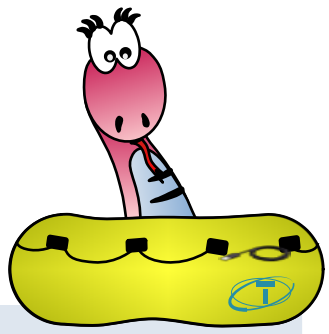
Architecture des fichiers de code micropython



Programme principal

C'est la base de l'architecture de fonctionnement de la Boopy.
Il respecte les éléments de l'arbre programmatique (voir la fiche idoine).
Le fichier *boot.py* contient les instructions executées à la mise sous tension du microcontrôleur accueillant l'interpréteur MicroPython

main.py

boot.py

Interface des capteurs

Ces codes permettent d'ajuster les données issues des capteurs aux besoins des expériences de la Boopy.
C'est dans ces codes sources que vous pouvez placer des algorithms en fonction des besoins de votre projet.

boopy_energy.py

boopy_extern.py

boopy_light.py

boopy_mpu.py

boopy_ptu.py

boopy_temp.py

Par exemple, dans le but d'évaluer l'agitation de la mer, nous allons retenir uniquement les accélérations maximales de la centrale inertielle pendant une séquence de mesures de 20s. (Voir fiche 6.1 : "Etat de la mer")
Ou réaliser un calcul sur une période de quelques minutes pour déterminer la hauteur des vagues en fonction des mesures de l'accélération verticale de la bouée.
Plus simplement, réaliser une moyenne de la température avec des mesures prises pendant quelques secondes.

Bibliothèques des capteurs

Permet de faire fonctionner correctement les composants/capteurs utilisés avec la Boopy.
Le code de ces bibliothèques respecte les procédures fournies dans les datasheets de ces composants.
En principe, une fois écrites et validées, ces bibliothèques sont rarement modifiées, sauf si une erreur est découverte, bien sûr.

INA219.py	Capteur de tension et de courant électrique
TSL2591.py	Capteur de lumière visible et infrarouge
MPU6050.py	Centrale inertielle 6 axes
BME280.py	Capteur de pression, température et humidité
MCP9808.py	Capteur de température

Fonctionnement des capteurs

Tous ces capteurs ont été choisis pour l'intérêt du paramètre qu'ils restituent, mais aussi parce qu'ils communiquent leurs informations de la même manière. Cela simplifie l'écriture des bibliothèques et permet leur échange et évolution plus rapidement. Ils utilisent le standard i2c qui est présenté dans les fiches de notre malle de formation "Boopy".
Nous fournissons également les *datasheets* des capteurs avec le code source de ces bibliothèques.

Bibliothèques de périphériques

Ces codes permettent de faire fonctionner correctement les composants de la Boopy. Ils sont similaires aux bibliothèques de capteurs.

RTC.py	Real Time Clock : horloge temps réel qui comporte une alarme permettant de réveiller la carte pour le passage de l'heure.	La RTC communique par bus SPI avec le microcontroleur et le micropython.
GPS rx interrupt GPS.py	Le GPS produit une trame RMC toute les secondes qui nous fournit l'heure, la date, la latitude et la longitude.	Le GPS et le modem Iridium communiquent par liaison serie (UART). Nous utilisons la bibliothèque *uasyncio* pour gérer de manière optimum les arrivées de données de ces périphériques.
IRIDIUM rx interrupt iridium.py	Le modem Iridium permet de transmettre les données sous la forme de trame de 49 octets. La description de la trame est faite dans le fichier *BOOPY_Frame.ods*.	

Bibliothèques intégrées dans Micropython

La plupart des codes sources utilisent des bibliothèques qui sont implémentées en 'dur' dans l'interpréteur micropython du microcontrôleur de la Boopy. En voici une rapide description. A noter que certaines bibliothèques sont directement inspirées de celles du Python classique de nos PC. Elles différent par leur code qui a été optimisé pour fonctionner sur microcontroleur et il est possible que toutes les fonctions ne soient pas reportées en Micropython. On les distingue par leur nomination précédée d'un 'u' de micro (µ).

utime : Contient les fonctions de timing, par exemple utime.sleep_ms(100), attend pendant 100ms
ustruct : Contient la fonction pack() de conversion d'entiers en binaire de plusieurs octets les 'bytearray()' pour les transmettre par Iridium
uasyncio : Contient les fonctions de gestion asynchrone qui permet de gérer les réceptions de données envoyées par le GPS ou le modem Iridium.
pyb : Contient les fonctions de gestion des bus de données (I2C, SPI, UART) et autres fonctions de gestion des LED, des ports d'entrée/sortie (GPIO)...

Pour plus d'information, consultez la documentation officielle : MicroPython v1.18