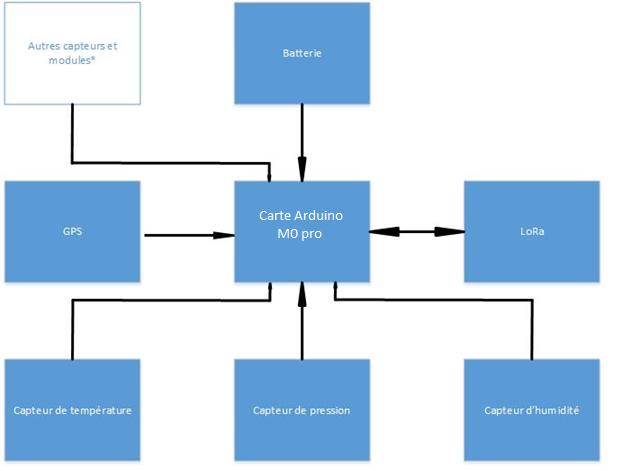
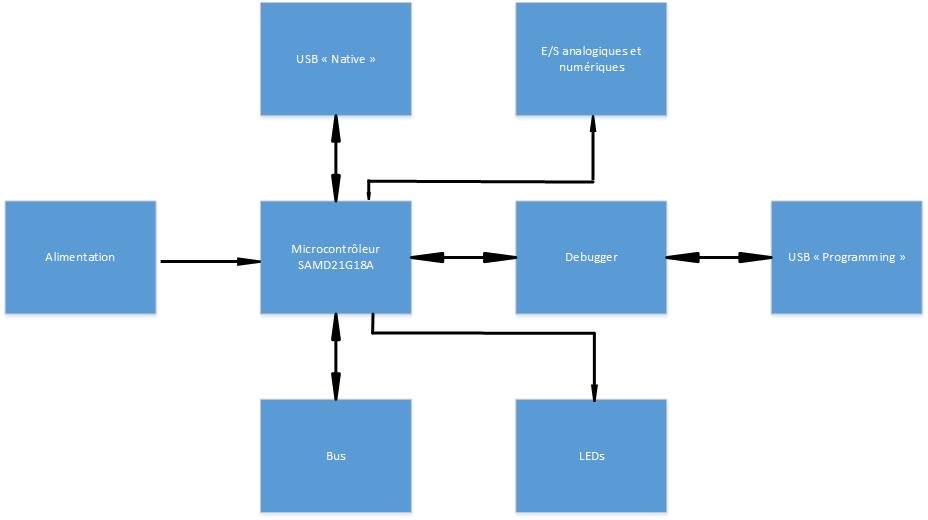
BRAISE

L’idée est de réaliser un réseau de balises de mesure dosimétrique et multi-capteurs, fonctionnant de manière autonome à basse consommation, et transmettant les données par ondes radio jusqu'à un concentrateur. La géolocalisation de ces nœuds est indispensable pour l'exploitation des données. La miniaturisation, le coût et la robustesse aux conditions climatiques extrêmes (hiver/été, variations de température et d'humidité) sont des enjeux majeurs pour les possibilités de déploiement.

* Synoptique du système complet :



* Synoptique carte Arduino M0 pro :

**

* Alimentation : ce bloc est composé d’un connecteur jack, d’un premier régulateur 7V-12V vers 5V puis d’un second 5V vers 3.3V. Il permet de générer la tension de fonctionnement du microcontrôleur.
* Microcontrôleur SAMD21G18A : faisant partie de la famille SAM D Cortex M0+ d’Atmel, ce microcontrôleur d’architecture ARM est dédié aux applications basse consommation. On y retrouve différents modules tels qu’un ADC 12 bits, un DAC 10 bits, une RTC (Real Time Clock), une PWM.
* Bus : ce bloc comprend les connecteurs par lesquels les bus de communication transitent, principalement I2C et SPI.
* LEDs : un ensemble de cinq leds dont quatre servent de témoin pour la mise sous tension, les transmissions (RX et TX) et le mode debug. La cinquième est reliée à une sortie du microcontrôleur, elle est gérée par l’utilisateur.
* E/S analogiques et numériques : ce sont les connecteurs périphériques de la carte. On y retrouve les 20 entrées/sorties, les sorties des régulateurs (5V et 3.3V), les entrées de l’ADC ainsi que la masse.
* Debugger : il s’agit du composant Atmel Embedded Debugger (EDBG) servant à programmer la carte, à accéder à la mémoire du microcontrôleur et à exécuter le programme en mode pas à pas via le logiciel Atmel Studio. On peut par exemple visualiser le contenu des registres via le logiciel AtmelStudio.
* USB « Programming » : un connecteur au format micro-USB utilisé pour implanter le programme via le debugger.
* USB « Native » : un second connecteur micro-USB directement connecté sur le microcontrôleur. Il est utilisé pour interfacer des périphériques (clavier par exemple) via une communication série et peu également utilisé pour programmer le microcontrôleur.
* Capteurs et bus de communication :
* Le capteur de pression, température et d’humidité nommé BME280 est connecté à la carte Arduino par l’intermédiaire d’un bus I2C (voir rapport de stage d’Anthony)
* Le module GPS utilise le bus SPI.
* Le module de communication sans fil « Lora » utilise le bus Série TTL.
* Tests de faisabilité :

Divers programmes permettent la mise en œuvre et le test des différents éléments gravitant autour de la carte Arduino (voir le readme dans l’espace de travail) comme par exemple les programmes nommés :

* ***Bme280***: Récupère les données provenant du capteur BME280 toutes les 5s environ
* ***bme280\_json*** : Récupère les données provenant du capteur BME280 ainsi que la date et l'heure et les encode au format Json
* ***gps\_simple*** : Acquisition gps, transmission lora via le shield
* ***lora\_config\_noshield*** : Configuration du lora
* …

La mise en situation du système en condition proche de la réalité se fait à l’aide de 2 plateformes :

* 1 carte nommée « master » équipée d’1 Arduino M0 pro et d’un module Lora
* 1 carte nommée « slave » équipée d’1 Arduino M0 pro, d’un module Lora, d’un GPS et du capteur environemental.

Les programmes à utiliser sont :

* Gps\_bme\_simple pour la carte « slave » (Acquisition capteurs, json, transmission)
* Gps\_bme\_master\_simple pour la carte « master » (Récupère les info gps et bme au format json et les décode)

Ou

* gps\_bme\_simple\_ack pour la carte « slave » (Acquisition capteurs, json, transmission avec acknowledge)
* gps\_bme\_master\_simple\_ack pour la carte « master » (Récupère les info gps et bme au format json et les décode avec acknowledge)
* Aide sur la compréhension du module Lora SX1272 :

<https://www.cooking-hacks.com/documentation/tutorials/extreme-range-lora-sx1272-module-shield-arduino-raspberry-pi-intel-galileo>