Projet Carte de télémétrie Open source


# Initiation

# Exemple de réalisation.

Lien vers le projet original :
https://github.com/openXsensor/openXsensor/tree/master/openXsensor
Discussion sur ce projet :
http://openrcforums.com/forum/viewforum.php?f=86
Le Wiki
https://github.com/openXsensor/openXsensor/wiki

# Suivi des modifications

| Indice de révision | Nature des modifications | date |
|---|---|---|
| OR | Version originale | 2021 05 07 |
| | | |
| | | |
| | | |
| | | |
| | | |

# Sommaire

# Présentation du projet

Le projet est basé sur un projet Open source qui se nomme OpenXsensor oXs

Le projet consiste à créer une carte de gestion de la télémétrie de nos modèles à l'aide de capteur facilement disponible pour un cout très raisonnable.
Cette carte est réalisée à base de carte Arduino pro mini ou Nano.
Nous pouvons donc avoir de la télémétrie pour les protocoles Frsky Hub, Frsky SPORT, Hott, JETI protocole EX et Multipex en fonction du type de radiocommande utilisée.
Ce projet est aussi capable de gérer un séquenceur ainsi qu'un module long range pour retrouver un modèle.

## Introduction

Basé sur une plate-forme Arduino, il se connecte à différents capteurs et transmet les mesures via plusieurs protocoles de télémétrie.

Il est compatible avec :

- Les récepteurs Multiplex
- Les récepteurs Frsky : *série D* (protocole HUB) et *série X* (protocole Smart Port appelé SPORT).
- Les récepteurs Graupner (protocole Hott)
- Les récepteurs Jeti (protocole EX uniquement)

## Fonctionnalités

- Altimètre / Variomètre
  - Utilise normalement un capteur MS5611 (module GY-63 ou GY-86), peut aussi utiliser un BMP085, un BMP180 ou un capteur BMP280 comme premier capteur
  - Un deuxième capteur barométrique peut être ajouté, uniquement un MS5611
  - Possibilité de changer la sensibilité variomètre en vol avec une voie de l'émetteur.
- Vitesse de l'air
  - Utilise normalement un capteur 4525DO-DS5AI001DP, peut aussi utiliser ADS1115 + MPXV7002DP. Ce capteur peut également être utilisé par les oXs pour compenser les mesures de vitesse verticale (dTE)
- GPS
  - Utilise un GPS U-Blox (NEO6/NEO7/NEO8)
- Accéléromètre/gyroscope
  - Peut être connecté à un capteur MPU6050 (par exemple GY-86 module)
  - Peut calculer la vitesse verticale basée sur l'accélération + baromètre (réaction plus rapide du variomètre.)
- Jusqu'à 6 mesures de tension avec Arduino
  - Mesures de tension de la batterie
  - Mesures des cellules de la batterie LiPo
- Utilise un module ADS1115 pour étendre les fonctionnalités du module

- Capteur RPM
- Capteur de courant (p. ex. ACS712, ACS754, ACS758)
- Capteur de débit (fournit le flux réel, la capacité restante du réservoir en ml et %)
- Peut utiliser n'importe quelle tension de sortie de capteur
    - Conversion de Volt à d'autres unités (p. ex. au degré pour capteur de température)
- Peut calculer le rapport de planeur pour améliorer la configuration du planeur pendant le vol
- Peut générer plusieurs séquences de signal (p. ex. pour séquenceur de lumière)
- Peut devenir un localisateur (pour retrouver un modèle perdu)
    - Utilise un module LORA (SX1276/RFM95) pour obtenir une transmission à longue portée
    - Utilise un deuxième Arduino pro mini (ou Arduino nano) avec un écran et un module LORA
    - afficher la position GPS et la force du signal (comme goniomètre)

**Remarque :** La configuration se fait en éditant manuellement les fichiers oXs_config_basic.h et dans certains cas oXs_config_advanced.h Tous les détails sur les exigences et la façon de configurer les oX sont donnés dans le fichier oXs_config_description.h Certaines explications sont données dans wiki (mais ne sont pas à 100% à jour). Il est fortement recommandé de lire oXs_config_description.h à p rtir de l'IDE Arduino ou le fichier

**Remarque** : La dernière version du projet inclut un configurateur oXs. Il permet :

- pour générer les fichiers config à l'aide d'une interface utilisateur graphique.
- pour enregistrer/recharger plusieurs ensembles de paramètres.
- pour afficher la documentation (contenu du fichier oXs_description.h Il faut que Python3.7 soit installé. Pourtant, pour Windows 10, il y a paquet précompilé qui ne nécessite pas d'installer Python. Le configurateur prend en charge la plupart (mais pas tous les paramètres de configuration oXs) de sorte que, dans certains cas, vous avez encore à modifier les fichiers config manuellement.

Il y a certaines restrictions sur les fonctionnalités en fonction de la marque de radio et du protocole utilisé.

## Le matériel

A l'aide d'une carte Arduino pro mini 5v 16mhz, oXs peut gérer un grand nombre de capteurs, comme par exemple

- GPS U-blox type Neo 6,7 ou 8. Le Betian BETIAN 220 fonctionne parfaitement
- Capteur barométrique
- Accéléromètre MPU6050
- Capteur de rotation RPM
- Capteur de débit pour la mesure de la consommation des modèles thermiques
- Sonde Pitot
- Sonde de température

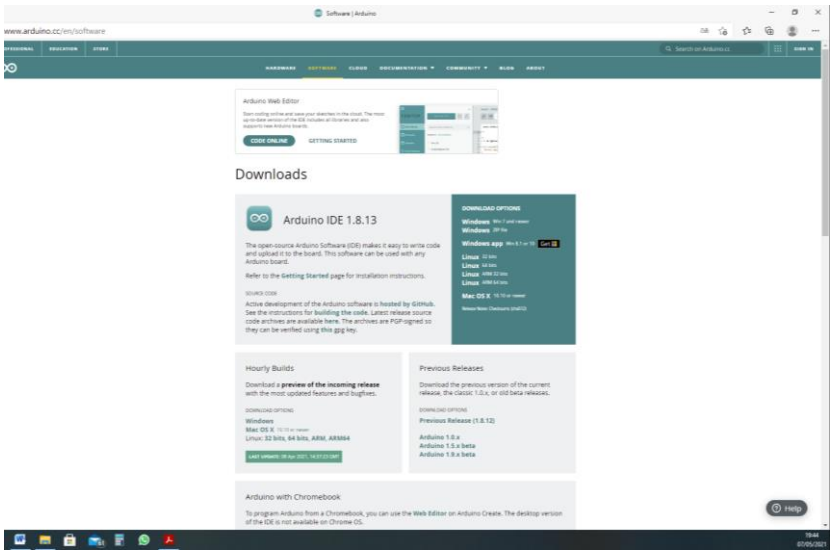
Programmateur FTDI ou utilisation d'une carte Arduino nano qui est plus grande mais qui a un connecteur USB

