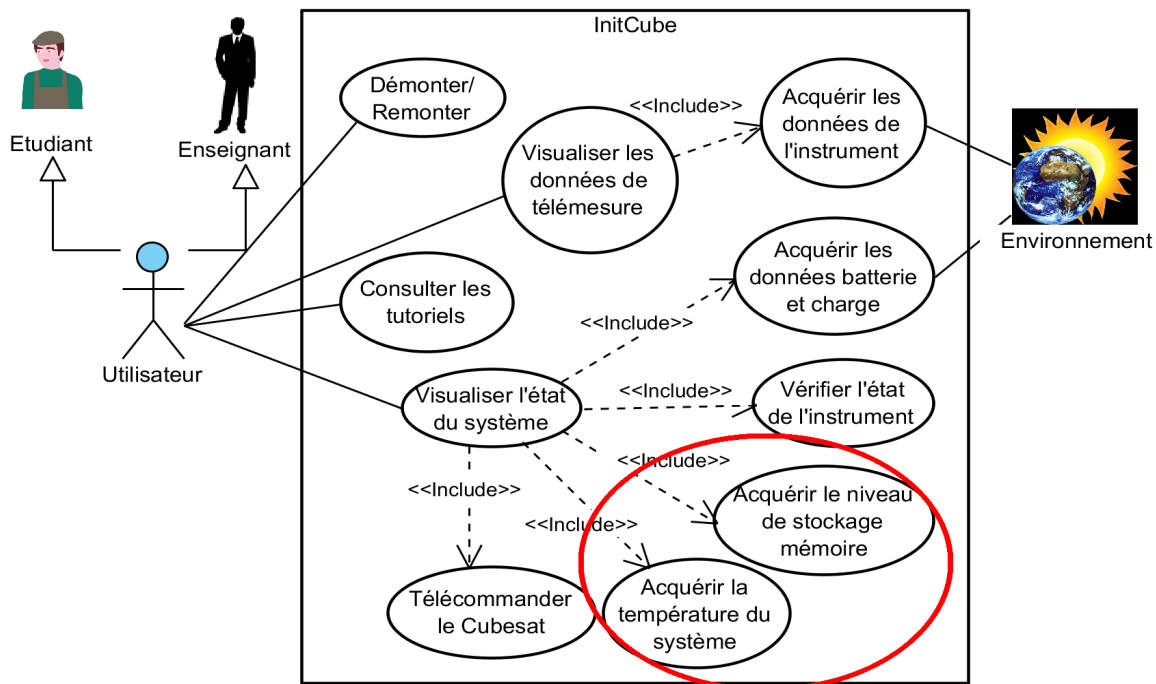
-La classe Stockage qui permet de créer un tube de communication unidirectionnel à l'aide de popen() où on y récupère l'exécution d'une commande Shell que je traite pour obtenir les valeurs qui m'intéressent.

-La classe Ordinateur qui permet de récupérer la température du processeur. Elle est en lien avec la classe Stockage puisque le lien est une composition ce qui permet à la classe Ordinateur de lancer le calcul de la mémoire à l'aide de la méthode obtenirStatus().

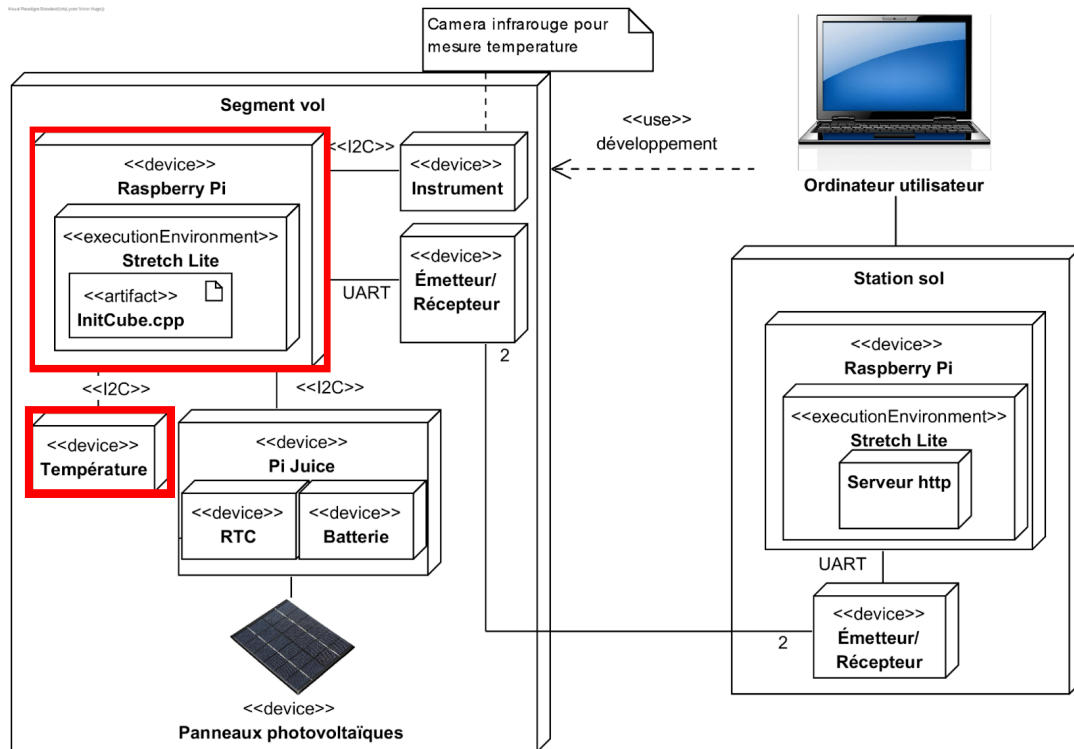
-La classe Temperature qui, quant à elle, permet de récupérer la température du système à l'aide du capteur de température.

