Projet Carte de télémétrie Open source

Description de la configuration

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
V0.OR	Version originale	2021 05 07

****

Information importante :
Pour configurer votre oXs, vous devez modifier au moins le fichier oXs_config_basic.h
Dans certains cas, vous devrez également modifier le fichier oXs_config_advanced.h
****

Voir le github OpenXsensor https://github.com/openXsensor/
Document mis à jour à l'aide de : https://github.com/openXsensor/openXsensor/tree/master/openXsensor
oXs_config_description_fr.h

Sommaire

Résumé des raccordements

- 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

Attention : garder la possibilité de débrancher le GPS pour le chargement du programme

- 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)**

Description générale de toutes les options

Les fichiers oXs_config_basic.h et oXs_config_advanced.h permettent à l'utilisateur de configurer différentes options. Voici un résumé des principales options.

Note : Les paramètres activés commencent par "#define", suivi du nom du paramètre et dans la majorité des cas d'une valeur pour ce paramètre.

Pour activer ou désactiver un paramètre il faut supprimer ou ajouter les "//" qui précède "#define" Exemple "// #define ParameterToto Bonjour" le paramètre Toto ne sera pas pris en compte car pour Arduino c'est considérer comme un commentaire.

1 - Choix du protocole de télémétrie

Actuellement oXs supporte 4 protocoles de télémétrie: Multiplex , Frsky, Jeti and Hott (=Graupner)
Un et un seul protocole peut-être sélectionné; Exemple si vous activez le protocole Multiplex, les protocoles Frsky, Jeti et Hott sont désactivés (et vice et versa)
FrSky utilise 2 protocoles différents :
- SPORT utilisé par les récepteurs de la série X (comme le X8R ou X6R)
- HUB utilisé par les récepteurs de la série D (comme le D4R-II)
oXs peut supporter les deux protocoles FrSky en même temps (en utilisant l'option FRSKY_SPORT_HUB). oXs peut normalement détecter automatiquement le type de récepteur auquel il est connecté
Si vous utilisez cette option vous n'aurez pas besoin de reprogrammer votre oXs si vous passez d'un récepteur série X à un de la série D.
Cependant, l'auto détection ne semble pas marcher a 100% et demande plus de mémoire…
Dans le cas où l'auto détection ne fonctionne pas ou si vous avez besoin de plus de mémoire (car vous utilisez beaucoup de capteurs connectés à votre oXs), vous pouvez forcer oXs à utiliser le protocole FRSKY spécifique que vous utilisez
Pour sélectionner le protocole utilisé par oXs, il faut remplir la ligne #define PROTOCOL avec le nom du protocole sélectionné avec l'une des valeurs suivantes: FRSKY_SPORT , FRSKY_HUB , FRSKY_SPORT_HUB , MULTIPLEX , HOTT; JETI

****

#define PROTOCOL FRSKY_SPORT // sélectionner une valeur parmi FRSKY_SPORT , FRSKY_HUB , FRSKY_SPORT_HUB , MULTIPLEX , HOTT, JETI

1.1 - Sélection de la sortie vers le récepteur

oXs doit être connecté au récepteur afin de transmettre les informations.
Pour cela, une sortie DIGITAL de l'Arduino doit être connectée au récepteur.
Vous devez spécifier la PIN Arduino utilisée pour cette fonction.
Valeur par défaut: **4** ; Les valeurs autorisées sont 2 ou 4 mais faites attention de ne pas utiliser la même PIN pour 2 fonctions.

****

#define PIN_SERIALTX 4

1.2 SPORT_SENSOR_ID (utilisé uniquement pour le protocol FrSky Sport)

Pour le protocole SPORT, il peut y avoir plusieurs capteurs connectés sur le même bus mais chaque capteur doit avoir un identifiant « SPORT_SENSOR_ID » diffèrent.
Pour le protocole SPORT, oXs peut utiliser jusqu'à 6 identifiants. Les 6 SPORT_SENSOR_ID utilisés par oXs sont :
DATA_ID_VARIO 0x00 // 0 utilisé pour l'altitude et la vitesse vertical
DATA_ID_FLVSS 0xA1 // 1 utilisé pour les mesures de tension de batterie
DATA_ID_FAS 0x22 // 2 utilisé pour vfas, courant et carburant
DATA_ID_GPS 0x83 // 3 utilisé pour les données GPS
DATA_ID_RPM 0xE4 // 4 utilisé pour les rpm, T1, T2, et la vitesse air
DATA_ID_ACC 0x67 // 7 utilisé pour les accélérations sur X, Y, Z
Au besoin (par exemple pour connecter 2 oXs envoyant deux données identiques), vous pouvez changer les valeurs de SPORT_SENSOR_ID mais vous devez uniquement sélectionner des adresses parmi les valeurs suivantes :

0x00,0xA1,0x22,0x83,0xE4,0x45,0xC6,**0x67**,0x48,0xE9,0x6A,0xCB,0xAC,0x0D,0x8E,0x2F,0xD0,0x71,0xF2,0x53,0x34,0x95,0x16,0xB7,0x98,0x39,0xBA,0x1B
Les valeurs du paramètre SPORT_SENSOR_ID sont définies dans le fichier oXs_config_advanced.h (section 1.2)

2 - Données à transmettre

Selon les paramètres sélectionnés dans le fichier config.h (et les capteurs connectés à oXs), oXs peut calculer plusieurs mesures.
Pour le protocole Multiplex, vous devez spécifier quelles mesures doivent être envoyées à l'émetteur (et sur quelle ligne de l'écran de l'émetteur chaque donnée doit être affichée) (voir ci-dessous)
Pour les protocoles SPORT, HUB, JETI et HOTT, les mesures sont automatiquement transmises dans des champs prédéfinis et certains protocoles autorisent d'envoyer certaines mesures en réutilisant certains champs. Une des mesures est la "Vitesse verticale principale".

2.1 Protocole FrSKY (SPORT et HUB)

Les mesures transmises automatiquement sont :
- Altitude relative (cm), Vitesse verticale principale (cm/sec) (si au minimum un capteur barométrique est connecté)
- Tensions des cellules (si NUMBER_OF_CELLS > 0)
- Courant (quand un capteur de courant est connecté)
- GPS (long, lat, vitesse, altitude, course) (quand un GPS est connecté)
- RPM (hz) (quand un capteur de vitesse de rotation est connecté)

En supplément de ces mesures, vous pouvez spécifier les paramètres des mesures suivantes Vfas, Fuel, A3, A4, T1, T2, AccX, AccY, AccZ (voir les options disponibles ci-dessous)
Commenter toutes les lignes qui ne doivent pas être transmises.
ex :

#define VFAS_SOURCE VOLT_4 // sélectionner une valeur parmi: VOLT_1, VOLT_2, VOLT_3 , VOLT_4, VOLT_5 , VOLT_6, ADS_VOLT_1, ADS_VOLT_2, ADS_VOLT_3, ADS_VOLT_4

#define FUEL_SOURCE VOLT_4 // sélectionner une valeur parmi: VOLT_1, VOLT_2, VOLT_3 , VOLT_4, VOLT_5 , VOLT_6, ADS_VOLT_1, ADS_VOLT_2, ADS_VOLT_3, ADS_VOLT_4

#define A3_SOURCE VOLT_4 // sélectionner une valeur parmi: VOLT_1, VOLT_2, VOLT_3 , VOLT_4, VOLT_5 , VOLT_6, ADS_VOLT_1, ADS_VOLT_2, ADS_VOLT_3, ADS_VOLT_4

#define A4_SOURCE VOLT_4 // sélectionner une valeur parmi: VOLT_1, VOLT_2, VOLT_3 , VOLT_4, VOLT_5 , VOLT_6, ADS_VOLT_1, ADS_VOLT_2, ADS_VOLT_3, ADS_VOLT_4

#define ACCX_SOURCE TEST_1 // sélectionner une valeur parmi: TEST_1, TEST_2, TEST_3, GLIDER_RATIO , SECONDS_SINCE_T0 ,AVERAGE_VSPEED_SINCE_TO , VOLT_1, VOLT_2, VOLT_3, VOLT_4, VOLT_5, VOLT_6, PITCH, ROLL, YAW, ADS_VOLT_1, ADS_VOLT_2, ADS_VOLT_3, ADS_VOLT_4

#define ACCY_SOURCE TEST_2 // sélectionner une valeur parmi: TEST_1, TEST_2, TEST_3, GLIDER_RATIO , SECONDS_SINCE_T0 ,AVERAGE_VSPEED_SINCE_TO , VOLT_1, VOLT_2, VOLT_3, VOLT_4, VOLT_5, VOLT_6, PITCH, ROLL, YAW, ADS_VOLT_1, ADS_VOLT_2, ADS_VOLT_3, ADS_VOLT_4

#define ACCZ_SOURCE TEST_3 // sélectionner une valeur parmi: TEST_1, TEST_2, TEST_3, GLIDER_RATIO , SECONDS_SINCE_T0 ,AVERAGE_VSPEED_SINCE_TO , VOLT_1, VOLT_2, VOLT_3, VOLT_4, VOLT_5, VOLT_6, PITCH, ROLL, YAW, ADS_VOLT_1, ADS_VOLT_2, ADS_VOLT_3, ADS_VOLT_4

#define T1_SOURCE GLIDER_RATIO // sélectionner une valeur parmi: TEST_1, TEST_2, TEST_3, GLIDER_RATIO , SECONDS_SINCE_T0 ,AVERAGE_VSPEED_SINCE_TO , SENSITIVITY , PPM, VOLT_1, VOLT_2, VOLT_3, VOLT_4, VOLT_5, VOLT_6, ADS_VOLT_1, ADS_VOLT_2, ADS_VOLT_3, ADS_VOLT_4

#define T2_SOURCE SENSITIVITY // sélectionner une valeur parmi: TEST_1, TEST_2, TEST_3, GLIDER_RATIO , SECONDS_SINCE_T0 ,AVERAGE_VSPEED_SINCE_TO , SENSITIVITY , PPM, VOLT_1, VOLT_2, VOLT_3, VOLT_4, VOLT_5, VOLT_6, ADS_VOLT_1, ADS_VOLT_2, ADS_VOLT_3, ADS_VOLT_4

2.2 Hott protocole

Les mesures transmises automatiquement sont :
- Altitude relative (cm), Vitesse verticale principale (cm/sec) (si au minimum un capteur barométrique est connecté)
- Tensions des cellules (si NUMBER_OF_CELLS > 0)
- jusqu'à trois mesures de tension de batterie (le setup spécifie quelle tension doivent être mesurées et transmises - Exemple VOLT_1,VOLT_2, ...)
- Courant (quand un capteur de courant est connecté)
- GPS (long, latitude, vitesse, altitude, course) (quand un GPS est connecté)
- RPM (hz) (quand un capteur de vitesse de rotation est connecté)