## Raccordement

- Connections de la sortie vers le récepteur **(Broche 4, Vcc, Gnd)**
- Connections de la sortie PPM (optionnel) **(Broche 2, Vcc, Gnd)**
- Connections de 1 ou 2 capteurs barométriques, IMU 6050 (capteur accéléromètre/gyroscope) (optionnel) et HMC5883 (magnétomètre) (optionnel) capteur vitesse air type sonde Pitot (optionnel) **(Bus I2C Broche A4 et A5, Vcc et Gnd)**
- Connections pour mesure des tensions **(Broche analogique A0 à A7, sauf A4 et A5 si utilisation I2C,)** Ajouter un pont diviseur sur chaque entrée Ax, calculé pour chaque tension pour ne pas dépasser Vcc
- Connections du capteur de mesure de courant (optionnel). **(Broche analogique A0 à A7, sauf A4 et A5 si utilisation I2C+ Gnd)**
- GPS (optionnel) **(Broche Rx et broche 6, Vcc et Gnd)** mettre résistance de 10 à 22k entre broche 6 Arduino et broche Rx du GPS
- Capteur de débit (optionnel) **(Broche 9, Vcc et Gnd)**
- Séquenceur (ON/OFF) pour certaines sorties digitales (Exemple : contrôle de lumières) **(Broche 8 à 13+ Gnd)**

# Installation IDE Arduino

IDE Arduino à télécharger sur le site : https://www.arduino.cc/en/main/software
L'IDE peut aussi être chargé depuis le Windows store en recherchant « Arduino » dans le moteur de recherche.




Pour plus de simplicité, laisser IDE Arduino faire ne cherchez pas à l'installer ailleurs que sur le chemin d'accès préconisé.
Il peut aussi être installez en mode portatif sur clé USB ou un support externe.

Si les transferts de données ne se font pas avec succès, les problèmes proviennent bien souvent des autorisations d'accès (mode admin) sur tel ou tel répertoire ou du pare-feu.

## Mise à jour de bibliothèque

Pas utile pour ce projet

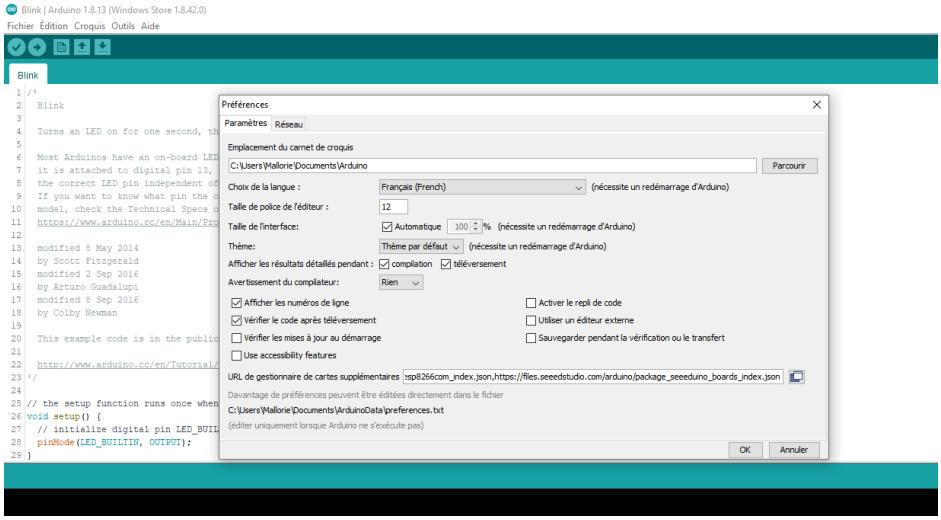
## Installation des bibliothèques

Pas utile pour ce projet

## Préférence

- Une fois l'IDE Arduino chargé il est intéressant de paramétrer ses préférences.
- Dans « Fichier », « Préférences », sélectionner les options suivantes
    - Afficher les résultats détaillés pendant la compilation et le téléversement
        - Cette option peut être utile pour aider à identifier un problème en cas de besoin.
    - L'affichage des numéros de ligne peut être un plus en cas de problème
    - Décocher l'option Sauvegarde pendant la vérification ou le transfert

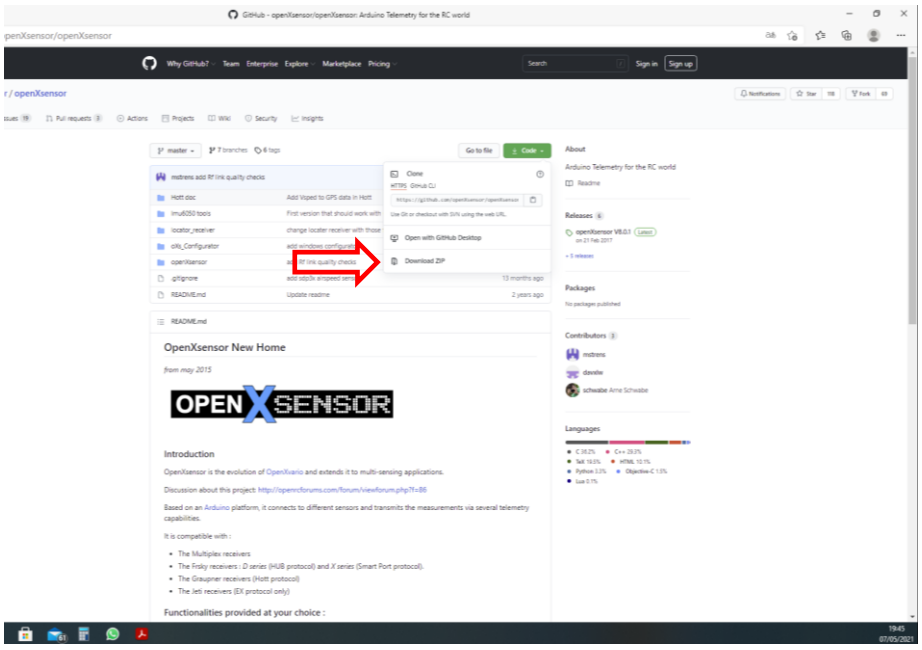
- La sauvegarde du projet ne se fera que sur une action volontaire au lieu de se faire dès qu'un téléversement est lancé. Cela permet de faire des essais sans sauvegarder le projet.