-La classe Reboot, qui permet de passer en mode survie lorsque les seuils de températures ou de mémoire définis sont dépassés.

Diagramme de cas d'utilisation :



Architecture matérielle :



2.2 Répartition dans le temps

Sprint 1 Étape Terminé

4 fév 2019 – 22 fév 2019

Profil par défaut 31 jh

Liste des livrables Ajouter

Installation	Terminé	1 jh · 4 fév 2019
Choix materiel	Terminé	1 jh · 5 fév 2019
Validation du matériel	Terminé	7 jh · 11 fév 2019
Cahier de recette	Terminé	4 jh · 22 fév 2019
Analyse UML	Terminé	6 jh · 22 fév 2019
Choix d'un protocole de communication	Terminé	8 jh · 22 fév 2019
Compte rendu réunion client	Terminé	2 jh · 14 fév 2019
Validation acces fichier configuration	Terminé	2 jh · 22 fév 2019

Lors du premier sprint, nous avons lu le sujet et étudié le cahier des charges du client. On a ensuite établi la répartition des tâches.

Dans un second temps, lorsque tout le monde savait ce qu'il avait à faire nous avons fait les choix matériels.

J'ai aussi commencé à prendre connaissance du capteur de température ainsi qu'à le tester avec un code en C. D'autant qu'il communique en I2C, j'ai dû me rappeler des notions vues en cours sur la communication I2C.

Sprint 2		Étape	Terminé		11 mar 2019 – 29 mar 2019
Profil par défaut 9 jh					
Liste des livrables					
Ajouter					
+	Revue 1 projet	Terminé			1 jh · 11 mar 2019
+	Corriger diagramme	Terminé			1 jh · 29 mar 2019
+	Rédiger les tests unitaires mission	Terminé			3 jh · 20 mar 2019
+	Développer les classes principales nécessaires à la mission	Terminé			4 jh · 29 mar 2019

Le deuxième sprint a été le début du c++. Le code de test que j'avais réalisé lors du sprint 1 n'étant pas sous forme de classe, il fallait que je traduise mon code en c++ pour pouvoir l'avoir sous la bonne forme.

Pour cela, l'équipe a opté pour créer ensemble une classe I2C ~~qui hérite de, pour ma part, Temperature.~~ dont la classe Temperature héritera.

Sprint 3		Étape	Terminé		1 avr 2019 – 19 avr 2019
Profil par défaut 21 jh					
Liste des livrables					
Ajouter					
+	Finir la rédaction des test unitaire	Terminé			2 jh · 2 avr 2019
+	Rédiger les plans de test intégration mission/mesure	Terminé			2 jh · 3 avr 2019
+	Développer les classes de mission	Terminé			8 jh · 19 avr 2019
+	Finir le développement des classes principales	Terminé			5 jh · 11 avr 2019
+	Réaliser les tests Unitaires	Terminé			3 jh · 3 avr 2019
+	Réaliser des tests Intégration	Terminé			1 jh · 4 avr 2019

Le sprint 3 a été la continuité du sprint 2 puisque j'ai continué à coder la classe Temperature que je n'avais pas fini lors du sprint 2 et j'ai ensuite commencé les tests de récupération de mémoire de la µSD.

Lorsque mes premiers tests étaient concluant, j'ai poursuivi mon travail avec les tests de récupération de la mémoire RAM. J'ai fini le sprint 3 en écrivant le test d'intégration de la classe Temperature.

☐ Sprint 4 Étape En cours ⋮

6 mai 2019 – 24 mai 2019

Profil par défaut 26 jh

Liste des livrables ➕ Ajouter

⊕ Finir de rediger et realiser les tests	En cours ⋮	4 jh · 9 mai 2019
⊕ Développement des classes d'état du système	En cours ⋮	8 jh · 17 mai 2019
⊕ Compléter dossier	En cours ⋮	5 jh · 17 mai 2019
⊕ Développer les classes en lien avec les mesures	En cours ⋮	? · 24 mai 2019
⊕ Effectuer l'intégration des classes de Mission	En cours ⋮	? · 24 mai 2019
⊕ Réunion de concertation/Modification Classe I2C	En cours ⋮	1 jh · 6 mai 2019
⊕ Réunion - sprint meeting	En cours ⋮	1 jh · 9 mai 2019
⊕ Configuration par défaut des missions	En cours ⋮	7 jh · 24 mai 2019

Le sprint 4 a été exclusivement du codage de classes, Stockage et Ordinateur. Lorsque j'ai terminé de coder ces classes, j'ai réalisé le test unitaire de chacune d'entre elle puis l'intégration. Quant au dernier, je me suis occupé de coder la classe Reboot ainsi que de réaliser le test unitaire.

3. Logiciel utilisés

Netbeans: Environnement de développement utilisé pour coder les classes.



Visual Paradigm: Logiciel de conception UML utilisé pour créer les différents diagrammes.