Lorsque vous mesurez des tensions de batteries, une alarme peut être configurée pour indiquer qu'une tension est inférieure à un niveau préréglé.
Pour activer cette alarme, dé-commenter la ligne ci-dessous (en milliVolt) pour régler le seuil d'alarme
//#define CELL_UNDERVOLTAGE_WARNING 3300 // Attention c'est en mV;

Vous pouvez aussi sélectionner comment les valeurs temperature1 et temperature2 seront remplies.
Note : ces deux champs peuvent seulement retourner des valeurs allant de -20 à 235; pour le mode PPM une valeur de -100 sera affiché 0 et +100 sera affiché 200
Mettre les lignes ci-dessous en commentaire quand aucune de ces mesures ne doit être transmise par télémétrie
ex:

#define BATTERY_1_SOURCE VOLT_4 // sélectionner une valeur parmi: VOLT_1, VOLT_2, VOLT_3 , VOLT_4, VOLT_5 , VOLT_6
#define BATTERY_2_SOURCE VOLT_2 // sélectionner une valeur parmi: VOLT_1, VOLT_2, VOLT_3 , VOLT_4, VOLT_5 , VOLT_6
#define MAIN_BATTERY_SOURCE VOLT_5 // sélectionner une valeur parmi: VOLT_1, VOLT_2, VOLT_3 , VOLT_4, VOLT_5 , VOLT_6
#define TEMPERATURE_1_SOURCE TEST_1 // sélectionner une valeur parmi: TEST_1, TEST_2, TEST_3 , GLIDER_RATIO , SENSITIVITY , PPM
#define TEMPERATURE_2_SOURCE SENSITIVITY // sélectionner une valeur parmi: TEST_1, TEST_2, TEST_3 , GLIDER_RATIO , SENSITIVITY , PPM

2.3 Multiplex

Pour le protocole Multiplex, vous devez définir quelle mesures de oXs sont transmises et ce sur quelle ligne de l'émetteur.
Vous devez aussi spécifier les échelles qui doivent être appliquées par oXs et la plage de valeurs afin de définir le on/off des alarmes.
Vous DEVEZ spécifie une ligne pour chaque mesure à envoyer à l'émetteur. Chaque ligne doit contenir :
- 1 : le numéro de ligne de l'écran de l'émetteur où doit être affichée la mesure. Les valeurs autorisées vont de 2 à 15. Attention de ne pas utiliser deux fois la même valeur.
- 2 : une virgule
- 3 : le nom du paramètre oXs qui doit être transmis sur cette ligne (exemple "VOLT_1") (voir note (2))
- 4 : une virgule
- 5 : un coefficient multiplicateur à appliquer à la valeur transmise (mettre "1" pour conserver la mesure original, 10 pour multiplier la mesure par 10 , ...) (voir note (3))
- 6 : une virgule
- 7 : un coefficient diviseur à appliquer à la valeur transmise (mettre "1" pour conserver la mesure original, 10 pour diviser la mesure par 10 , ...) (voir note (3))

- 8 : une virgule
- 9 : un décalage (offset) à appliquer à la valeur après avoir appliqué les coefficients multiplicateur et diviseur ((mettre "0" pour conserver la mesure d'origine, "100" pour ajouter 100, ...)
- 10 : une virgule
- 11 : une valeur pour la valeur minimum de l'alarme (voir note (4))
- 12 : une virgule
- 13 : une valeur pour la valeur haute de l'alarme (voir note (4))
- 14 : une virgule et un "\" s'il y a une mesure après celle-ci (attention à ne pas mettre ",\" pour la dernière mesure);

/!\ ATTENTION que "\" doit être le dernier caractère de la ligne (aucun espace et toléré après)
/!\ ATTENTION que aucune ligne de commentaire ("*...") ne soit glissée entre les différentes lignes
Note (2) : Une mesure oXs (exemple VOLT_1) ne peut pas être déclarée dans plusieurs lignes.
Note (3) : Les coefficients multiplicateur, diviseur et l'offset doivent être une valeur entière (pas de chiffre avec une virgule); ils peuvent être négatifs (exemple "-100").
Les coefficients multiplicateurs et diviseur peuvent être utiles dans les cas suivant:
- pour convertir d'une unité de mesure a une autre (mètre <> pied)
- pour convertir en pourcentage (exemple multiplier par 100 et diviser par 4000 afin d'obtenir un % de consommation pour un accu de 4000 mAmph)
- pour ajuster le nombre de décimale à afficher sur l'émetteur.
- pour avoir par exemple une valeur de quantité de carburant commençant à 100 (en pourcent) et diminuant à 0 quand la consommation augmente (dans ce cas vous devez utiliser un coefficient multiplicateur négatif et un offset de 100%).

Multiplicateur/diviseur/offset doivent être définis mais ne sont pas toujours utilisés par oXs. C'est le cas quand:
- CELLS_1_2, CELLS_3_4, CELLS_5_6 sont transmis (car ces champ requièrent un formatage spécial au niveau de l'émetteur)

Note (4) : les limites d'alarme doivent être des nombres entiers (pas de chiffre avec une virgule); ils peuvent être négatifs (exemple "-100").
Afin de définir les alarmes prévues dans le protocole Multiplex, oXs va procéder de la façon suivante:
- adapter le nombre de décimales de la mesure faite par oXs (la plupart des mesures de oXs ont plus de décimales au sein de oXs que ce que prévoit le protocole Multiplex)
- appliquer les coefficients multiplicateurs, diviseur et l'offset
- comparer le résultat avec les niveaux haut et bas des alarmes

Les limites doivent être fixées en tenant compte de tous les digits visibles à l'écran.
Exemple : Les tensions sont en principe en mVolt. Si vous voulez une alarme quand la tension est inférieure à 11.2 Volt, régler la valeur d'alarme minimum a 112.
Afin d'inhiber les alarmes mettre la valeur minimum à -16384 et/ou la valeur maximale a 16383 (valeur minimal et maximal pour le protocole de type Multiplex sur 15 bits)

Ci-dessous un exemple de réglage pour transmettre sur le protocole Multiplex
- sur la ligne 3, l'altitude relative (sans alarme)
- sur la ligne 6, la vitesse verticale (avec une alarme si la valeur excède 50m/s vers le haut ou vers le bas)

```
#define SETUP_MULTIPLEX_DATA_TO_SEND \
3 , REL_ALTIMETER , 1 , 1 , 0 , -16384 , 16383,\
6 , VERTICAL_SPEED , 1 , 1 , -500 , 500
```

/!\ IMPORTANT : toujours conserver la ligne "#define SETUP_DATA_TO_SEND \" ; ne pas insérer de ligne avant ou après les lignes utilisées pour définir les données à transmettre.

Ici vous trouverez la liste des mesures que peut envoyer oXs.
Code à utiliser Unité Signification
ALTIMETER m Altitude absolue (sur base du premier capteur baro)
REL_ALTIMETER m Altitude relative (sur base du premier capteur de baro)
ALTIMETER_MAX m Altitude relative maximale
VERTICAL_SPEED cm/s Vitesse verticale principale (configurée dans VSPEED_SOURCE)
SENSITIVITY none Sensibilité du Vario
VOLT_1 0.1V(5) Valeur lue sur la première PIN_VOLTAGE
VOLT_2 0.1V(5) Valeur lue sur la 2ème PIN_VOLTAGE
VOLT_3 0.1V(5) Valeur lue sur la 3ème PIN_VOLTAGE
VOLT_4 0.1V(5) Valeur lue sur la 4ème PIN_VOLTAGE
VOLT_5 0.1V(5) Valeur lue sur la 5ème PIN_VOLTAGE
VOLT_6 0.1V(5) Valeur lue sur la 6ème PIN_VOLTAGE
CURRENTMA 0.1A MilliAmp mesuré par le capteur de courant
MILLIAH mAh Consommation en Milli Amp heure
RPM Hz Rotation par minute
AIR_SPEED 0.1 km/h Vitesse air
CELL_1 0.1V(5) Valeur basée sur la première PIN_VOLTAGE
CELL_2 0.1V(5) Valeur basée sur la première et la 2ème PIN_VOLTAGE
CELL_3 0.1V(5) Valeur basée sur les 2ème et 3ème PIN_VOLTAGE
CELL_4 0.1V(5) Valeur basée sur les 3ème et 4ème PIN_VOLTAGE
CELL_5 0.1V(5) Valeur basée sur les 4ème et 5ème PIN_VOLTAGE
CELL_6 0.1V(5) Valeur basée sur les 5ème et 6ème PIN_VOLTAGE
CELL_MIN 0.1V(5) Valeur basée sur CELL_1 ... CELL_6
CELL_TOT 0.1V(5) Valeur basée sur VOLT1 ... VOLT6
PPM Valeur définie par l'émetteur pour contrôler certaines fonctions (sensibilité, ...) (la plage est normalement -100 / +100)
GPS_COURSE 0.1 deg Direction du mouvement
GPS_SPEED 0.1 km/h Vitesse sol (2D ou 3D)
GPS_ALTITUDE m Altitude absolue
GPS_DISTANCE m Distance du point de départ
GPS_BEARING 0.1 deg Direction par rapport au point de départ
TEST_1 Valeur utilisée pour des tests
TEST_2 Valeur utilisée pour des tests
TEST_3 Valeur utilisée pour des tests
(5) Les unités dépendent des paramètres utilisés (exemple quand une tension est fournie par un capteur de température, l'unité peut être des degrés)
Pour mesurer des cellules de lipo, vous devez vous assurer que la calibration soit telle que les valeurs internes soient en millivolt (et oXs va convertir les mVolt en 0.1Volt au moment de la transmission).