# Installation du logiciel OpenXsensor

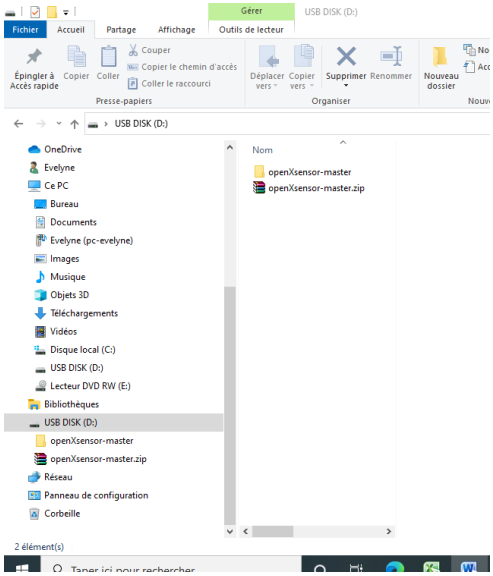
## Installation du logiciel

Télécharger openXsensor : https://github.com/openXsensor/openXsensor




Une fois l'archive téléchargée, extraire les fichiers dans le répertoire de votre choix. Eviter les chemins trop longs pour sauvegarder le projet.

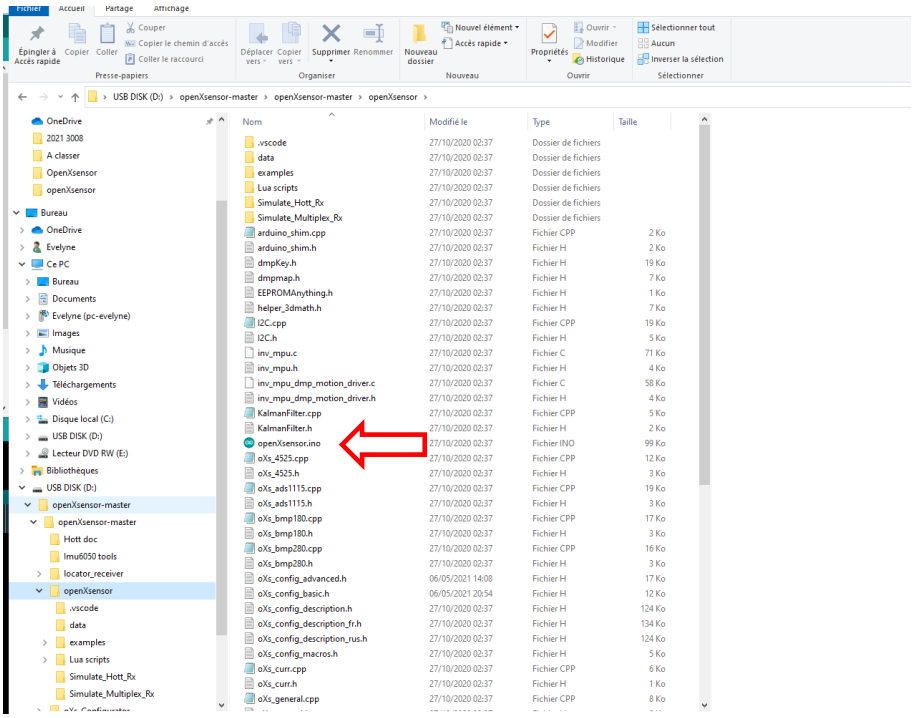
Ici openXsensor est chargé sur une clé USB, mais vous pouvez le mettre dans n'importe quel répertoire de votre disque dur si vous le souhaitez.




Cliquez sur

- openXsensor-master
  - à nouveau sur openXsensor-master
    - et sur openXsensor

Vous arrivez sur l'arborescence ci-dessous
Double-cliquez sur le fichier openxsensor.ino




Ca y est l'IDE Arduino vient de s'ouvrir avec votre projet.

**Attention**
Vous aurez une configuration pour chaque montage différent, il est donc nécessaire de prévoir une organisation dès le début pour limiter le risque d'erreur par la suite. Cette organisation peut se faire pour le projet complet ou seulement pour les fichiers de configuration qui se nomment oXs_config_basic et oXs_config_advance.

### Paramétrage

Pour paramétrer notre projet en fonction des besoins, il y a deux2 fichiers à mettre à jour.
Il s'agit des fichiers oXs_config_basic et oXs_config_advanced

Nous allons paramétrer le projet au travers d'un exemple de réalisation.

Une présentation détaillée des possibilités et du paramétrage sont décrit dans le fichier oXs_config_description_fr 2.docx

## Exemple de réalisation

### Description

Ensemble Baro, GPS et mesure de tension d'une batterie (sans le détail de chaque élément) avec un GPS, un capteur barométrique et une mesure totale de la batterie d'alimentation.
Ce montage sera paramétré pour une radio Frsky en sport.