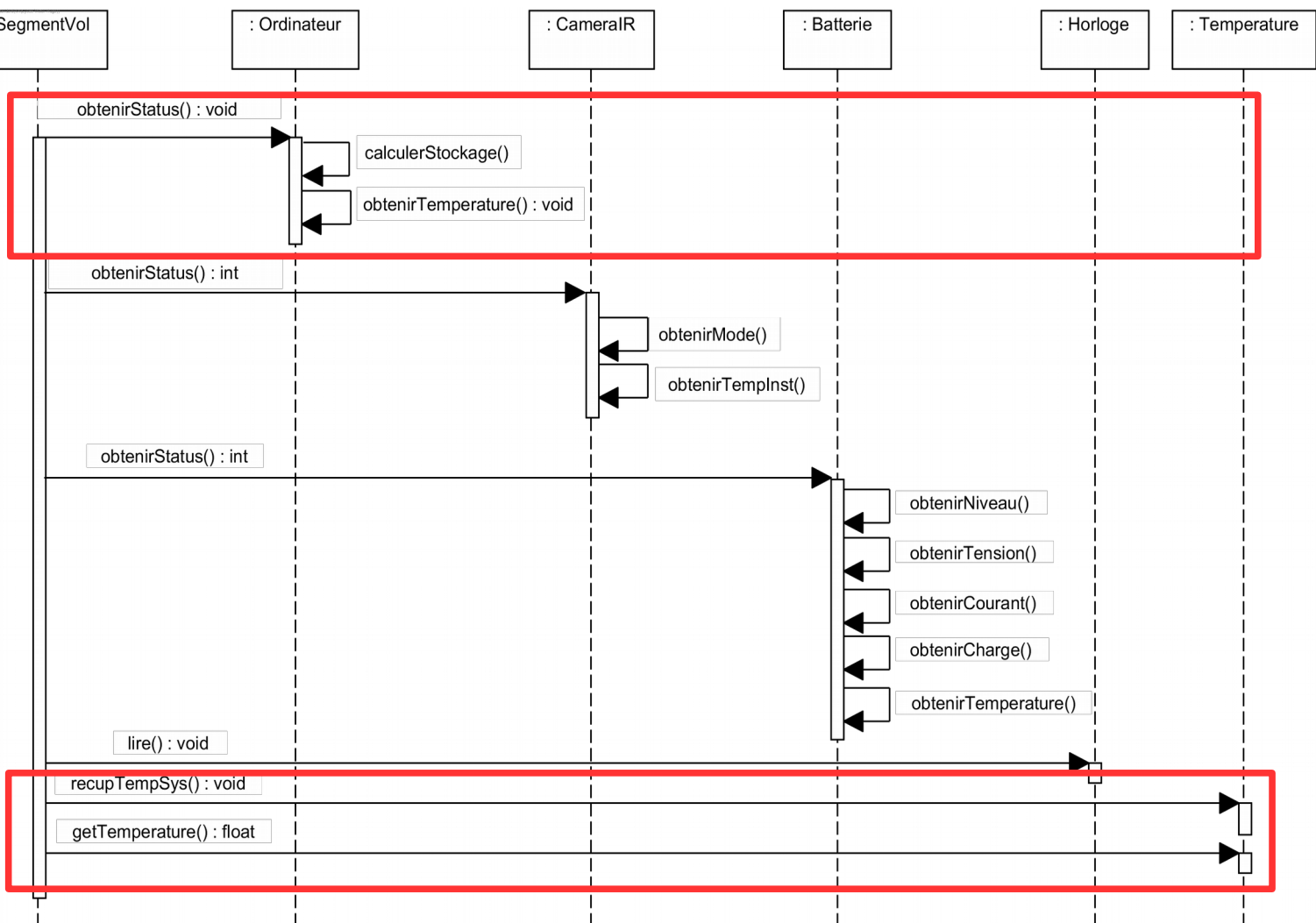


TimePerformance: Logiciel qui nous a été utile pour organiser notre temps et notre projet.