2.4 - Données Jeti

Les mesures transmises automatiquement sont :
- Altitude relative (cm), Vitesse verticale principale (cm/sec) (si au minimum un capteur barométrique est connecté) et l'altitude maximum
- Tensions des cellules (si NUMBER_OF_CELLS > 0)
- Courant (quand un capteur de courant est connecté)

- GPS (longitude, latitude, vitesse, altitude, course) (quand un GPS est connecté)
A ceci vous pouvez aussi demander l'envoi de:
- une des six tensions (si la tension est mesurée); quand un capteur NTS est utilisé pour mesurer une température, il faut définir aussi un VOLTAGE_SOURCE (voir la section a propos des capteurs NTC)
- une température: vous pouvez sélectionner la température fournie par le MS5611 ou une sonde NTC.
D'autres champs peuvent être ajoutés à la demande.
//#define VOLTAGE_SOURCE VOLT_1 // sélectionner une valeur parmi: VOLT_1, VOLT_2, VOLT_3 , VOLT_4, VOLT_5 , VOLT_6
//#define TEMPERATURE_SOURCE MS5611 // sélectionner une valeur parmi: MS5611 and NTC

2.5 - Comment remplir TEST_1, TEST_2, TEST_3

oXs peut calculer plus de mesures que ce que prévoit les protocoles.
Certaines peuvent être directement configurées dans les sections 2.1 à 2.4 (exemple VOLT_3 peut être transmit dans T1 (pour le protocole SPORT)
D'autres sont rarement utilisées et requièrent de procéder en deux étapes;
- Première étape, décider quelles mesures vont remplir les champs TEST_1, TEST_2, TEST_3 (dans la section 2.5)
- Seconde étape, décider dans quel paramètre de télémétrie ces mesures seront envoyées (dans la section 2.1 to 2.4)
Pour choisir comment ces paramètres TEST_1, TEST_2, TEST_3 doivent être remplis dé-commentez une ou plusieurs lignes ci-dessous.
Si vous dé-commentez plusieurs lignes, prenez garde à ne pas utiliser plusieurs fois le même paramètre.
****

//#define FILL_TEST_3_WITH_EXPECTED_ALT // dé-commenter cette ligne si oXs doit calculer une altitude "prévue" sur base de l'altitude actuelle, de la Vitesse verticale et de l'acceleration verticale
//#define EXPECTED_ALT_AT_SEC 0.2 // temps (en sec) pour l'altitude "prévue" (Cette ligne doit être dé-commentée si la précédente l'est aussi)
//#define FILL_TEST_1_2_3_WITH_LINEAR_ACC // dé-commenter cette ligne si oXs doit remplir les paramètres TEST_1, TEST_2, TEST_3 avec les accélérations linéaires
//#define FILL_TEST_1_2_WITH_VSPEED_AND_ALT_FROM_SECOND_VARIO // dé-commenter pour activer cette option
//#define FILL_TEST_1_WITH_DTE // dé-commenter pour activer cette option
//#define FILL_TEST_2_WITH_PPM_AIRSPEED_COMPENSATION // dé-commenter pour activer cette option
//#define FILL_TEST_1_WITH_YAWRATE // dé-commenter pour activer cette option
//#define FILL_TEST1_WITH_HEADING_FROM_MAGNETOMETER // dé-commenter pour activer cette option
//#define FILL_TEST_3_WITH_FLOW_SENSOR_CONSUMPTION // dé-commenter pour activer cette option

3 - Configuration de PPM

oXs peut (en option) changer certains paramètres et/ou remettre à zéro certaines données à partir d'actions prises sur l'émetteur
Pour tous les protocoles, cela peut être réalisé à partir d'une sortie servo du récepteur pour recevoir les informations de l'émetteur.
Cette fonction permet depuis l'émetteur:
- de changer la sensibilité du vario en utilisant ou un potard ou un interrupteur de votre émetteur.
- de changer entre un vario compensé et non compensé (par exemple quand on utilise 2 capteurs barométriques dont l'un est connecté sur une sonde TEK)
- de réinitialiser le capteur de vitesse air (quand la valeur dévie avec la température)
- de changer le facteur de compensation du vario (quand le vario compensé utilise le capteur de vitesse air) en utilisant un potentiomètre ou un interrupteur de l'émetteur.
Ces différentes fonctions demandent des mixages dans l'émetteur et une configuration particulière dans le fichier oXs-config_advanced.h (voir ci-dessous).

Si vous voulez utiliser ces fonctions, vous devez;
- dé-commenter les 3 lignes suivantes (en supprimant les // devant la ligne) et
- spécifier à la ligne PIN_PPM, la PIN Arduino qui sera connectée à la sortie servo du récepteur
Par défaut**: 2** ; vous pouvez aussi connecter la PIN 3 pour lire le signal PPM.
Faire bien attention d'utiliser la pin sélectionnée (2 ou 3) uniquement pour le signal PPM.
Garder cette ligne en commentaire si vous ne voulez pas utiliser cette fonction.
- spécifier à la ligne PPM_MIN_100 la largeur d'impulsion (en micro seconde) quand la voie envoie la valeur "logique" = -100.
La valeur par défaut est 988.
- spécifier à la ligne PPM_PLUS_100 la largeur d'impulsion (en micro seconde) quand la voie envoie la valeur "logique" = +100.
La valeur par défaut est 2012.
Les deux dernières valeurs sont utilisées afin de calibrer oXs avec l'émetteur quand la fonction PPM est utilisée
Il est probable que si vous utilisez les valeurs par défaut cela fonctionne.
Il est plus sûr pour le bon fonctionnement de calibrer ces paramètres (uniquement si vous utilisez la fonction PPM).
Pour ce faire, assurez-vous de transmettre le champ de mesure oXs "PPM" à émetteur (par exemple en l'envoyant comme T1).
Noter les valeurs affichées sur l'émetteur (par exemple dans le champ T1) lorsque le TX envoie respectivement -100 et +100 sur le canal PPM (utilisez un commutateur ou un pot).
Remplissez ces valeurs dans ces paramètres et rechargez votre programme dans l'Arduino.

Si vous utilisé le protocole FRSKY SPORT (uniquement pour récepteur de la série X) avec openTX 2.2(ou dessus) vous pouvez envoyer le signal PPM sans avoir à connecter oXs à une sortie servo du récepteur.
OpenTx 2.2 vous permet de lancer un script LUA qui envoie des valeurs depuis votre émetteur vers oXs
Pour activer cette option, vous devez :
- dé-commenter la ligne #define PPM_VIA_SPORT
- lancer le script LUA qui vous permet d'envoyer via le protocole SPORT le même type de valeur qu'une voie du récepteur. (Cette valeur est comprise entre -100 and 100)
Ce script LUA doit utiliser une commande comme ceci :
Local ret = sportTelemetryPush(0x0D , 0x10 , 0x0010 , -52)
ou

- 0xOD : est le numéro d'identification utilisé (défini dans les fichiers oXs_config_advanced.h)
- 0x10 : est une valeur non utilisé
- 0x0010 : est le code utilisé pour identifier qui s'agit d'une valeur PPM
- -52 : est la valeur PPM transmise (-52 est juste un exemple; elle doit être adaptée à la valeur à transmettre)

Vous pouvez trouver un exemple de script LUA dans ce package (dans le dossier "lua scripts"). Le nom du fichier est oXsppm.lua et doit être installé dans le répertoire SCRIPTS/FUNCTIONS de la carte SD.

****

#define PIN_PPM 2
#define PPM_MIN_100 988
#define PPM_PLUS_100 2012
#define PPM_VIA_SPORT // dé-commenter cette ligne pour recevoir un signal PPM par le protocole SPORT a la place d'une sortie servo du récepteur (Cela nécessite un émetteur équiper d'openTX et contenant le script en LUA)

4 - Configuration du vario

4.1 - Connections de 1 ou 2 capteurs barométriques (optionnels)

oXs peut être connecté à différents capteurs barométriques.
Les meilleurs résultats sont obtenus avec un capteur barométrique MS5611. Ce type de capteur est généralement monté sur un module (comme le GY-63 ou GY-86) avec un régulateur de tension de 3.3 volt et un adaptateur de voltage pour bus I2C.
Il est possible d'utiliser aussi des modules avec des capteurs BMP085 ou BMP180 ou BMP280 (qui sont plus économiques mais moins précis).
Il est possible de connecter 1 ou 2 capteurs barométriques ; le premier peut être un capteur de type BMP085 ou BMP180 ou BMP280 ou encore un MS5611; le second capteur doit être un MS5611 (car les capteurs de type BMPxxx ont qu'une adresse I2C).
Note : Toujours utiliser un module qui possède un régulateur de tension 3,3Volt et si possible adaptateur de voltage pour le bus I2C quand on utilise 5 Volt comme Vcc de l'Arduino.
Chaque capteur peut calculer l'altitude absolue ou relative (en mètre avec 1 décimale) et la vitesse verticale (en mètre/seconde avec 2 décimales).
Un second capteur peur être utile si associé à une sonde TEK (pour avoir un vario compensé pneumatiquement) et si l'option PPM est active, il est possible de sélectionner depuis l'émetteur le capteur qui vas générer le son du vario.
Ceci permet de basculer entre un vario compensé pneumatiquement et un vario non compensé.
Quand on utilise deux capteurs barométriques, oXs peut transmettre la moyenne des vitesses verticales des deux capteurs. Ceci ne semble toutefois pas très utile.
Vous devez spécifier le type de capteur utilisé pour le premier vario avec la ligne #define FIRST_BARO_SENSOR_USE MS5611
La valeur à renseigner doit être une valeur de la liste suivante: NO_BARO , MS5611, GY86 , BMP085 , BMP180 , BMP280
Vous devez spécifier le type de capteur utilisé pour le second vario avec la ligne #define SECOND_BARO_SENSOR_USE NO_BARO
La valeur à renseigner doit être soit NO_BARO soit MS5611
Les deux capteurs sont connectés en parallèle (utilisent les pins A4 et A5 Arduino).
Le premier capteur (MS5611 ou BMPxxx) répond sur l'adresse 0x77 du bus I2C.
/!\ ATTENTION ne pas oublier de connecté la PIN SDO à VDD sur le BMP280 pour obtenir l'adresse 0x77

S'il est utilisé, le second capteur (un MS5611) doit avoir la PIN CE connecté à VDD. Comme cela l'adresse I2C sera 0x76. (Note: par défaut la pine CE du MS5611 est connectée à GND ainsi son adresse I2C est 0x77)
Note : Quand un seul capteur est connecté, il doit être connecté comme premier capteur.

4.2 - Type de vitesse verticale transmise

oXs peut calculer différentes vitesses verticales et permet de choisir celle doit être transmise.
Selon le contenu de la ligne #define VSPEED_SOURCE, la vitesse verticale sera basée sur
- le premier capteur barométrique (= cas normal)
- le deuxième capteur barométrique
- la moyenne des deux capteurs barométriques
- le premier capteur barométrique et la compensation par la vitesse air (avec tube de pitot/Prandtl) (= dtE)
- le premier capteur barométrique + le module IMU (accéléromètre + gyroscope)
- soit un premier soit un type de vitesse verticale en fonction du signal PPM envoyé par l'émetteur

****

#define VSPEED_SOURCE FIRST_BARO // select between FIRST_BARO, SECOND_BARO , AVERAGE_FIRST_SECOND, AIRSPEED_COMPENSATED , BARO_AND_IMU or PPM_SELECTION

4.3 - Définition de la sensibilité par programmation

Quand un vario est utilisé, oXs peut prendre en compte certains paramètres pour ajuster la sensibilité
La sensibilité peut être réglée entre les valeurs:
- 20 (réglage conservatif, temps de réaction = plusieurs secondes)
- 150 (rapide mais beaucoup d'erreurs, temps de réaction = bien inférieur à la seconde)

50 est une valeur "normale" pour mesurer de petites vitesses verticales (recherche d'ascendances en planeur); ce réglage donne une réaction autour de la seconde.
La sensibilité peut être réglée par programmation et/ou par l'émetteur.