### Matériel

- Carte Arduino pro mini 5v 16Mhz
  - Vous pouvez trouver cette carte par exemple ICI

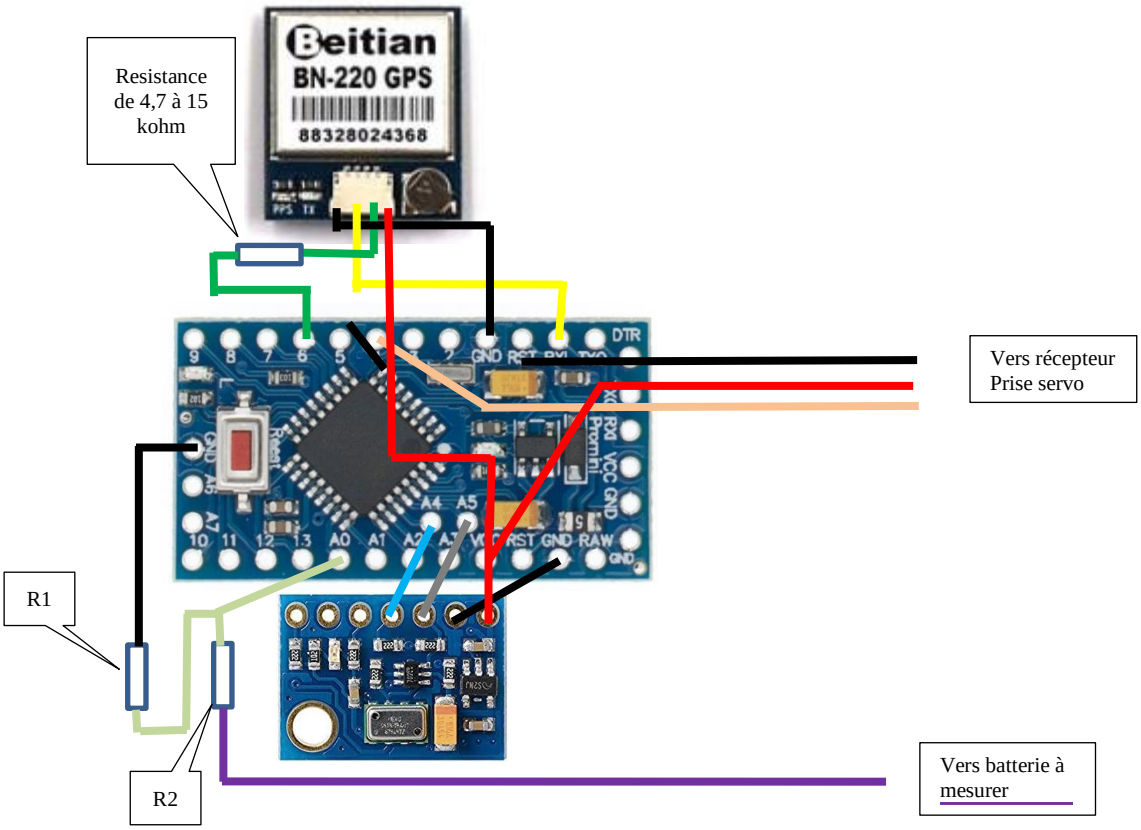
- GPS BETIAN BN220
  - Vous pouvez trouver cette carte par exemple ICI

- Carte avec capteur Barométrique
  - Vous pouvez trouver cette carte par exemple ICI

- Une prise type servo
- 1 résistance de 15k
- 2 résistances R1 et R2 pour le pont diviseur qui sert à la mesure de tension (voir valeur dans chapitre câblage.
- Du fil, de la soudure, de la gaine thermo, etc…

## Câblage

Version Avec GPS BN220, Baro MS5611 et mesure de tension




**Attention :** L'alimentation du module doit se faire sur la broche Vcc si le récepteur est alimenté en 5v et sur la broche Raw si le récepteur est alimenté par une tension supérieure.

Le pont diviseur formé par R1 et R2 devra être calculé en fonction de la tension à mesurer pour ne jamais dépasser la tension de référence paramétrée.

Exemple :
Pour mesurer une batterie 4S, soit 4x4.2v maxi = 16.8v, si on prend comme référence la tension interne de l'Arduino qui est de 1.1v (voir paramétrage)
Sachant qu'il ne faut pas dépasser 1.1v sur l'entrée Arduino et même 0.9v avec une petite marge de sécurité le ratio entre les résistances sera le même que celui entre les tensions.

Aller un peu de math. On aura donc :
Tension aux bornes de l'Arduino VS / tension batterie VE =R1+R2/R1
VE / VS =R2+R1 / R1
D'où R1 = (Vs*R2) / (Ve-Vs)

Par exemple, Si on prend 100khom pour R2 on aura :
R1= (0.9*100) / (16.8 -0.9) = 5.66kohm, on prendra la valeur normalisée en dessous soit 5.6kohm

On aura donc 100kohm pour R2 et 5.6kohm pour R1 si on veut mesurer la tension d'une batterie 4S.

**Nota :** Il est tout à fait possible de mesurer la tension d'une batterie de moins de 4S (mais jamais plus) avec ce pont diviseur, mais un calcul adapté permet d'obtenir toujours la précision maximum.

**Remarque :** Les valeurs de R1 et R2 ne sont pas critiques et peuvent varier dans de très grandes proportions, c'est le ratio qui permet de protéger l'Arduino qui ne supporte pas des tensions supérieures à la tension de référence programmée.

Il est préférable que la somme de R1 et R2 soit au moins de plusieurs kOhms afin de ne pas trop consommer inutilement.

## Paramétrage

### Généralité

Pour le paramétrage, je vous rappelle que deux fichiers sont à paramétrer en fonction de vos besoins et surtout de votre câblage.
Il s'agit des fichiers oXs_config_basic et oXs_config_advance.

### Paramétrage : fichier oXs_config_basic.h

Chapitre 1 :
Sélectionnez le type de protocole Jeti, Hott, etc. et copiez-le à la suite de define PROTOCOL dans notre cas FRSKY_SPORT

```
// --------- 1 - Telemetry protocol ---------
#define PROTOCOL FRSKY_SPORT // select between FRSKY_SPORT, FRSKY_HUB, FRSKY_SPORT_HUB, MULTIPLEX, HOTT, JETI
```

Chapitre 4 :
Sélectionnez le type de BARO utilisé dans notre cas le MS5611

```
// --------- 4 - Vario settings ---------
// ***** 4.1 - Connecting 1 or 2 barometric sensor(s) *****
#define FIRST_BARO_SENSOR_USE MS5611 // select between NO_BARO, MS5611, GY86, BMP085, BMP180, GY87, BMP280
```

Ce chapitre permet d'indiquer à oXs quel capteur est utilisé pour calculer la vitesse verticale.

```
/ ***** 4.2 - Type of Vspeed to transmit  *****
#define VSPEED_SOURCE FIRST_BARO    // select between FIRST_BARO, BARO_AND_IMU, SECOND_BARO, AVERAGE_FIRST_SECOND, AIRSPEED_COMPENSATED or PPM_SELECTION
```