4. Travail réalisé

4.1 Diagramme de séquence :

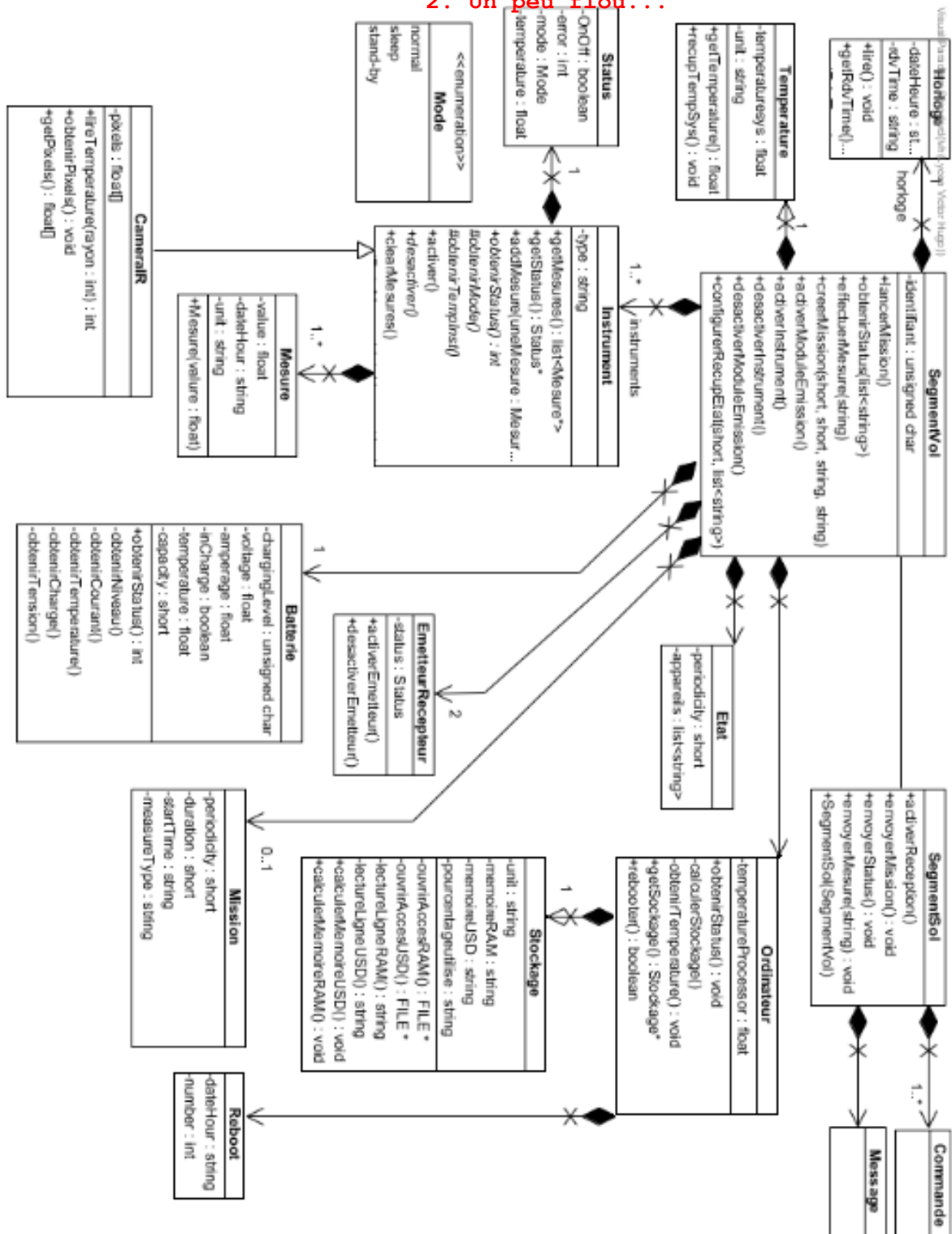


A commenter!

On voit sur ce diagramme comment la classe `SegmentVol` de mon camarade M. Derozie appellera les méthodes de ma classe `Ordinateur` puis de ma classe `Temperature`. `Ordinateur` est composée de la classe `Stockage` et `Reboot`.

4.2 Diagramme de classe

1. Ce n'est pas la bonne version
2. Un peu flou...

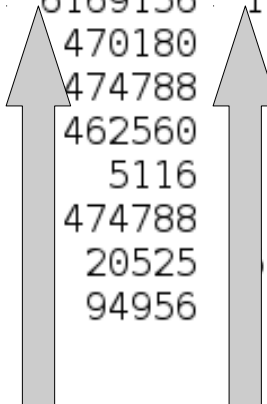


4.3 Récupération mémoire :

Résultat de la commande Shell : df -k

Un peu gros quand même...

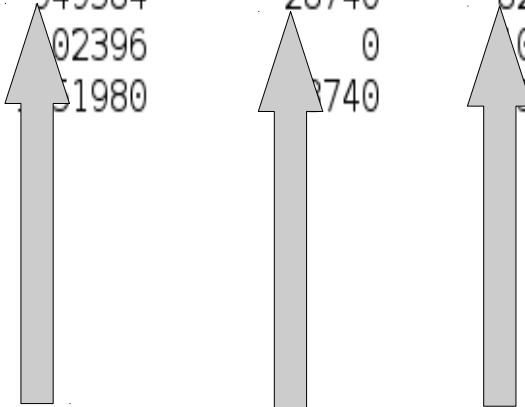
```
pi@raspberrypi:~ $ df -k
Filesystem      1K-blocks    Used Available Use% Mounted on
/dev/root        7564920 1055828   6169156  15% /
devtmpfs         470180      0    470180   0% /dev
tmpfs            474788      0    474788   0% /dev/shm
tmpfs            474788    12228   462560   3% /run
tmpfs             5120         4     5116   1% /run/lock
tmpfs            474788      0    474788   0% /sys/fs/cgroup
/dev/mmcblk0p1   41853     21328    20525   1% /boot
tmpfs            94956      0     94956   0% /run/user/1000
```



Valeur que nous voulons récupérer pour la mémoire de la µSD

Résultat de la commande Shell : free -t

```
pi@raspberrypi:~ $ free -t
              total        used        free      shared  buff/cache   available
Mem:          949584        28740       829340        12196        91504       860552
Swap:          02396           0        02396
Total:        951980        28740       831736
```



Afin de l'obtenir en pourcentage...

Je récupère ces trois valeurs puisque l'occupation RAM ne peut que être calculée. Pour cela je récupère la mémoire RAM totale et la mémoire utilisée. Je fais un simple calcul qui est $\text{memoireramutilisé}/\text{memoireramtotale}$.

4.4 Qu'est ce que popen() ?

Cette fonction permet de créer un canal de communication entre deux processus et récupérer le résultat de l'exécution d'une commande. Dans ce cas, nous mettons le résultat de cette commande dans une variable qui est de type FILE * mais qui est considéré comme un tube..