Si vous voulez une réponse très rapide du vario la meilleure solution est d'utiliser un module gy-86 qui comprend un capteur MPU6050 (accéléromètre/gyroscope) en complément du capteur barométrique (voir ci-dessous).
Note : quand le vario est utilisé (#define VARIO dé-commenter), vous devez spécifier (dans la section 9) comment la vitesse verticale est calculée.

Le réglage de la sensibilité utilise 4 paramètres:
SENSIVITY_MIN = Cette sensibilité est la sensibilité de base du vario.
Ce paramètre est donc le paramètre principal de contrôle de la sensibilité du vario.
Cette valeur est utilisée par oXs quand la différence entre la valeur de la Vitesse verticale actuelle (calculée toutes les 20 ms) et la précédente valeur de la vitesse verticale est inférieure à la valeur définie par SENSITIVITY_MIN_AT ; la valeur typique est 40.
SENSIVITY_MAX = c'est la sensibilité maximale utilisée quand la vitesse verticale change très rapidement.
Cette valeur est utilisée par oXs quand la différence entre la valeur de la Vitesse verticale actuelle et la précédente valeur de la vitesse verticale est supérieure à la valeur définie par SENSITIVITY_MAX_AT ; la valeur typique est 300.
SENSITIVITY_MIN_AT = vitesse verticale pour laquelle SENSITIVITY_MIN est applicable (voir ci-dessus) (cm/s); la valeur typique est 100 (cm/s).
SENSITIVITY_MAX_AT = vitesse verticale pour laquelle SENSITIVITY_MAX est applicable (voir ci-dessus) (cm/s); la valeur typique est 1000 (cm/s).

Entre ces valeurs, la sensibilité est calculée par interpolation entre SENSITIVITY_MIN et SENSITIVITY_MAX.
Ces paramètres permettent à oXs d'avoir un temps de réaction rapide quand la vitesse verticale change rapidement et d'avoir un vario silencieux dans des conditions calmes.
Note: SENSITIVITY_MAX peut-être égale au paramètre SENSITIVITY_MIN si aucune adaptation automatique n'est désirée en cas de changement rapide.

****

```
#define SENSITIVITY_MIN 50
#define SENSITIVITY_MAX 300
#define SENSITIVITY_MIN_AT 100
#define SENSITIVITY_MAX_AT 1000
```

4.4 - Modification de la sensibilité depuis l'émetteur

La sensibilité peut être ajustée depuis l'émetteur grâce à un interrupteur ou un potentiomètre et quelques mixages sur une voie de sortie servo du récepteur.
Pour utiliser cette option, une sortie servo doit être connectée à oXs (voir « Configuration de PPM » dans la section 3.
Quand oXs reçoit un signal de l'émetteur, la valeur SENSIVITY_MIN définie dans la section 4.3 est négligée et remplacée par une valeur calculée sur base des paramètres ci-dessous.

Le réglage utilise 4 paramètres :
SENSITIVITY_MIN_AT_PPM = quand l'émetteur envoie cette valeur sur la sortie PPM, le paramètre ENSITIVITY_MIN (voir section 4.2) est remplacé par la valeur du paramètre SENSITIVITY_PPM_MIN; Une valeur typique peut être 10.
SENSITIVITY_MAX_AT_PPM = quand l'émetteur envoie cette valeur sur la sortie PPM, le paramètre SENSITIVITY_MAX (voir section 4.2) est remplacé par la valeur du paramètre SENSITIVITY_PPM_MAX; Une valeur typique peut être 40.
Les paramètres SENSIVITY_PPM_MIN+MAX définissent la plage dans laquelle vous aller pouvoir ajuster la sensibilité :
SENSITIVITY_PPM_MIN = valeur minimum pour le paramètre SENSITIVITY_MIN qui peut lui être assigné par PPM; Une valeur typique peut être 20.
SENSITIVITY_PPM_MAX = valeur maximum pour le paramètre SENSITIVITY_MAX qui peut lui être assigné par PPM; Une valeur typique peut être 100.
SENSITIVITY_MIN est automatiquement interpoler entre le paramètre SENSITIVITY_PPM_MIN et le paramètre SENSITIVITY_PPM_MAX selon le signal PPM reçu.
la sensibilité change seulement si la valeur du signal PMM est comprise entre SENSITIVITY_MIN_AT_PPM - 5 et SENSITIVITY_MAX_AT_PPM + 5.
/!\ note importante : la sensibilité est changée sur base de la valeur absolue du signal PPM; donc PPM = -25 a le même effet que PPM = 25.
Ceci est important pour pouvoir combiner un changement de sensibilité avec une sélection du type de vitesse verticale (ceci permet de passer d'un vario non compensé à un vario compensé sur l'émetteur).

****

```
#define SENSITIVITY_MIN_AT_PPM 10
#define SENSITIVITY_MAX_AT_PPM 40
#define SENSITIVITY_PPM_MIN 20
#define SENSITIVITY_PPM_MAX 100
```

4.5 - Hystérésis et compensation de l'altitude en fonction de la température du capteur de pression

oXs peut appliquer un hystérésis sur les changements de vitesse verticale.
Ceci permet de ne pas envoyer trop souvent un changement de vitesse verticale (ce qui peut provoquer de fausses alertes sonores). C'est un autre moyen de réduire la sensibilité.
VARIOHYSTERESIS veux dire que la vitesse verticale change si la vitesse verticale mesurée (après filtrage) diffère de la valeur précédemment transmise de plus que la valeur de ce paramètre.
Une valeur typique peut être 5 (= 5 cm/s); 0 veux dire pas d'hystérésis.

oXs permet aussi d'avoir une compensation depuis le capteur MS5611 (pas depuis le capteur BMPxxx) afin de compenser la dérive de l'altitude due à l'augmentation de température interne du capteur après sa mise en route.
La température interne du composant MS5611 augmente de plusieurs degrés lors de sa lecture par l'Arduino.
En principe, le capteur MS5611 a des paramètres de compensation pour pallier à l'augmentation de sa température interne.
Pourtant il semble que ces paramètres de correction ne soit pas toujours optimum.
oXs permet d'ajouter une compensation supplémentaire pour pallier ce problème. Pour cela vous devez remplir le paramètre ALT_TEMP_COMPENSATION.
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.
Les informations suivantes peuvent vous être utiles:
- 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.

Note importante: donner une valeur au paramètre ALT_TEMP_COMPENSATION ne va pas éliminer totalement la dérive d'altitude car:
- cela peut être le résultat d'une variation de pression atmosphérique (celle-ci ne peut pas être compensée)
- la dérive peut venir d'autres facteurs
- la compensation n'est pas linéaire sur toutes les valeurs de pression et de température

****

#define VARIOHYSTERESIS 5

4.6 - Choix du type de vitesse verticale en fonction du signal PPM (optionnel)

Quand vous utiliser 2 capteurs barométriques ou un capteur barométrique et un capteur de vitesse Pitot (4525D0 - voir section 5) ou un capteur barométrique et un IMU, oXs peut calculer différentes vitesses verticales (ou dTE).
Quand l'option PMM est activée, oXs permet de choisir entre 2 types de vitesse verticale (vitesse vertical du premier ou second capteur ou compensé par le Pitot, etc...) celle qui doit être envoyée à l'émetteur.
Ceci permet de changer sur l'émetteur entre un vario compensé et un non compensé.
Bien que oXs puisse calculer jusqu'à 5 types de vitesse verticale (FIRST_BARO, SECOND_BARO , AVERAGE_FIRST_SECOND, AIRSPEED_COMPENSATED , BARO_AND_IMU), il est uniquement possible de changer entre 2 vitesses verticales de son choix.
Pour activer cette option, plusieurs paramètres sont nécessaires,
1) Spécifier le type de vitesse vertical qui sera la primaire et la secondaire en utilisant les lignes ci-dessous:
#define VARIO_PRIMARY XXXXXXX
#define VARIO_SECONDARY YYYYYYY
Ou XXXXXX et YYYYYY est une valeur sélectionnée dans la liste suivante: FIRST_BARO, SECOND_BARO , AVERAGE_FIRST_SECOND, AIRSPEED_COMPENSATED , BARO_AND_IMU
2) Spécifier la plage de valeurs PPM qui déterminent le fait que oXs doit envoyer la vitesse verticale "primaire" ou la "secondaire" dans les lignes ci-dessous.
#define SWITCH_VARIO_MIN_AT_PPM 10
#define SWITCH_VARIO_MAX_AT_PPM 90
Quand la valeur absolue de PPM est entre le paramètre SWITCH_VARIO_MIN_AT_PPM (valeur typique = 10) et le paramètre SWITCH_VARIO_MAX_AT_PPM (valeur typique = 90),
- oXs vas switcher sur la vitesse verticale primaire si la valeur de PPM est positive
- oXs vas switcher sur la vitesse verticale secondaire si la valeur de PPM est négative
Note: Quand la valeur absolue the PPM est hors de la plage de valeur, oXs ne va pas changer de type de calcul de vitesse vertical et va rester avec le type actuel.
Ce principe permet d'utiliser un interrupteur en même temps qu'un potentiomètre pour contrôler la sensibilité.
Le passage de positif à négatif sur openTx peut être réalisé à l'aide d'un mixage "MULTIPLY by -100%".
L'envoi d'une valeur PPM hors de cette plage de valeurs permet d'appliquer une autre fonction (exemple la remise à zéro de offset de la vitesse air) sans changer le calcul de la vitesse vertical.
3) Spécifier dans la section 4.2 que la sélection de la source de vitesse verticale est "PPM_SELECTION"

****

#define VARIO_PRIMARY AIRSPEED_COMPENSATED // sélectionner une valeur parmi: FIRST_BARO, SECOND_BARO , AVERAGE_FIRST_SECOND, AIRSPEED_COMPENSATED , BARO_AND_IMU
#define VARIO_SECONDARY FIRST_BARO // sélectionner une valeur parmi: FIRST_BARO, SECOND_BARO , AVERAGE_FIRST_SECOND, AIRSPEED_COMPENSATED , BARO_AND_IMU
#define SWITCH_VARIO_MIN_AT_PPM 10
#define SWITCH_VARIO_MAX_AT_PPM 90

4.7 - vitesse verticale analogique (optionnel)

oXs peut aussi donner la vitesse verticale (uniquement celle du capteur 1) sous forme de signal analogique.
Ceci peut être utilisé par exemple sur des récepteurs Frsky ayant des entrées analogiques mais pas les protocoles HUB ou SPORT.
Ceci peut être utile si votre récepteur ne possède pas de communication digital (ou si celle-ci est déjà utilisé par un autre capteur)