Chapitre 6 :
Ce chapitre permet d'indiquer à oXs que l'on veut mesurer au moins une tension. Il faudra aussi paramétrer oXs_config_advance
***** 6.2 - Voltage parameters *****
#define ARDUINO_MEASURES_VOLTAGES **YES** // select between YES, NO (When NO, following line is discarded)

******Chapitre 6.3******
#define NUMBEROFCELLS **0** // on ne veut pas mesurer la tension de chaque cellule de batterie

Chapitre 9 :
// --------- 9 - GPS ----------
#define A_GPS_IS_CONNECTED **YES** // select between YES, NO

Les valeurs des chapitres suivant doivent être à NO : Chapitre 6.5, 6.6, 7, 8, 10, 11, 12, 13

Paramétrage : fichier oXs_config_advance.h

Définir le pin de communication du Smartport pin 4 dans notre cas :
// ****** 1.1 - Pin connected to Rx ********
#define PIN_SERIALTX **4** // The pin which transmits the serial data to the telemetry receiver, Usually pin 4 (otherwise pin 2)

// --------- 6 - Voltages & Current sensor settings ---------
// ***** 6.1 - Voltage Reference to measure voltages and current *****
//#define USE_INTERNAL_REFERENCE // uncomment this line if you use 1.1 volt internal reference instead of Vcc (voltage divider mst be used to reduce voltages to 1.1 volt max)
Ce paramètre permet d'indiquer à l'Arduino la tension de référence pour la mesure de tension.
Décommenter #define USE_INTERNAL_REFERENCE et commenter les autres lignes avec des // dans le chapitre 6.1

// ***** 6.2 - Voltage parameters *****
// Chaque ligne contient 6 valeurs, la premier pour Volt1, la deuxième pour Volt 2, Etc. jusqu'à 6.
Nous avons branché notre mesure de tension sur la broche A0, ce qui correspond Volt1. Il faudra donc rentrer les valeurs du pont diviseur pour volt1 afin que oXs indique une valeur représentative.
#define PIN_VOLTAGE **0, 8, 8, 8 ,8 ,8** //Mettre 8 pour les mesures qui ne sont pas utilisées.
#define RESISTOR_TO_GROUND **5.6, 0, 0, 0, 0 , 0** // mettre 0 si on utilise pas de pont diviseur.
#define RESISTOR_TO_VOLTAGE **100,0, 0, 0 , 0, 0** // mettre 0 si on utilise pas de pont diviseur.
#define OFFSET_VOLTAGE **0, 0 ,0 ,0 ,0 ,0**

#define SCALE_VOLTAGE **1.0 , 1.0 , 1.0 , 1.0 , 1.0 , 1.0**

Les lignes Offset et échelle pourront être ajustées en fonction des mesures réalisées. (Voir rubrique étalonnage des capteurs) étalonnage)

Chapitre 9 Programmation du GPS
Il faut enlever les commentaires // de la ligne #define GPS_SPEED_3D, le taux de rafraichissement est mis sur 5 dans notre cas, c'est amplement suffisant. Plus on demande au GPS d'être rapide, plus il va chauffer.
// --------- 9 - GPS -----------
//#define GPS_SPEED_IN_KMH
#define GPS_SPEED_**3D** //A tester la vitesse 3D étant certainement moins précise que la 2D
#define GPS_REFRESH_RATE **5** // On obtient aussi des résultats tout à fait satisfaisant en laissant le GPS a 1Hz.

Une fois ces modifications terminées il faut téléverser le programme dans l'Arduino.

**Nota :** L'ensemble des modifications précédentes peuvent aussi être réalisées en ouvrant les fichiers oXs_config_basic et oXs_advanced qui se trouvent dans le répertoire OpenXsensor dans un éditeur de texte type bloc-notes ou Word, il faudra être vigilant lors de l'enregistrement et vérifier que l'extension du fichier soit la bonne et repasser par l'IDE Arduino pour programmer la carte

## Téléversement du programme dans l'Arduino

### Matériel

Pour téléverser le soft dans la carte Arduino, vous aurez besoin d'un programmateur FTDI

Voici un exemple de programmateur FTDI.

Vous trouverez ICI un modèle qui fonctionne bien et qui a un câblage qui permet de se connecter fil à fil avec la carte Arduino pro mini.

Raccordement programmateur
1. FTDI USB-to-TTL: TX -> Arduino Mini Pro RX
2. FTDI USB-to-TTL: RX -> Arduino Mini Pro TX
3. FTDI USB-to-TTL: GND -> Arduino Mini Pro GND
4. FTDI USB-to-TTL: +5Vcc -> Arduino Mini Pro VCC
5. FTDI USB-to-TTL: DTR -> Arduino Mini Pro Reset
6. FTDI USB-to-TTL: CTS (non connecté)

**Attention :**
Vérifier que votre programmateur est bien reconnu par votre ordinateur lors de son branchement, si ce n'est pas le cas, il faudra installer les driver correspondants à la puce qui est sur votre modèle. Une puce très souvent utilisé pour ce genre de programmateur est la CH340.