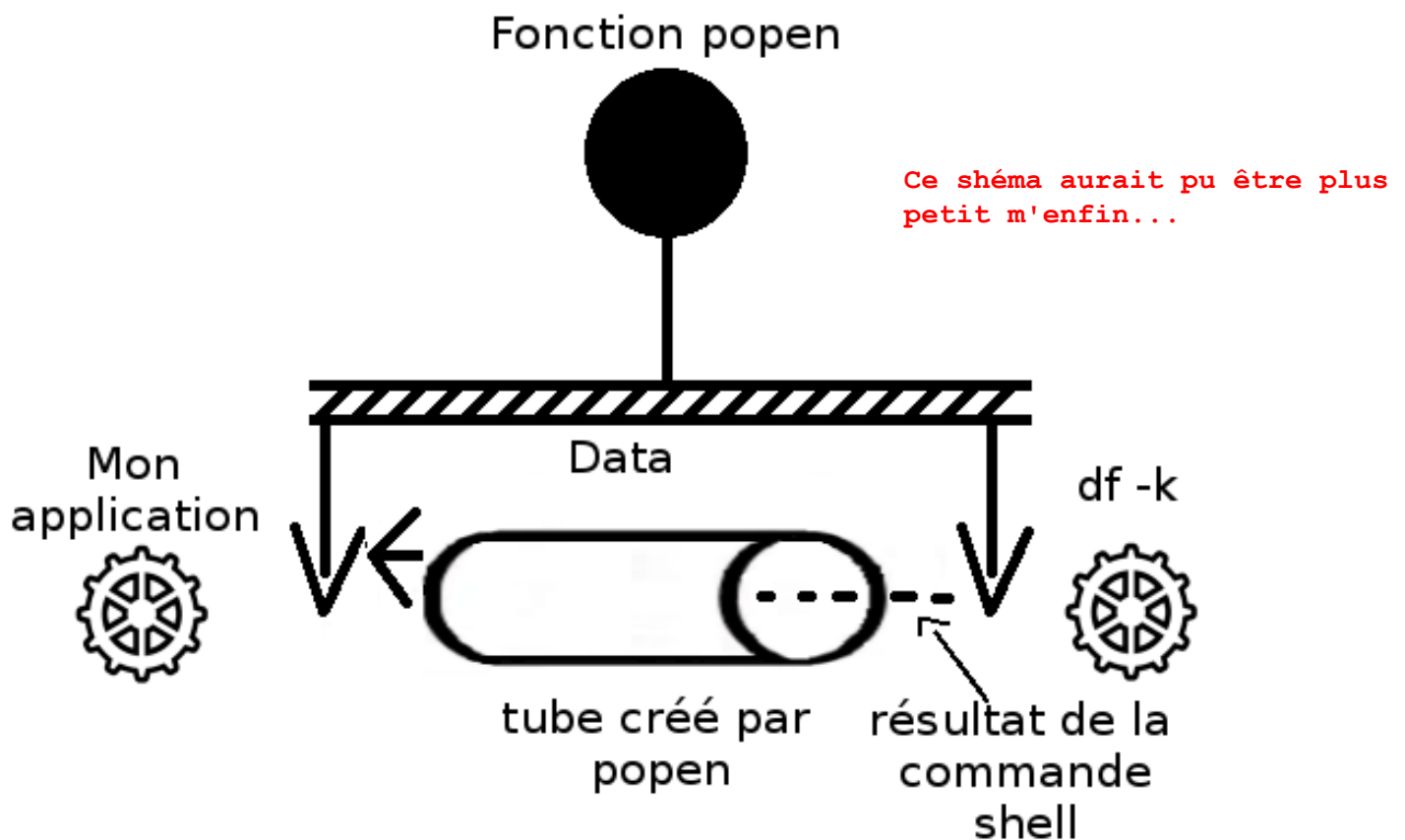
Ligne de code :

```
FILE * fichierUSD = popen("df -k", "r");
```

```
FILE * fichierRAM = popen("free -t", "r");
```

Principe

d'utilisation :

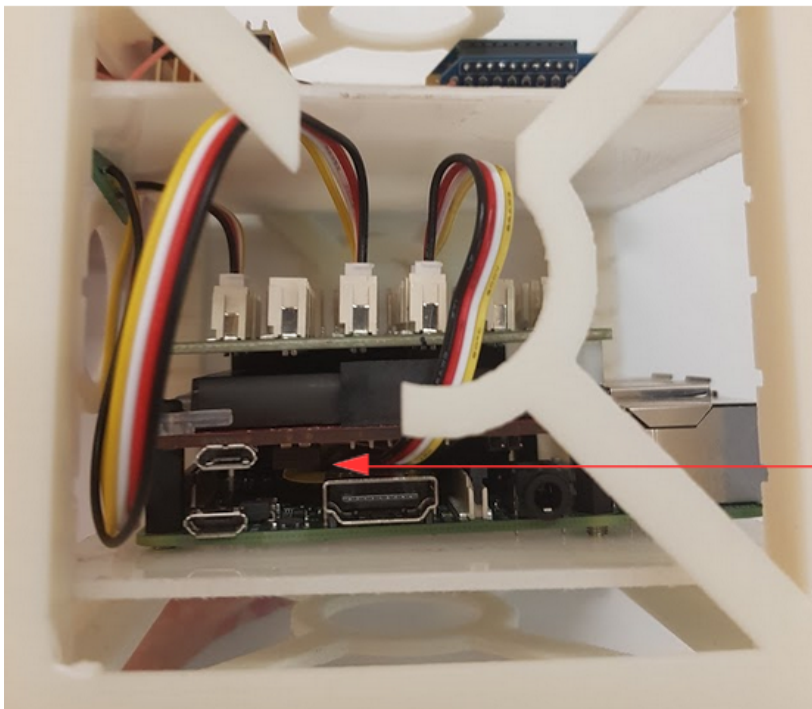


Le résultat de la commande **df -k** est donc stocké dans le tube créé nommé fichierUSD que nous ouvrons en lecture d'où le « r » qui permet de lire le tube.