/! Des composent supplémentaires (1 résistance et 1 condensateur) sont nécessaires! Lire le Wiki si vous voulez utiliser cette option.
Pour activer cette option:
- dé-commenter les trois paramètres suivant
- définir dans le paramètre PIN_ANALOG_VSPEED la sortie Arduino DIGITAL qui va être utilisée (voir ci-dessous)
Par défaut 3; la PIN utilisée ne peut être que la PIN 3 ou 11 car la fonction Arduino timer 2 doit être utilisée par oXs.
- définir les limites min et max de la vitesse verticale (en mètre/sec)
- ANALOG_VSPEED_MIN : une vitesse verticale inférieure ou égale va envoyer 0 Volt au récepteur
- ANALOG_VSPEED_MAX : une vitesse verticale supérieure ou égale va envoyer 3.2Volt au récepteur

****

```
#define PIN_ANALOG_VSPEED 3
#define ANALOG_VSPEED_MIN -3
#define ANALOG_VSPEED_MAX 3
```

4.8 - Calcul de la finesse du planeur, du taux moyen de montée/descente, de l'altitude moyenne gagnée/perdue

oXs peut calculer et envoyer certaines données quand la vitesse air et la vitesse verticale restent stables (dans une certaine tolérance) plus de X secondes (exemple entre 5 à 10 sec)
Les valeurs calculées sont :
- le temps écoulé depuis le début du calcul de la finesses du planeur et de taux moyen de montée/descente.
- le taux moyen de montée/descente (= différence d'altitude / temps écoulé)
- finesse planeur (= distance / différence d'altitude) (en fait = vitesse air temps écoulé / différence d'altitude)
Le calcul de la finesse planeur est un paramètre qui permet d'affiner les réglages du planeur. Cela n'a de sens que si les vitesses (verticale et air) sont stables.
oXs effectue ces calculs uniquement quand:
- la vitesse air ne change pas d'un certain pourcentage (comparé à la vitesse au début du temps écoulé). Ce pourcentage peut être défini par l'utilisateur dans le paramètre SPEED_TOLERANCE.
- la vitesse verticale reste dans les limites définies par l'utilisateur dans les paramètres VSPEED_MIN_TOLERANCE et VSPEED_MAX_TOLERANCE
Toutes les 0.5sec, oXs vérifie que les mesures instantanées restent dans les tolérances définies.
Si les mesures effectuées par oXs sont hors des tolérances définies, oXs arrête le calcul en cours, et reprend avec de nouvelles mesures. Finesse du planeur et taux moyen de montée/descente sont remis à 0.
Si les mesures d'oXs restent dans les tolérances définies par l'utilisateur au moins pendant dans le temps défini par le paramètre GLIDER_RATIO_CALCULATED_AFTER_X_SEC, oXs procède au calcul.
Note : Dans cette version d'oXs, si vous voulez envoyer les paramètres calculés, vous devez faire une configuration qui demande l'envoi des paramètres de la façon suivante:
- TEST1 pour le temps écoulé (en 1/10 de sec)
- TEST2 pour la vitesse verticale moyenne (en cm/sec comme la vitesse verticale)
- TEST3 pour la finesse du planeur (en 1/10 d'unité)
Le calcul de finesse du planeur nécessite un tube de Pitot et un vario;
Le taux moyen de montée/descente peut être calculé sans sonde de Pitot(mais la correction par la vitesse ne pourra pas ce faire)
Afin d'activer la fonction vous devez dé-commenter et remplir les lignes suivantes.

Pour désactiver la fonction il vous suffit de mettre en commentaire la ligne suivante:
GLIDER_RATIO_CALCULATED_AFTER_X_SEC

****
#define GLIDER_RATIO_CALCULATED_AFTER_X_SEC 10 // Temps écoulé pour effectuer le calculer et le transmettre(en sec)
#define SPEED_TOLERANCE 5 // En % de la vitesse
#define VSPEED_MIN_TOLERANCE -200 // Hors de la tolérance quand la vitesse vertical est inferieur a cette valeur (cm/sec)
#define VSPEED_MAX_TOLERANCE -10 // Hors de la tolérance quand la vitesse vertical est supérieur a cette valeur (cm/sec)
****

Note : Il n'est pas utile de mettre en commentaire les paramètres de sensibilité, hystérésis etc. quand un vario n'est pas utilisé (ces données seront juste négligées lors de la compilation)
Les paramètres de sensibilité, hystérésis sont communes aux deux capteurs barométriques

5 - Configuration de la vitesse air (optionnel)

oXs peut mesurer la différence de pression entre une pression statique et une pression dynamique grace à une sonde de Prandtl (ou plus souvent connue sous le nom de sonde de Pitot)
Les meilleurs résultats sont obtenus avec un capteur 4525DO-DS5AI001DP.
Il est à noter que ce capteur fonctionne à Vcc = 5 Volt et qu'il peut mesurer des vitesses jusqu'à 360 km/h.
Il existe d'autre version du capteur 4525DO fonctionnant en Vcc = 3.3 Volt ou permettant de mesurer de plus forte pressions donc de plus haute vitesse.
Le Capteur 4525DO est connecté à L'Arduino en parallèle avec le capteur barométrique MS5611 (en utilisant le bus I2C et les pin **A4 et A5** en plus de Vcc et Ground)
Il est recommandé d'ajouter un condensateur (100nf) le plus près possible du 5424DO entre Ground et Vcc.
Note : A faible vitesse air (exemple sous 10 km/h), la différence de pression à mesurer est très faible ce qui la rend moins fiable.
Le capteur 4525DO-DS5AI001DP est assez difficile à se procurer et est assez coûteux.
Une alternative consiste à utiliser un capteur MPXV7002 combiné à un ADC ADS1115. Voir la section ADS1115 ci-dessous.

Dans la ligne #define AIRSPEED_SENSOR_USE vous devez définir le type de capteur que vous utilisez
Sélectionner une valeur parmi: NO_AIRSPEED , MS4525 , MPXV7002

Le capteur de pression nécessite une calibration afin de retourner la valeur "0" quand la vitesse air est nulle.
oXs calibre automatiquement le capteur quand il est mis sous tension (en considérant que la valeur lue à ce moment correspond à une vitesse nulle)
Pour cela, il est important de ne pas exposer au vent la sonde de Prandtl (Pitot) lors du démarrage de oXs.
Malgré cela, une dérive peut être observée lors de l'échauffement du capteur.
oXs permet de calibrer le capteur depuis l'émetteur via la fonction PPM (voir section 3).
Le paramètre AIRSPEED_RESET_AT_PPM défini la valeur que doit envoyer l'émetteur pour forcer la re-calibration du capteur.
La valeur par défaut est 100.
Le plus simple est de configurer l'envoi d'une impulsion par l'émetteur (par exemple envoyer la valeur 100 pendant 1 seconde).

/!\ Note : l'ordre de re-calibration doit être envoyé uniquement quand la vitesse air est vraiment nulle sinon cela génèrera de fausses mesures.

oXs peut calculer deux types de vitesse air:
- la première est une vitesse air basée sur la densité de l'air à 15°C et 11325 hPa (au niveau de la mer). C'est la vitesse air utilisée dans l'aéronautique.
Exemple la vitesse de décrochage ne varie pas avec l'altitude
- La seconde tient compte de la pression (fourni par le capteur barométrique) et de la température interne du capteur afin de calculer la "vraie" vitesse (à comparer avec la vitesse GPS par vent nul).
La vitesse air normalisée est calculée quand la ligne #define AIRSPEED_AT_SEA_LEVEL_AND_15C n'est pas en commentaire. Pour avoir la vraie vitesse air, mettre cette ligne en commentaire.

oXs peut envoyer la vitesse air en (1/10) de knot/h ou en (1/10) km/h. Pour OpenTx, vous devez utiliser l'option knot/h (seule certaines vielles versions ont besoin de l'option km/h).
Pour activer l'option km/h, il faut dé-commenter la ligne #define AIRSPEED_IN_KMH
Sinon mettre cette ligne en commentaire pour activer l'option knot/h.

oXs peut aussi utiliser la mesure de vitesse air pour calculer le variomètre compenser appeler PRANDTL_DTE (=delta total energy).
Aller voir sur le web pour plus d'information a propos dTE (= electronically compensated vario).
Le principe)e est d'essayer de détecter uniquement les vrai ascendance en neutralisant les effet de la gouverne de profondeur.
Normalement, DTE doit être transmis par le paramètre vitesse vertical (vitesse verticale = Valeur par défaut) car OpenTX ne supporte pas encore un champ spécifique pour celui-ci.

Lorsque l'option PPM est implémentée, il est également possible d'ajuster le facteur de compensation entre 2 valeurs.
Cela nécessite de dé-commenter 4 paramètres:
COMPENSATION_MIN_AT_PPM Spécifiez la valeur PPM afin de définir la compensation à la valeur spécifiée par COMPENSATION_PPM_MIN; Défaut = 60
COMPENSATION_MAX_AT_PPM Spécifiez la valeur PPM afin de définir la compensation à la valeur spécifiée par COMPENSATION_PPM_MAX; Défaut = 90
COMPENSATION_PPM_MIN Compensation minimale; en % ; Par défaut 80
COMPENSATION_PPM_MAX 120 Compensation maximal; en % ; Par défaut 120

****
#define AIRSPEED_SENSOR_USE MS4525
#define AIRSPEED_AT_SEA_LEVEL_AND_15C // Si cette ligne est commentée, la vitesse est calculée à l'aide de la pression et de la température du capteur barométrique (donc une vitesse "vraie" au lieu de la vitesse normale).
//#define AIRSPEED_IN_KMH // Décommentez cette ligne si la vitesse doit être en km / h au lieu de nœud / h (sauf certaines anciennes versions, OpenTx s'attend à nœud / h)
#define AIRSPEED_RESET_AT_PPM 100
#define COMPENSATION_MIN_AT_PPM 60
#define COMPENSATION_MAX_AT_PPM 90
#define COMPENSATION_PPM_MIN 80
#define COMPENSATION_PPM_MAX 120

6 - Configuration du capteur de tension et de courant (optionnel)

6.1 - Tension de référence pour mesurer des tensions et des courants

Le courant et les tensions peuvent être mesurés en (1023) pas soit par rapport à VCC, soit par rapport à une référence interne d'environ 1.1Volt soit par rapport à une référence externe.
Si VCC est très stable, il est probablement plus précis et plus facile de mesurer le courant et les tensions sur base de VCC (donc en 1023 pas de VCC).
Pourtant, cela exige que la tension appliquée sur la broche "RAW" d'Arduino soit régulée ou au moins toujours supérieure à 5,5 volts (afin de laisser le régulateur propre à la carte Arduino assurer la régulation à 5 volts).
Si la tension sur la broche "Raw" est inférieure à 5,5 volts et change (par exemple en raison de la consommation des servos ou de la tension de la batterie), les mesures de courant et de tension seront fausses.
Si VCC ne peut être très stable (par exemple, Arduino est alimenté par le RX sous 4.8 Volt NiMh) et vous n'avez besoin que des tensions (pas besoin de mesure de courant), il est possible de mesurer en fonction de la référence de tension interne Arduino (1.1 volt).
Si vous avez besoin également d'une mesure de courant, l'utilisation de la tension interne (1.1 volt) n'est pas la solution, car le capteur de courant a besoin d'une tension stable aussi.
Veillez à ce que les diviseurs de tension (voir ci-dessous) soient configurés pour que la tension de la broche analogique Arduino ne dépasse pas VCC ou 1,1 volt ou la tension externe en fonction de l'option que vous choisissez.
- Dé-commentez la "#define USE_INTERNAL_REFERENCE" pour activer la référence à la tension interne 1.1 volt (sinon, les mesures seront basées sur VCC).
- Dé-commentez la "#define USE_EXTERNAL_REFERENCE" pour activer la référence de tension externe (sinon, les mesures seront basées sur VCC).
Note : pour utiliser une référence de tension externe, vous devez connecter une tension de référence à la pin AREF votre Arduino. Ceci est assez difficile avec l'Arduino pro mini car la PIN AREF n'est pas disponible.
- Pour obtenir les meilleures mesures, vous pouvez configurer la référence de tension en utilisant la ligne #define REFERENCE_VOLTAGE.
La valeur doit être définie en millivolt (par exemple 5100 pour 5,1 volts) et ne pas contenir de décimales ;
Si cette valeur est omise, oXs appliquera automatiquement 1100 (lorsque USE_INTERNAL_REFERENCE est défini) et 5000 dans les autre cas.
Lorsqu'une référence externe est utilisée, REFERENCE_VOLTAGE doit être spécifié.