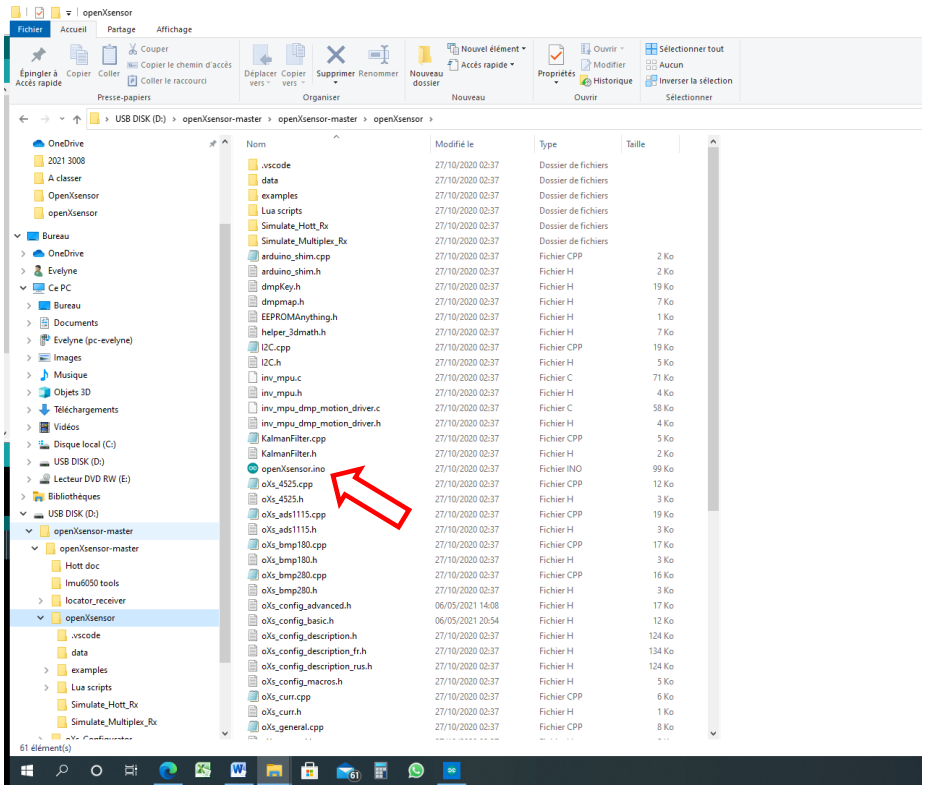
## Paramétrage

Ca y est on y est presque ! !

Nous sommes déjà dans l'IDE Arduino puisque nous venons de paramétrer notre projet (à moins que vous ayez paramétré les fichiers oXs_config_basic et oXs_config_advance avec un éditeur de texte)

Au cas où un petit rappel, on revient dans l'arborescence de notre ^projet

Double-cliquez sur le fichier openxsensor.ino

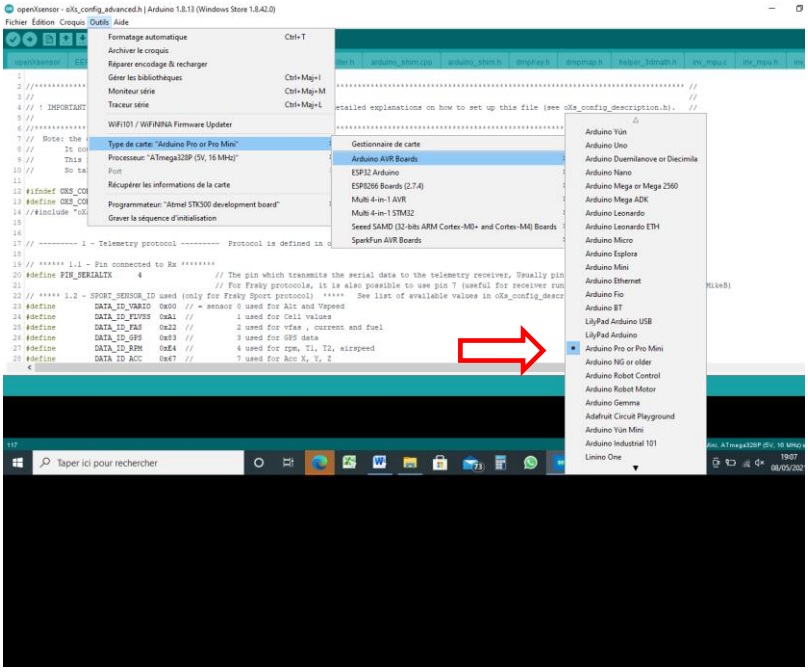



Ca y est l'IDE est ouvert avec notre projet et nos deux fichiers de config paramétrés

Il faut maintenant sélectionner la carte Arduino utilisée, dans notre cas il s'agir d'une carte pro mini.

Dans l'onglet « outils », cliquer sur :

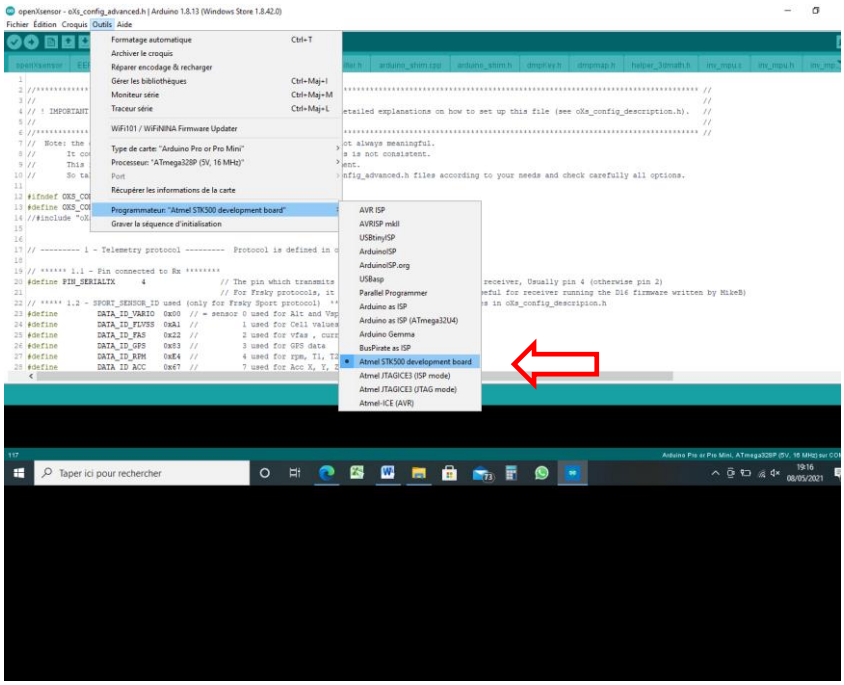
- type de carte
  - Arduino AVR boards
    - Arduino pro or pro mini




La carte est sélectionnée.