Le résultat de la commande **free -t** est donc stocké dans le tube créé nommé fichierRAM que nous ouvrons en lecture.

Je traite donc ces deux fichiers, qui sont en fait des tubes, pour récupérer les données montrées au 4.3

4.5 Capteur de température du système



Positon de mon capteur sur dans l'InitCube.



Sur ou dans???

4.5.1 Choix du capteur

	MCP 9808	HIH 6120
Vdd (V)	Mini : 2.7 Max : 5.5	Mini: 2.3 Max : 5.5
Courant	Max : 400µA	1 mA
Temperature range (°C)	-40 à 125	-25 à 80
Courant de sommeil (µA)	0.1	3.3

Suite à ce tableau de comparaison, j'ai choisi le capteur MCP 9808 en raison du courant de sommeil qui est bien moindre puisque l'InitCube n'est pas illimité en énergie, il faut faire attention à la dépense d'énergie. Les autres points relevés étant similaires ou presque, j'ai donc fait le choix du capteur par rapport au critère du courant de sommeil.

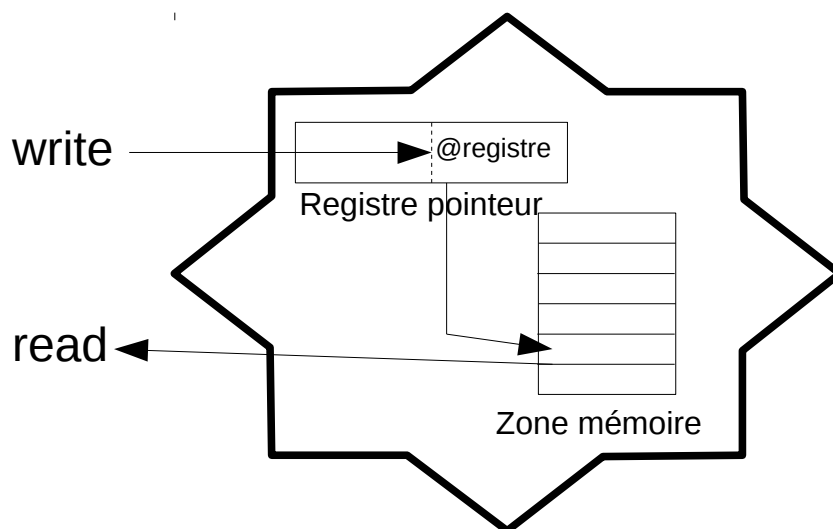
Ce capteur communique en I2C.

4.5.2 Comment trouver l'adresse du capteur ?

Dans un premier temps, j'ai cherché l'adresse de mon capteur pour pouvoir communiquer avec lui. Je me suis servi de la commande `i2cdetect -y 1`

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
00:				--	--	--	--	--	--	--	--	--	--	--	--	--
10:	--	--	--	--	--	--	--	--	--	--	--	--	1c	--	--	--
20:	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
30:	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
40:	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
50:	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
60:	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
70:	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

Sur la documentation, il est expliqué comment marche le capteur et notamment comment la mémoire est répartie. Voici un schéma explicatif :



Ce schéma montre que le module capteur choisi est composée en plus du capteur en lui même et d'une interface de communication I2C, d'un pointeur registre et d'un tableau de registres.

Le pointeur de registre nous permet de positionner la lecture en face du registre qui nous intéresse.

~~Nous pouvons donc voir que la mémoire du capteur est composée de deux zones.~~

~~La première en haut à gauche nous permet de voir à quoi correspondent les registres de pointeurs.~~ Dans la documentation cela est traduit comme cela :

de registre

- 0000 = RFU, Reserved for Future Use (Read-Only register)
- 0001 = Configuration register (CONFIG)
- 0010 = Alert Temperature Upper Boundary Trip register (T_{UPPER})
- 0011 = Alert Temperature Lower Boundary Trip register (T_{LOWER})
- 0100 = Critical Temperature Trip register (T_{CRIT})
- 0101 = Temperature register (T_A)
- 0110 = Manufacturer ID register
- 0111 = Device ID/Revision register
- 1000 = Resolution register
- 1xxx = Reserved⁽¹⁾

Note 1 : Some registers contain calibration codes and should not be accessed

Je voulais donc récupérer la température et celle-ci se trouve à l'adresse de registre 0101 qui est le code binaire. Par simple calcul, $2^0 + 2^2 = 1 + 4 = 5$. L'adresse est donc 0x05.

Après avoir demandé au processeur de communiquer en I2C avec mon capteur (méthode `ioclt()`), nous nous plaçons à l'adresse du registre et nous lisons à partir d'ici.

4.5.3 Conversion du résultat obtenu

Maintenant que nous savons où sont les données qui nous intéressent, nous pouvons traiter ce tableau encore une fois présent dans la documentation du capteur.