****

```
//#define USE_INTERNAL_REFERENCE
//#define USE_EXTERNAL_REFERENCE //Dé-commentez cette ligne si vous utilisez une référence externe au lieu de Vcc
#define REFERENCE_VOLTAGE 5000
```

6.2 - Configuration des tensions

oXs peut mesurer jusqu'à 6 tensions d'entrée avec l'Arduino ADC (veuillez noter que chez certains fabricants, les Arduino pro mini ont moins de broches analogiques disponibles).
Pour mesurer les tensions, vous:
- Devez spécifier YES dans la ligne ARDUINO_MEASURES_VOLTAGES
- Devez spécifier les broches analogiques (A0 à A7) connectées à une batterie (par exemple un lipo multicellulaire) ou à un capteur analogique (par exemple, un capteur de température qui transforme la température en une tension qui elle peut être mesurée)

- Devez spécifier les valeurs des résistances utilisées pour les ponts diviseurs de tension (voir ci-dessous)
- Pouvez spécifier un décalage et/ou une mise à l'échelle à appliquer.

Note : Une broche analogique peut également être utilisée pour mesurer un courant à l'aide d'un capteur de courant; La configuration d'un capteur de courant est décrite dans la section 6.4 (voir ci-dessous);
N'utilisez pas la même broche analogique pour mesurer une tension et un courant.
/!\ Attention: ne pas utiliser les broches A4 et A5 si vous utilisez un vario ou un capteur de vitesse air ou un ads1115 (ces broches sont réservées au bus I2C de ces capteurs).

/!\ Attention :

- si la masse est déjà référencée pour l'alimentation au travers du récepteur, ne pas brancher la masse pour la mesure des éléments de batteries. Si la masse est branchée sur la prise d'équilibrage, lorsque l'on débranche le connecteur de puissance, tout reste sous tension. (Le récepteur, les servos, et la télémétrie)
- Dans le cas d'utilisation d'Oxs avec une tension de 5V, il est possible de raccorder l'alimentation du module sur la broche Vcc de la carte Arduino.
- Dans tous les autres cas, en particulier si la tension d'alimentation est supérieure à 5v, il faudra alimenter le module par la broche RAW de la carte Arduino.

Si vous ne mesurez pas de voltages avec l'Arduino ADC, mettez NO dans la ligne ARDUINO_MEASURES_VOLTAGES
Les broches utilisées pour mesurer certaines tensions sont définies dans la ligne #define PIN_VOLTAGE. Lorsqu'elle est utilisée, cette ligne doit contenir 6 valeurs (séparées par des virgules); la première valeur est utilisée pour mesurer VOLT_1, la deuxième VOLT_2, ... jusqu'à VOLT_6
Chaque valeur doit être un nombre de 0 à 7 (0 signifie A0 = pin analogique 0, 1 signifie A1, ... 7 signifie A7) ou la valeur "8" (quand la tension ne doit pas être mesurée)
Note : les mêmes valeurs de broches analogiques peuvent être utilisées dans plusieurs tensions (par exemple pour VOLT1 et VOLT6).

Note : Si les tensions de chaque élément d'un LiPo doivent être mesurées, il FAUT commencer à partir de VOLT1 pour 1s, VOLT2 pour 2s ....., d'autres tensions peuvent être mesurées mais elles doivent être définies après celles correspondant au nombre de cellules à mesurer
Et le nombre max d'éléments **DOIT** être spécifié dans "numberofcells" (voir la section suivante)
Les numéros de broche Ax peuvent ne pas être consécutives (par exemple, la cellule 1 peut être connectée à A2, la cellule 2 à A0, ...)

/! Veillez à ce que la tension appliquée à la broche Arduino ne dépasse pas Vcc (normalement 5 volts) ou 1,1 volt (si la tension de référence interne est utilisée).
Il se peut que vous deviez utiliser un diviseur de tension afin de réduire la tension appliquée sur la broche Arduino par rapport à la tension que vous souhaitez mesurer.
Pour chaque tension à abaisser aux bornes Ax de oXs, procédez comme suit:
- Faire un diviseur avec 2 résistances

------> Point final = connexion à l'appareil à mesurer (batterie, capteur de courant, ...)

|
__|__
| |
| | R2 (Resistor_To_Voltage)
| |
|_____|
|
|------> Point milieu = connexion à la broche Arduino A0, A1, A2, A3, A6 ou A7
__|__
| |
| | R1 (Resistor_To_Ground)
| |
|_____|
|
------> Connection à Ground

Note : Un condensateur (par exemple 100nf) pourrait également être ajouté entre le ground et le point milieu afin d'obtenir des valeurs plus stables.

- R1 et R2 sont choisis pour s'assurer que la tension appliquée à l'Arduino est proche de (mais ne dépasse jamais) VCC ou 1.1 volt ou la tension de référence selon votre choix en ce qui concerne la mesure de courant et de tension (voir ci-dessus).
- la tension max sur la pin Arduino = VCC (ou 1.1 volt ou la référence externe) = "tension maximale à mesurer à partir de ce capteur" R1 / (R1 + R2)
- R1 pourrait être de 10 kOhm; Donc R2 = R1 (("tension maximale à mesurer à partir de ce capteur" / VCC [ou 1.1 volt ou la référence]) - 1)
Par exemple, en utilisant 1.1 volt comme référence de tension, si on veut mesurer max 6 volts et R1 = 10000, alors R2 = 10000 ((6 / 1.1) - 1) = 45545 Ohm ;
Il faut alors arrondir à la valeur supérieure disponible, par exemple 47000 ohm.

Les résistances utilisées dans les diviseurs de tension doivent être spécifiées dans les lignes #define RESISTOR_TO_GROUND (=R1) et #define RESISTOR_TO_VOLTAGE (=R2).
Chaque ligne, doit contenir 6 valeurs (une pour chaque tension à mesurer); Chaque valeur représente la résistance en Ohm ou kOhm mais il faut utiliser la même unité pour les deux lignes.
Si un diviseur n'est utilisé pour une certaine tension, mettez les deux valeurs de résistances à 0 (zéro)
Si aucun diviseur n'est utilisé, les lignes peuvent être mises en commentaires (en ajoutant "//" à l'avant)

Afin d'obtenir des mesures de tension plus précises, l'oXs peut être calibré. Ce processus facultatif permet de compenser les tolérances sur les résistances et sur l'Arduino ADC (convertisseur analogique / numérique).
Pour étalonner chaque mesure de tension, procédez comme suit afin de trouver les meilleures valeurs à mettre dans les lignes #define OFFSET_VOLTAGE et #define SCALE_VOLTAGE
- Préparer un test
- Laisser OFFSET_VOLTAGE = 0 et SCALE_VOLTAGE = 1 (ce réglage n'ajoute aucune compensation)
- Sélectionner un champ pour transmettre la tension souhaitée (par exemple T1 pour VOLT3) et remplir la ligne "#define SETUP_DATA_TO_SEND" en conséquence
- Charger le programme dans Arduino

- Appliquer plusieurs tensions différentes sur le point final (**/!** ne dépasser pas la tension maximale autorisée en fonction de R1 et R2)
- Pour chaque tension appliquée, mesurer la tension appliquée avec un voltmètre et lire la valeur reçue sur le panneau de télémétrie sur l'émetteur
- Mettre ces valeurs dans Excel (ou sur un graphique) et calculer les meilleures valeurs OFFSET_VOLTAGE et SCALE_VOLTAGE (à l'aide d'une régression linéaire)
Si cela semble trop complexe, utilisez simplement 2 tensions aussi différentes que possible (mais dans la gamme des valeurs normales que vous souhaitez mesurer)
Et appliquer un calcul basé sur l'exemple suivant :
Je m'attends à ce que la tension soit normalement entre 4 volts et 6 volts, donc j'applique 2 tensions proches de ces valeurs au point final
- Pour la première tension, le voltmètre donne 3510 millivolt et la télémétrie donne 3622
- Pour la deuxième tension, le voltmètre donne 5900 millivolt et la télémétrie donne 6013
Alors SCALE_VOLTAGE = (5900-3510) / (6013-3622) = 0.99958
et OFFSET_VOLTAGE = -3510 + (3622 0.99958) = 110
Note: Vous pouvez appliquer le même principe si vous mesurez autre chose qu'une tension.
Par exemple. Vous pouvez calibrer un capteur de température et régler le paramètre d'étalonnage afin d'obtenir une mesure en degré sur l'affichage de l'émetteur.

Les lignes #define OFFSET_VOLTAGE et #define SCALE_VOLTAGE sont facultatives (elles peuvent être mises en commentaire en ajoutant "//" à l'avant))
Si elle est définie, chaque ligne doit contenir 6 valeurs. Utilisez 0 (pour le décalage => OFFSET) et 1 (pour l'échelle => SCALE) si aucun étalonnage n'est effectué pour une certaine tension.