Dans l'onglet « outils », cliquer sur :

- programmateur
  - Atmel STK500 development board

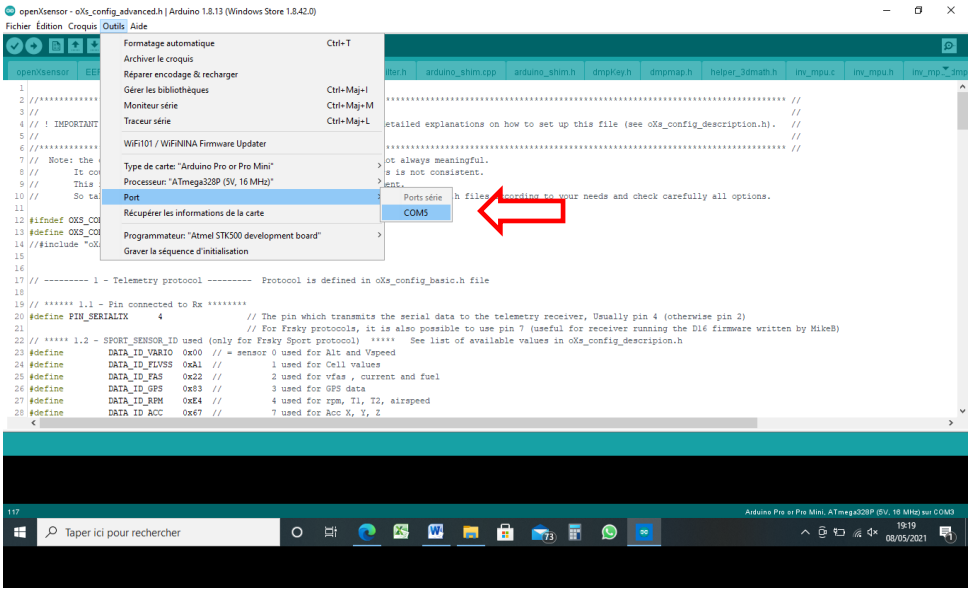



Le programmateur est sélectionné

Dans l'onglet « outils », cliquer sur :

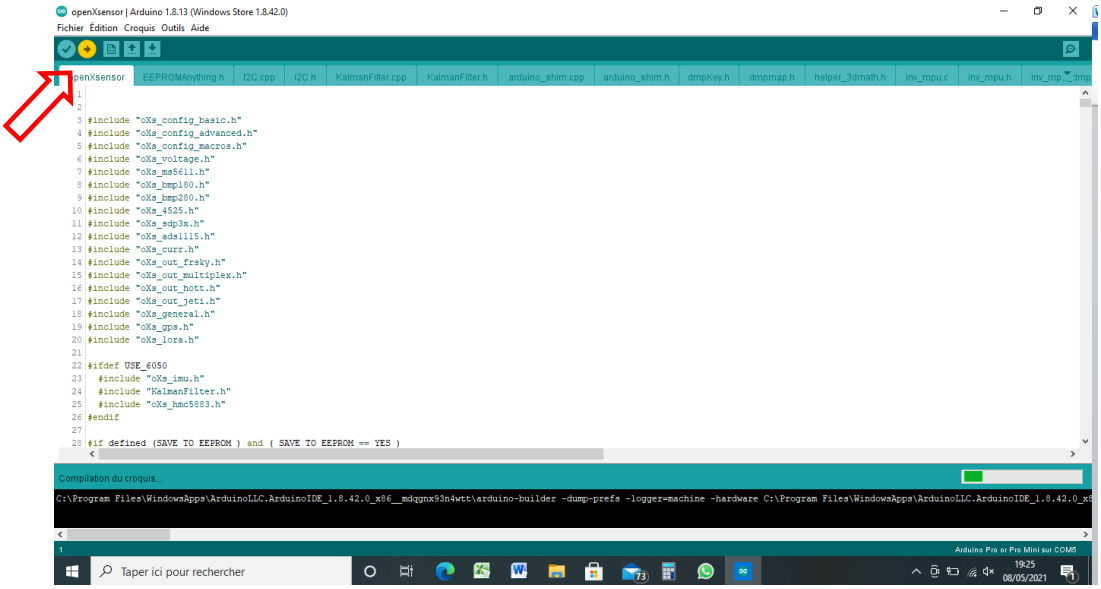
- port
    - Sélectionner le port qui s'affiche.

**Attention :**
Pour sélectionner le port de communication il faut que votre programmateur FTDI soit branché sur un port USB de votre ordinateur

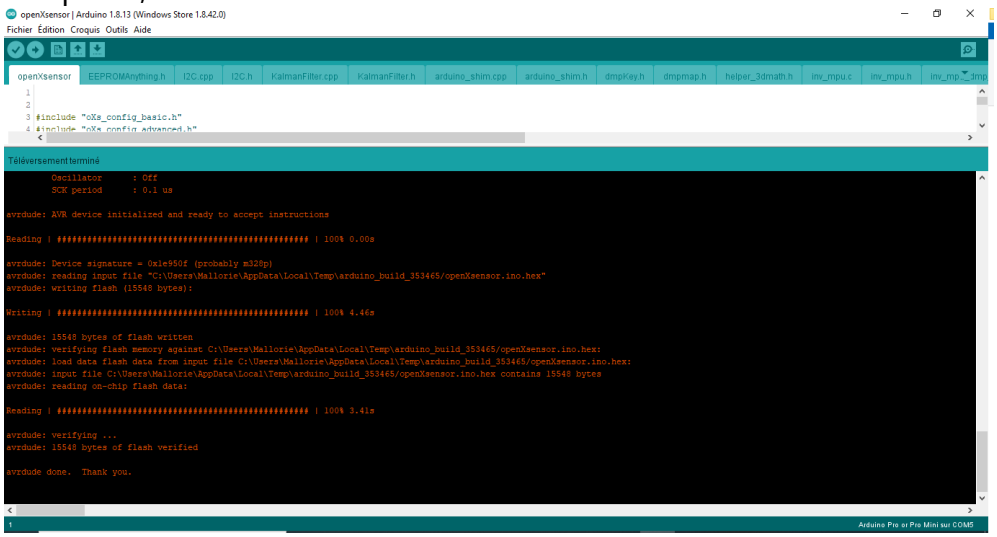


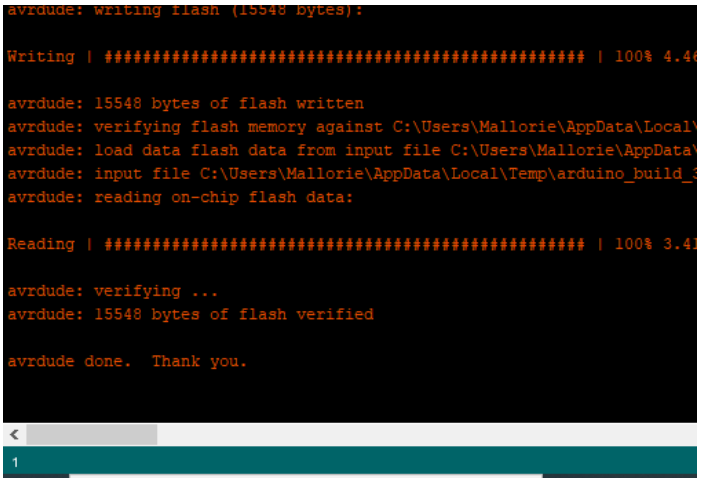

Le port de communication est maintenant sélectionné et nous sommes prêt à téléverser le programme dans la carte Arduino