Register Pointer (Hex)	MSB/LSB	Bit Assignment							
		7	6	5	4	3	2	1	0
0x00	MSB	0	0	0	0	0	0	0	0
	LSB	0	0	0	1	1	1	1	1
0x01	MSB	0	0	0	0	0	Hysteresis		SHDN
	LSB	Crit Loc	Win Loc	Int Clr	Alt Stat	Alt Cnt	Alt Sel	Alt Pol	Alt Mod
0x02	MSB	0	0	0	SIGN	2 ⁷ °C	2 ⁶ °C	2 ⁵ °C	2 ⁴ °C
	LSB	2 ³ °C	2 ² °C	2 ¹ °C	2 ⁰ °C	2 ⁻¹ °C	2 ⁻² °C	0	0
0x03	MSB	0	0	0	SIGN	2 ⁷ °C	2 ⁶ °C	2 ⁵ °C	2 ⁴ °C
	LSB	2 ³ °C	2 ² °C	2 ¹ °C	2 ⁰ °C	2 ⁻¹ °C	2 ⁻² °C	0	0
0x04	MSB	0	0	0	SIGN	2 ⁷ °C	2 ⁶ °C	2 ⁵ °C	2 ⁴ °C
	LSB	2 ³ °C	2 ² °C	2 ¹ °C	2 ⁰ °C	2 ⁻¹ °C	2 ⁻² °C	0	0
0x05	MSB	T _A ≥ T _{CRIT}	T _A > T _{UPPER}	T _A < T _{LOWER}	SIGN	2 ⁷ °C	2 ⁶ °C	2 ⁵ °C	2 ⁴ °C
	LSB	2 ³ °C	2 ² °C	2 ¹ °C	2 ⁰ °C	2 ⁻¹ °C	2 ⁻² °C	2 ⁻³ °C	2 ⁻⁴ °C
0x06	MSB	0	0	0	0	0	0	0	0
	LSB	0	1	0	1	0	1	0	0
0x07	MSB	0	0	0	0	0	1	0	0
	LSB	0	0	0	0	0	0	0	0
0x08	LSB	0	0	0	0	0	0	1	1

Nous ne prendrons pas en compte les 3 bits de poids fort puisqu'ils nous aideront pas à obtenir la température.

Le 12ème bit nous indique le signe où, bit 12 **Sign:** Sign bit

0 = T_A ≥ 0°C
1 = T_A < 0°C

Suite à ça, il ne me manque plus qu'à récupérer ces 12 autres bits et grâce à la formule présente dans la documentation retrouver la valeur décimale de la température.

pour Temperature T_A ≥ 0°C

$$T_A = (UpperByte \times 2^4 + LowerByte \times 2^{-4})$$

A corriger

Temperature < 0°C

$$T_A = 256 - (UpperByte \times 2^4 + LowerByte \times 2^{-4})$$

Where:

T_A = Ambient Temperature (°C)
UpperByte = T_A bit 15 to bit 8
LowerByte = T_A bit 7 to bit 0

Code de conversion :

```
if((valeurLue[0] & 0x10) == 0x10){  
    //TEMPERATURE NEGATIVE  
    temperaturesys = (float) (256.0 - (((0x0F & valeurLue[0]) *  
16.0) + (valeurLue[1] / 16.0)));  
}  
else{  
    //TEMPERATURE POSITIVE  
    temperaturesys = (float) ((0x0F & valeurLue[0]) * 16.0 +  
valeurLue[1] / 16.0);  
}
```

Les 4 bits de poids forts (alerte
dépassements sont ici mis à 0)

Ici, valeurLue[0] contient les 8 bits de poids fort et valeurLue[1] contient les 8 bits de poids faible. Le if nous permet de différencier le calcul d'une température positive d'une température négative. Quant au calcul, j'applique simplement la formule de la documentation.

Montrer quelques résultats de test...

Vous n'avez pas traité le cas de la température processeur...

5. Test Unitaire

5.1 Classe Ordinateur

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

5.1.1 Description du test

Description :	Test de la communication entre les deux classes en simulant la classe Stockage à l'aide de cout. Il s'agit de récupérer la température du processeur et de communiquer avec les classes Stockage et Ordinateur.
----------------------	---

Scénarios concernés	/ Acquérir la température du système
Description	Test de la communication entre les deux classes en simulant la classe Stockage à l'aide de cout.
Environnement nécessaire	NetBeans, RPI
Situation initiale	/
Classes de tests nécessaires	Stockage et Ordinateur Reboot
Nom du script	TestUOrdinateur.cpp TestUOrdinateur.cpp

5.1.2 Auteur du test

Nom du testeur	Date	Conclusions	Validation
MARYNUS Lucas	08/04/19		oui

5.1.3 Description du script

<i>N°</i>	Traitement	Paramètres en entrée	Résultats attendus
1	Récupération de la mémoire ram libre.		Affichage de la mémoire RAM libre.
2	Récupération de la mémoire µSD disponible.		Affichage de la mémoire µSD disponible.
3	Récupération du pourcentage de la mémoire occupé de la µSD.		Affichage du pourcentage de la mémoire occupée de la µSD.
4	Récupération de la température du processeur		Affichage de la température du processeur.
5			
6			
7			
8			
9			

5.1.4 Problèmes rencontrés

<i>N°</i>	Traitement	Résultats obtenus	Gravité de l'erreur

5.2 Classe Stockage

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

Description :	Test de la récupération de la mémoire avec la classe Ordinateur.
----------------------	--