****
VOLT1 VOLT2 VOLT3 VOLT4 VOLT5 VOLT6
#define ARDUINO_MEASURES_VOLTAGES YES
#define PIN_VOLTAGE 2 , 0 , 2 , 3 , 8 , 8
#define RESISTOR_TO_GROUND 12 , 20 , 30 , 40 , 50 , 60 // Mettre la valeur à 0 lorsqu'aucun diviseur n'est utilisé pour une de ces tensions
#define RESISTOR_TO_VOLTAGE 50, 100.1 , 200, 300 , 500 , 600 // Mettre la valeur à 0 lorsqu'aucun diviseur n'est utilisé pour une de ces tensions
#define OFFSET_VOLTAGE 0 , 0 , 0 , 0 , 0 , 0 // Peut être négatif, doit être un nombre entier
#define SCALE_VOLTAGE 1 , 1 , 1 , 1 , 1 , 1 // Peut être négatif, peut avoir des décimales

6.3 - Maximum de cellules LiPo à mesurer (et à envoyer à l'émetteur)

Les différentes tensions mesurées comme expliqué dans la section 6.2 sont toutes référencées à Ground.
oXs peuvent utiliser certains d'entre elles pour calculer par différence la tension de cellules lipo individuelles.
Définissez ici le nombre maximal de cellules lipo que vous souhaitez mesurer/transmettre; La valeur peut être 0 (sans cellules), 1,2,3,4,5,6. *La valeur peut aussi être a 0 si on ne veut que la valeur global de la batterie définie dans Vfas et pas les tensions éléments par éléments.*
Si une valeur supérieure à 1 est définie, l'oXs calculera la tension de chaque cellule en fonction de la différence entre 2 tensions successives à partir de la tension1.
Dans openTx, l'émetteur affichera la tension totale dans un champ de télémétrie nommé "Cells".
L'émetteur identifiera également la cellule avec la tension la plus basse et l'affichera dans un champ nommé "Cell".
L'émetteur à également un écran spécial où toutes les tensions s'affichent (voir le manuel Taranis).

Par exemple. Si nombre de cellules = 3,
La tension sur la cellule 1 sera la tension mesurée sur la première broche définie dans PIN_Voltage
La tension sur la cellule 2 sera la différence entre la mesure des tensions sur la deuxième broche et la première broche (donc VOLT2 - VOLT1)
La tension sur la cellule 3 sera la différence entre la mesure des tensions sur la troisième broche et la deuxième broche (donc VOLT3 - VOLT2)
Etc.
Pour mesurer les tensions des cellules, vous ne devez pas oublier de configurer PIN_VOLTAGE, RESISTOR_TO_GROUND, RESISTOR_TO_VOLTAGE (et optionnellement les paramètres d'étalonnage).
Les broches utilisées pour mesurer des cellules DOIVENT être mises au début et en séquence (VOLT1 pour la première cellule, VOLT2 pour la deuxième, ...)
Les broches non utilisées peuvent être utilisées pour transmettre d'autres tensions (par exemple: un capteur de température)
Par exemple. Si NUMBEROFCELLS = 3, la première broche (dans la liste de 6) doit être connectée à la cellule 1 (via un diviseur de tension calculé pour environ 4,5 volts)
La deuxième broche doit être connectée à la cellule 2 (via un diviseur de tension calculé pour environ 9 volts)
La troisième broche doit être connectée à la cellule 3 (via un diviseur de tension calculé pour environ 13 volts)
D'autres broches peuvent encore être utilisées pour d'autres données (température, courant, ...)
Remarques: Vous devez utiliser des diviseurs de tension pour réduire les tensions sur chaque broche de la boucle d'équilibrage lipo
Si vous utilisez la référence interne 1.1 et si R1 = 10 kOhm pour chaque diviseur, alors R2 pourrait être
33 kOhm pour la tension1, 68 kOhm pour la tension2, 120 kOhm pour la tension 3 et 150 kOhm pour la tension4
Si vous utilisez 5V Vcc comme référence, vous ne devez pas utiliser de diviseur pour Volt1 et si pour les autres voltages vous utilisez R1 (resistor to ground) = 10 kOhm. Alors R2 (resistor to voltage) peut être
8.7 kOhm pour Voltage2, 22 kOhm pour Voltage3 et 27 kOhm pour voltage 4

Veuillez noter que plus vous disposez de cellules, plus la marge d'erreur augmente si vous ne calibrez pas les tensions.
Si vous ne souhaitez pas transmettre de tension de cellule, réglez la valeur sur 0 (zéro) ou mettez la ligne en commentaire.
Ce paramètre définit le nombre maximal de cellules que vous attendez à transmettre.
Si oXs est connecté à un lipo ayant moins de cellules que défini, oXs réduira automatiquement le nombre de cellules ce qui permet à l'émetteur de continuer à calculer la tension totale et la tension de cellule la plus basse

****

#define NUMBEROFCELLS 3

// 6.4 - Conversion de tension en température (° Celsius)

Pour mesurer une température, il est possible d'utiliser un composant électronique spécial (comme le LM35) qui génère une tension proportionnelle à la température
Dans ce cas, vous pouvez simplement connecter la sortie de ce composant à une broche analogique Arduino et configurer oXs comme décrit dans la section 6.2 afin de mesurer la tension
En ajoutant un décalage "OFFSET" et une mise à l'échelle "SCALE" (voir la section 6.2) en fonction de la caractéristique de votre composant, oXs peut calculer directement la température correspondant à la tension mesurée.

Vous pouvez ensuite demander à oXs de transmettre cette tension dans le champ de télémétrie souhaité (par exemple, T1) en remplissant la configuration comme expliqué dans la section 2.

Ce type de composant (ex LM35) n'est pas prévu pour mesurer une température élevée (par exemple supérieure à 150 ° C)
Vous pouvez alors utiliser des thermistances (= NTC).
L'inconvénient des NTC est que la tension générée n'est pas proportionnelle à la température et nécessite des calculs supplémentaires.
oXs vous permet d'utiliser une ou plusieurs NTC (par exemple une par cylindre) afin de mesurer de haute(s) température(s).
Pour chaque NTC, vous devez ajouter une résistance en série suivant le schéma ci-dessous:

------> < Arduino Vcc >
|
__|__
| |
| | [serie resistor]
| |
|____|
|
|------> <Arduino analog pin>
__|__
| |
| | [NTC]
| |
|____|
|
------> <ground>

Si vous utilisez plusieurs NTC, toutes les NTC et les résistances doivent être identiques et doivent être connectées à des broches analogiques consécutives (tel que défini dans PIN_VOLTAGE)

La configuration donc être spécialement adaptée pour que oXs calcule correctement.
Par conséquent, aux sections 6.1 et 6.2,
USE_INTERNAL_REFERENCE doit être mis en commentaire (donc avec // en avant)
USE_EXTERNAL_REFERENCE doit être mis en commentaire (donc avec // en avant)
REFERENCE_VOLTAGE doit être mis en commentaire (donc avec // en avant)
RESISTOR_TO_GROUND doit être mis à 0 (pour l'index utilisé)
OFFSET_VOLTAGE doit (normalement) doit être mis à 0 (pour l'index utilisé)
SCALE_VOLTAGE doit être configuré sur 204.6 (= 1000 1023/5000) (pour l'index utilisé)
Ensuite, vous devez définir certains paramètres dans la section 6.4
FIRST_NTC_ON_VOLT_NR spécifie l'indice de la première tension utilisée pour la conversion en température (par exemple, 3 signifie VOLT_3) (donc ce n'est pas le code de la broche analogique, celui-ci est défini dans la section 6.2)
LAST_NTC_ON_VOLT_NR spécifie l'indice de la dernière tension utilisée pour la conversion en température (par exemple, 5 signifie VOLT_5)

Dans cet exemple, cela signifie que vous prévoyez de mesurer 3 températures basées sur NTC connectées aux broches utilisées pour VOLT_3, VOLT_4 et VOLT_5
Note: si vous utilisez un seul NTC, vous devez spécifier la même valeur pour FIRST_NTC_ON_VOLT_NR et pour LAST_NTC_ON_VOLT_NR
Si vous n'utilisez pas NTC, mettez cette ligne en commentaire
SERIE_RESISTOR spécifie le résistance (en Ohm) connecté entre Arduino Vcc et NTC (et la broche analogique); Sélectionnez une valeur presque égale à la résistance de la NTC dans la gamme de température où vous voulez obtenir la meilleure précision
Les 3 paramètres suivants sont spécifiques à la NTC que vous utilisez. Les valeurs entre parenthèses sont les valeurs nominales pour un NTC bon marché disponible sur aliexpress.com avec 100k ohms à 25 ° C et un coefficient bêta de 3950 pour la gamme 25/50.
STEINHART_A (e.g. 7.00111E-4)
STEINHART_B (e.g. 2.1644E-4)
STEINHART_C (e.g. 1.0619E-07)
Si vous ne connaissez pas ces 3 paramètres, vous pouvez les calculer en mesurant la résistance NTC à 3 températures différentes. Utiliser, par exemple, La formule donnée sur Wikipédia pour la thermistance
Lorsque vous utilisez 1 ou 2 NTC, la température (s) sera enregistrée dans VOLT_X et VOLT_Y où X est la valeur de FIRST_NTC_ON_VOLT_NR et Y la valeur de LAST_NTC_ON_VOLT_NR
Si vous utilisez plus de 2 NTC, oXs inscrit dans:
VOLT_X la température la plus basse
VOLT_X + 1 l'indice de la température la plus élevée (1 étant l'indice du premier NTC)
VOLT_Y la température la plus élevée
Vous pouvez alors définir comment transmettre ces données dans la section 2

****

//#define FIRST_NTC_ON_VOLT_NR 5 // Décommettez cette ligne lorsque la thermistance est utilisée; Spécifiez l'indice de la première tension utilisée pour la conversion en température (par exemple, 5 signifie VOLT_5)
#define LAST_NTC_ON_VOLT_NR 6 // Spécifiez l'indice de la dernière tension utilisée pour la conversion en température (par exemple 6 signifie VOLT_6)
#define SERIE_RESISTOR 4700 // Résistance connectée à Arduino Vcc (en Ohm)
#define STEINHART_A 7.00111E-4
#define STEINHART_B 2.1644E-4
#define STEINHART_C 1.0619E-07

6.5 – Configuration ou la référence externe. du capteur de mesure de courant.

Il est possible de mesurer un courant (et une consommation de courant) si un capteur de courant est connecté.
La connexion d'un capteur de courant est une option.
Il nécessite un matériel supplémentaire. Il peut s'agir d'un IC comme le ACS712 (pour 5, 20, 30 ampères) ou le ACS758 (pour 50, 100, 150, 200 ampères).
La plupart des capteurs sont bidirectionnels, mais le ACS758 de type "U" ne peut mesurer qu'un courant unidirectionnel (fournissant alors une sensibilité plus élevée).
Ces capteurs de courant sont assez bon marché (voir par exemple ebay ou aliexpress) et renvoient une tension proportionnelle au courant. Cette tension est mesurée par oXs via une broche analogique.
La valeur de la PIN à remplir dans oXs_config_advanced.h est un nombre de 0 à 7 (0 signifie A0, 1 signifie A1, ... 7 signifie A7).