Il suffit maintenant de cliquer sur la flèche en haut à gauche (en jaune) et c'est parti ! ! ! Le téléversement démarre.



Si tout c'est bien passé, vous aurez l'écran suivant




.....Et on vous dit même Merci ! !

## Etalonnage des capteurs

### Tension de référence

Comme indiqué précédemment les valeurs de tension de la carte sont provisoires. Après la première programmation et mise en service, il faudra mesurer la tension entre le pin VCC et Ground sur l'Arduino, convertir la valeur obtenue en mVolt (Ex. :4.85v =4850) et remplacer la valeur dans:
#define REFERENCE_VOLTAGE 4890 (4.90v =4890) par la nouvelle valeur obtenue et téléverser à nouveau. La précision sera grandement améliorée !

**NOTA** : Dans le cas de notre projet, cet étalonnage ne sera pas nécessaire car nous utilisons la tension interne de l'Arduino (1.1v) comme référence et pas la tension d'alimentation de la carte (Vcc)

## Mesure de tension

Il y a deux paramètres pour ajuster la valeur affichée des tensions mesurées. Un paramètre d'offset (décalage) et un paramètre d'échelle.
Mettre en route l'émetteur et la télémétrie, mesurer la tension entre le fil de mesure et la masse, noter la valeur et la comparer avec celle affichée sur la télémétrie. Afin d'ajuster le bon paramètre, il est nécessaire de faire cette mesure au moins deux fois avec une batterie bien chargée et une batterie déchargée.
Pour chacune de ces deux mesures, on divise la valeur mesurée sur le fil de mesure par la valeur lue sur la télémétrie. Si les deux résultats sont à peu près identiques, il faudra modifier le paramètre scale avec la valeur trouvée.
Si les deux mesures montrent un décalage identique, par exemple 0.2v d'écart dans les deux cas, il faudra modifier le paramètre d'offset.

Ces deux paramètres sont dans le fichier oXs_config_advance dans le chapitre 6.2 :

#define SCALE_VOLTAGE **1.0** , 1.0 , 1.0 , 1.0 , 1.0 , 1.0 // optionnal, can be negative, can have decimals

#define OFFSET_VOLTAGE **0** , 0 , 0 , 0 , 0 , 0 // remplacer le 1er 0 par la valeur mesurée

## Baromètre MS5611

Afin de définir la meilleure valeur pour votre capteur (cela peut varier d'un MS5611 à l'autre), la façon la plus facile est de tester plusieurs valeurs. Il est préférable de faire les essais dans des conditions atmosphériques calmes (journée calme sans vent)
- Assurez-vous que l'altitude est affichée sur votre émetteur
- Assurez-vous que le capteur MS5611 est protégé de la lumière (le capteur est photosensible)
- Premièrement avec le paramètre ALT_TEMP_COMPENSATION mis à zéro (ou mis en commentaire), démarrer votre émetteur et votre récepteur. Ne bouger pas votre oXs (l'altitude affichée sur votre émetteur doit être 0).
- Après 5 minutes noter la différence d'altitude.
- Eteignez votre récepteur et votre émetteur pour 5 minutes minimum afin de faire redescendre la température interne du capteur.
- Répéter cette opération une ou deux fois.
- Si la déviation d'altitude est environ la même entre chaque essais et dépasse 1 mètre, il sera possible de réduire cette dérive avec une valeur dans le paramètre ALT_TEMP_COMPENSATION
- Si la dérive est positive, alors augmenter le paramètre ALT_TEMP_COMPENSATION autrement réduise le; ALT_TEMP_COMPENSATION peut être négatif (cela peut être le cas si la dérive est négative).
- Il est très compliqué de calculer la valeur de correction. Sur mon capteur j'ai une valeur = 1000 pour compenser une variation de 3 mètre mais cela peut varier sur votre capteur.

- Mettre une valeur au paramètre ALT_TEMP_COMPENSATION, répéter les tests précédent (allumer votre émetteur et votre récepteur, attendre 5 min, noter la dérive d'altitude) et augmenter ou diminuer la valeur si besoin.

Ce paramètre est dans le fichier oXs_config_advance dans le chapitre 4.5 :
#define ALT_TEMP_COMPENSATION **800**

Et téléverser à nouveau.

## Mise en route

Les modules de télémétrie OpenXsensor ont été essayés dans notre club sur des ensemble Frsky et Jeti avec succès.

### Radio Frsky ou open Tx

Avec openTx il est nécessaire de découvrir les capteurs de télémétrie afin de pour ensuite les afficher.

**Remarque :** Si un GPS est programmé, la découverte des capteurs se fera sans découvrir le GPS tant que celui-ci n'aura pas son « FIX », c'est à dire tant qu'il n'aura pas trouvé suffisamment de satellites pour donner une information fiable.

### Radio JETI

Avec un émetteur JETI, il est nécessaire de découvrir les capteurs de télémétrie afin de pouvoir ensuite les afficher. Votre module de télémétrie apparaitra comme un capteur nommé oXs.

**Remarque :** Un projet oXs est vu par votre émetteur comme un capteur unique. Dans les émetteur JETI il semblerait qu'un capteur ne puisse pas avoir plus de 15 informations.
Ceci n'est pas un problème avec l'exemple décrit mais c'est à prendre en compte si vous souhaitez réaliser vos propres modules avec des paramétrages différents.
Pour info, le module pris en exemple envoi déjà 12 infos à l'émetteur JETI

Bon vol, avec télémétrie ! ! !