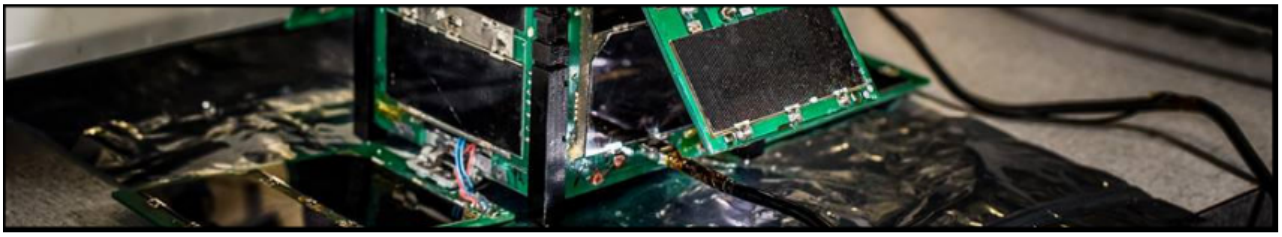
AR-020319-001

[DOCUMENT TYPE] – IPSA ONE 2019
ENGLISH



TITLE OF THE MISSION

DOCUMENT TITLE

[MONTH] [YEAR]

Written by : Lucas Bellegy	Date and signature :
Checked by :	Date and signature :

Orbital **N**ano **E**xperiments
Association Law 1901 hosted at Institut Polytechnique des Sciences Avancées
63 boulevard de Brandebourg – 94200 Ivry-Sur-Seine
Mail : ipsa-one@ipsa.fr

CHANGE LOG

Ed.	Rev.	Date	Modifications
1	0	13/04/2020	Document creation

Contents

0.1 Les types de liaisons

0.1.1 UART

La liaison UART, ou liaison série, est une liaison dite asynchrone avec laquelle seulement 2 éléments peuvent communiquer entre eux. C'est-à-dire qu'il n'y a pas d'horloge qui définit la fréquence à laquelle sont envoyées les données. Il y a 2 broches, RX et TX, l'une est pour la réception d'information et l'autre pour l'envoi d'information. Comme il n'y a pas d'horloge pour définir la fréquence de transmission, on doit, pour tout échange de données, paramétrer un nombre de baud(bits/sec).

0.1.2 I2C

La liaison I2C est une liaison synchrone qui permet la communication de théoriquement autant d'appareils qu'on veut. En réalité, une adresse est implémentée dans chaque élément communiquant sur la liaison I2C et ensuite le maître (Master) choisira l'adresse du composant esclave (Slave) avec lequel il veut communiquer. La communication en I2C se fait sur 2 broches, SCL, l'horloge et SDA, la où les données sont échangées.

0.2 Les Processeurs

0.2.1 Architecture d'un processeur

Un processeur est un regroupement de beaucoup de transistors (vraiment beaucoup) qui permettent de créer des portes logiques pour ensuite effectuer différentes actions. Ces actions vont pouvoir être exécutées un certain nombre de fois dans une seconde, c'est ce qu'on appelle la fréquence du processeur. Associé à ces transistors, il y a de la RAM (Random Access Memory) qui permet pendant l'exécution d'un programme l'enregistrement de plusieurs donnée nécessaires à la suite de l'exécution du programme. La RAM s'efface à chaque fois que le processeur est éteint. Il y a aussi la ROM (Read Only Memory), une mémoire qui ne s'efface pas et sur laquelle sont enregistrées les différentes instructions pour l'exécution du programme.

0.2.2 Les différents processeurs

La carte ARM mbed LPC1768 est une carte programmable de petite taille et qui consomme peu, ce qui permet donc d'être très facilement intégrée à un système embarqué. Elle possède 25 broches programmable plus les broches d'alimentions. L'autre processeur sur lequel nous avons travaillé est le SAMV71Q21 d'Atmel. Cette carte est plus développée que celle de mbed, elle dispose de nombreux ports, capteurs et types de mémoires différents pour faire des tests. On a accès à des Il est conseillé d'utiliser des gants pour la manipuler. Nous avons aussi différents capteurs (la plupart fonctionnant en UART) que nous pouvons utiliser pour faire des programmes tests.

0.2.3 Le mapping

Un mapping de la ARM mbed LPC1768 et les différents capteurs est en cours mais pas encore fini.

0.2.4 Les environnements de développement

L'environnement de développement sur mbed se fait entièrement en ligne et le développement se fait en c++ avec les bibliothèques développées par mbed. Cependant, cet environnement de développement ne nous laisse pas une entière liberté pour regarder dans les bibliothèques ou voir les erreurs exactes. Heureusement il est aussi possible de développer avec visual code ce qui permet de régler les problèmes précédemment cités. Pour le MCU SAMV71, on doit utiliser le logiciel Atmel Studio. L'initialisation des différentes bibliothèques (composants, horloges, fonctions, variables) disponibles est faite automatiquement, ce qui permet de simplement écrire son code dans un fichier « main.c » (le langage est soit le C soit le C++). On peut aussi utiliser le site Atmel Start pour générer des codes spécifiques avec les bibliothèques des différents capteurs et autres drivers. On a aussi accès à des codes exemples.

0.3 Programmation

0.3.1 Principe

Une machine ne communique à la base qu'en binaire et ce serait une tache infiniment compliquée que d'écrire des programmes aussi complexes qu'aujourd'hui en binaire. C'est pour ça qu'on utilise des langages de programmation. Ainsi, on peut programmer des taches avec un langage qui nous est bien plus facile à comprendre et ce code sera ensuite compilé pour qu'il puisse être compris par la machine.

0.3.2 Compilateur

Un compilateur est un programme qui va prendre en entrée un code source écrit dans le langage informatique de notre choix, ici, le c++, et va le transformer en un langage objet, un langage exécutable par la machine. Pour faire ceci, Le compilateur va découper le code en ce qu'on appelle des tokens puis, il va analyser ces tokens pour identifier la structure syntaxique du programme (comment il est écrit, y-a-t-il des erreurs dans l'écriture), il va ensuite vérifier s'il n'y a pas d'autre type d'erreur comme la mauvaise déclaration d'une variable ou d'une fonction, il va ensuite générer un code intermédiaire pour appliquer des optimisations et enfin créer le code objet final.

0.3.3 Programmation de la communication avec l'antenne

L'antenne fonctionne exactement comme une simple liaison UART, il suffit donc d'utiliser la bibliothèque mbed pour utiliser la liaison UART.

0.4 Feedback sur la phase I

- Récupérer le plus de documents possibles - Ajouter la liaison CAN à la documentation - Faire des recherches supplémentaires sur le fonctionnement de la mémoire, les périphériques, la DMA, MMU - Créer des tutos pour l'utilisation des logiciels (Atmel Studio)