5.2.1 Description du test

Scénarios concernés	Acquérir le niveau de stockage mémoire
Description	Test de la récupération de la mémoire avec la classe Ordinateur.
Environnement nécessaire	NetBeans, RPI
Situation initiale	
Classes de tests nécessaires	Stockage et Ordinateur
Nom du script	TUStockage.cpp

5.2.2 Auteur du test

Nom du testeur	Date	Conclusions	Validation
MARYNUS Lucas	03/04/2019		oui

5.2.3 Description du script

N°	Traitement	Paramètres en entrée	Résultats attendus
1	Récupération de la mémoire ram disponible de la raspberry.		Affichage de la mémoire RAM.
2	Récupération de la mémoire morte totale μ SD de la raspberry.		Affichage de la mémoire morte.
3	Récupération du pourcentage de la mémoire morte utilisée.		Affichage du pourcentage de la mémoire morte.
4	Récupération de l'occupation de la mémoire RAM		Occupation réelle mémoire RAM
5			
6			
7			
8			
9			

5.2.4 Problèmes identifiés

N°	Traitement	Résultats obtenus	Gravité de l'erreur
	1		
	2		
	3		

5.3 Classe Temperature

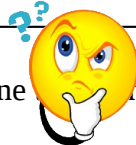
Projet	Nom de l'élément testé	Type de l'élément testé	Version	Incrément
InitCube	Temperature	Classe	1	1

5.3.1 Description du test

Description :	Test de la classe Temperature avec la classe I2C en C++.
----------------------	--

Scénarios concernés	Acquérir la température du système
Description	Test de la classe Temperature avec la classe I2C en C++
Environnement nécessaire	Netbeans, RPI
Situation initiale	
Classes de tests nécessaires	Temperature, I2C.
Nom du script	TII2CTemperature.cpp

5.3.2 Auteur du test

Nom du testeur	Date	Conclusions	Validation
Lucas MARYNUS	01/04/2019	La classe Batterie fonctionne 	Oui

5.2.3 Description du script

<i>N°</i>	Traitement	Paramètres en entrée	Résultats attendus
1	Récupération de la température		Affichage de la température
2	Faire varier la température		Affiche la température
3			
4			
5			
6			
7			
8			
9			

5.2.4 Problèmes identifiés

<i>N°</i>	Traitement	Résultats obtenus	Gravité de l'erreur

6. Test Intégration

6.1 Classe Ordinateur/SegmentVol

Projet	Nom de l'élément testé	Type de l'élément testé	Version	Incrément
InitCube	Ordinateur / SegmentVol	Classe		

6.1.1 Description du test

Description :	Test intégration des classes Ordinateur et SegmentVol
----------------------	---

Scénarios concernés	Communication entre la classe Ordinateur et SegmentVol
Description	Test intégration des classes Ordinateur et SegmentVol
Environnement nécessaire	Netbeans, Raspberry Pi
Situation initiale	
Classes de tests nécessaires	Temperature et SegmentVol
Nom du script	TI OrdinateurSegmentVol.cpp

6.1.2 Auteur du test

Nom du testeur	Date	Conclusions	Validation
MARYNUS Lucas	17/04/2019	La communication entre la classe SegmentVol et la classe Ordinateur est fonctionnelle.	Oui

6.1.3 Description du script

<i>N°</i>	Traitement	Paramètres en entrée	Résultats attendus
1	Récupérer la mémoire USD libre		Affichage de la mémoire USD libre
2	Récupérer la mémoire RAM libre		Affichage de la mémoire RAM libre
3	Récupérer le pourcentage de la mémoire USD utilisée		Affichage du pourcentage de la mémoire USD utilisée
4	Et l'occupation mémoire.		
5			
6			
7			
8			
9			

6.1.4 Problèmes rencontrés

<i>N°</i>	Traitement	Résultats obtenus	Gravité de l'erreur

6.2 Classe Ordinateur/Stockage

Projet	Nom de l'élément testé	Type de l'élément testé	Version	Incrément
InitCube	Température SegmentVol	/ Classe		

6.1.1 Description du test

Description :	Test intégration des classes Temperature et SegmentVol
----------------------	--

Scénarios concernés	Communication entre la classe Temperature et SegmentVol
Description	Test intégration des classes Temperature et SegmentVol
Environnement nécessaire	NetBeans, Raspberry Pi
Situation initiale	
Classes de tests nécessaires	Temperature et SegmentVol
Nom du script	T TemperatureSegmentVol.cpp

6.1.2 Auteur du test

Nom du testeur	Date	Conclusions	Validation
MARYNUS Lucas	17/04/2019	La communication entre la classe SegmentVol et la classe Temperature est fonctionnelle.	Oui

6.1.3 Description du script

N°	Traitement	Paramètres en entrée	Résultats attendus
1	Récupérer la température		Affichage des informations de la température du système.
2			
3			
4			
5			
6			
7			
8			
9			

6.1.4 Problèmes rencontrés

N°	Traitement	Résultats obtenus	Gravité de l'erreur

7- Bilan

Pour conclure, ce projet m'a permis d'approfondir mes connaissances informatiques. J'ai pu perfectionner le codage en c++ ainsi que le travail de groupe. En effet, avoir un planning à respecter m'a poussé à travailler dans de réelles conditions et de mener à bien ce projet.

J'ai pu constater^{er} que d'être plongé dans un même sujet pendant plus de deux mois m'a facilité mon envie d'y travailler, de plus les tâches n'étaient pas tous les jours les mêmes.