Si un capteur de courant est utilisé, ne pas utiliser une broche déjà utilisée pour mesurer une tension.
/! Veillez à ce que la tension appliquée à la broche de l'Arduino ne dépasse pas Vcc (normalement 5 volts) ou 1,1 volt (si la tension de référence interne est utilisée)
Il se peut que vous deviez utiliser un diviseur de tension afin de réduire la tension appliquée à la broche Arduino.
Voir l'explication ci-dessus (paragraphe 6.2) sur le diviseur de tension.
Attention : ne pas utiliser les broches A4 et A5 si vous utilisez un vario ou un capteur de vitesse ou un ads1115 (ces broches sont réservées aux capteurs avec bus i2C).
Remarque : Le capteur de courant est normalement alimenté par le VCC 5 volts d'oXs (identique au capteur barométriques).
Il existe des capteurs bidirectionnels et des capteurs unidirectionnels.
Pour les capteurs bidirectionnels, la sortie est normalement égale à VCC / 2 lorsque le courant = 0 Amp et, pour les unidirectionnels, la sortie est normalement de 0,6 volt pour 0 Amp.
Si oXs est connecté à une batterie qui ne dépasse pas 5,2 volts, la tension d'alimentation du capteur de courant varie avec la tension d'alimentation oXs.
Par conséquent, VCC / 2 (= O amp) varie avec VCC.
Ceci est un problème si l'Arduino ADC est configuré pour utiliser la référence interne de 1.1 volts avec un capteur bidirectionnel.
Donc, dans ce cas, il est préférable de configurer l'ADC afin d'utiliser VCC comme référence pour la conversion.
Pour utiliser un capteur de courant, vous devez mettre YES dans la ligne ARDUINO_MEASURES_A_CURRENT et vous devez spécifier dans PIN_CURRENTSENSOR la broche Arduino connectée au capteur utilisé.
Vous devez également définir 2 paramètres en fonction du type de capteur utilisé; Ces paramètres sont donnés dans la fiche technique du capteur.
- MVOLT_AT_ZERO_AMP = MilliVolt généré par le capteur lorsque le courant est 0 Amp: la valeur normale est:
- Pour un capteur bidirectionnel: Vcc du capteur de courant / 2 (donc = 2500 si le capteur est connecté à Arduino Vcc et Arduino Vcc est de 5 Volt).
- 600 pour capteur unidirectionnel
- MVOLT_PER_AMP = MilliVolt par Amp. La valeur dépend de la sensibilité du capteur (par exemple, un ACS712ELCTR-30A-T a une sensibilité de 66 mvolt/Amp, un ACS758LCB-050U a une sensibilité de 60 mvolt/Amp)

Si vous utilisez la référence interne 1.1 volt pour mesurer le voltage et le courant, vous devez également utiliser un diviseur de tension pour réduire la tension produite par le capteur actuel.
Voir la section 6.2 ci-dessus sur le diviseur de tension. Le principe est tout simplement le même, mais les noms des 2 paramètres sont:
- RESISTOR_TO_GROUND_FOR_CURRENT
- RESISTOR_TO_CURRENT_SENSOR
Remarque : ces paramètres sont automatiquement négligés lorsque ARDUINO_MEASURES_A_CURRENT est NO
Remarque: Lorsqu'un capteur de courant est utilisé, oXs peut également calculer et transmettre la consommation de courant (milliAh) et le "carburant restant" (fuel) (en % descendant de 100% à 0%).
Si vous voulez ce dernier, utilisez une configuration comme "Fuel , MILLIAH , -100 , 4000 ,0" dans la section "données à transmettre" (et remplacez 4000 par la capacité - en milliAmph - de votre batterie) (voir ci-dessous) .
Avec les émetteurs utilisant le logiciel OpenTx ou Ersky9x, il est préférable de laisser l'émetteur calculer ces valeurs en fonction du courant.

Cela garantit que les valeurs sont cohérentes et permet de réinitialiser les valeurs du côté émetteur; Il permet de changer la valeur de la capacité de la batterie du côté de émetteur (donc sans devoir recharger une autre configuration oXs dans Arduino).
Par exemple: avec Ersky9x, dans le menu de télémétrie configuré "courant source" défini "FAS"; Dans «Alarme mAh», réglez le mah que vous désirez pour que l'alarme sonne et sélectionnez le son d'avertissement / la voix, Pour n'utiliser que 70% d'une lipo de 2200 mAh, utilisez 1540 comme capacité. Le pourcentage de FUEL commencera à 100% et descendra 0% lorsque 1540 sont consommés.

****

```
#define ARDUINO_MEASURES_A_CURRENT NO
#define PIN_CURRENTSENSOR 2
#define MVOLT_AT_ZERO_AMP 600
#define MVOLT_PER_AMP 60
#define RESISTOR_TO_GROUND_FOR_CURRENT 10
#define RESISTOR_TO_CURRENT_SENSOR 40
```

6.6 - Paramètres de Ads1115

Il est possible de connecter un ADC externe du type ads1115.
Ce composant est très bon marché (environ 2 €) et peut fournir plus de précision que l'ADC interne de Arduino.
Il a une résolution de 16 bits, une référence de tension interne précise, un amplificateur de gain programmable et la capacité de mesurer directement la différence de tension entre 2 broches
Voir la fiche technique de ADS1115 pour plus de détails
oXs peut être connecté à l'un de ces composant via le bus I2C. Il peut alors fournir jusqu'à 4 mesures de tension appelées ADS_VOLT_1 ... ADS_VOLT_4
oXs permet de convertir une des mesures de tension en courant et consommation (lorsque l'ads1115 est connecté à un capteur de courant)
oXs permet également de convertir une des mesures de tension en vitesse et vario compensé (lorsque l'ads1115 est connecté à un capteur baro différentiel comme le MPXV7002)
Le capteur MPXV7002 est une alternative moins coûteuse et plus facile à trouver que le capteur MS4525 (mais elle est moins précise)
Pour utiliser un ads1115 ADC, vous devez sélectionner YES dans la ligne AN_ADS1115_IS_CONNECTED (dans oXs_config_basic.h) et spécifier plusieurs groupes de 4 paramètres.
/!\ Attention: mettez NO AN_ADS1115_IS_CONNECTED si un ads1115 n'est pas utilisé (afin d'éviter les erreurs I2C et de ralentir oXs)
Dans ACD_MEASURE, vous spécifiez les broches ads115 utilisées pour les mesures de tension
Remplissez toujours les 4 valeurs. Pourtant, si vous n'avez pas besoin de toutes les 4 mesures, utilisez la valeur ADS_OFF pour les mesures non requises
Note : Plus vous demandez de mesures, plus il faudra de temps pour obtenir chacune d'elles car elles sont mesurées l'une à la suite de l'autre
Sélectionnez 4 valeurs parmi: A0_TO_A1, A0_TO_A3, A1_TO_A3, A2_TO_A3, A0_TO_GND, A1_TO_GND, A2_TO_GND, A3_TO_GND, ADS_OFF
Dans ADC_FULL_SCALE_VOLT, vous spécifiez le paramètre de gain ads1115 pour chacune des 4 mesures.
Remplissez toujours 4 valeurs même si vous n'avez pas besoin des 4 mesures
Cela permet d'amplifier une faible tension appliquée sur les broches d'entrée avant qu'il ne soit converti par l'ADC. La précision de la conversion est donc optimale.

Sélectionnez entre MV6144 MV4096 MV2048 MV1024 MV512 MV256 où les chiffres donnent le mvolt maximal appliqué sur la broche (par exemple pour A0_TO_GND) ou entre 2 broches (par exemple pour A0_TO_A1)
Dans ADS_SCALE, vous spécifiez un facteur d'échelle à appliquer sur chaque mesure afin d'obtenir une valeur conforme vos attentes.
Remplissez toujours 4 valeurs même si vous n'avez pas besoin des 4 mesures
Lorsque le paramètre de mise à l'échelle = 1, oXs renvoie une valeur = 1 lorsque la tension appliquée sur la/les PIN de ads1115 est la tension maximale définie par ADC_FULL_SCALE_VOLT.
Donc, par exemple, si ADC_FULL_SCALE_VOLT est réglé sur MV512, lorsque la tension d'entrée sera de 512mv (ou plus), oXs renverra 1 si ADS_SCALE = 1
Si vous ne disposez pas d'un diviseur de tension sur l'ads1115, vous vous attendez probablement à ce que oXs renvoie 512; définissez alors ADS_SCALE à 512.
Si vous avez un diviseur de tension, la tension que vous souhaitez mesurer est supérieure à la tension appliquée sur la broche de l'ads1115 et vous devez augmenter ADS_SCALE.
Par exemple. Si votre diviseur de tension divise votre tension par un facteur 10, vous devez définir ADS_SCALE à 5120 (= 512 10)
Note : ADS_SCALE peut avoir des décimales (par exemple, 100.5). Il peut être positif ou négatif; Il ne peut pas être 0
Dans ADS_OFFSET, vous spécifiez un décalage à appliquer.
Remplissez toujours 4 valeurs même si vous n'avez pas besoin des 4 mesures
Lorsqu'aucun décalage ne doit être appliqué, mettez la valeur à 0
Le décalage spécifié est ajouté à la valeur calculée après la mise à l'échelle
Chaque valeur doit être un nombre entier (positif ou négatif); Il peut être 0
Dans ADS_RATE, vous spécifiez le nombre de milli sec que l'Ads1115 prend pour convertir une tension.
Remplissez toujours 4 valeurs même si vous n'avez pas besoin des 4 mesures.
L'utilisation d'une valeur élevée réduit la consommation d'énergie, mais elle réduit le nombre de mesures pouvant être effectuées / transmises par seconde
Sélectionnez les valeurs entre MS137, MS69, MS35, MS18, MS9, MS5, MS3, MS2; Les chiffres correspondent au nombre de milli sec (par exemple MS18 signifie 18 ms)
Remarque: oXs attendra au moins le délai spécifié, mais il se peut que le délai soit plus élevé en raison d'autres tâches devant être exécutées par oXs
Dans ADS_AVERAGING_ON, vous spécifiez le nombre de mesures de tensions successives à effectuer afin de calculer une moyenne qui elle sera exploitée.
Remplissez toujours 4 valeurs, même si vous n'avez pas besoin des 4 mesures. Si vous ne désirez pas de moyenne, définissez la valeur sur 1
Cela doit être un nombre entier, positif et différent de 0.
Note : L'augmentation de la valeur est un moyen de réduire le bruit (et donc d'augmenter la précision), mais il augmente le délai entre 2 transmissions
Dans ADS_CURRENT_BASED_ON, vous spécifiez quelle mesure de tension (le cas échéant) est utilisée pour calculer un courant (et la consommation actuelle)
Dé-commentez cette ligne uniquement lorsqu'un capteur de courant est connecté à l'ads1115.
Gardez cette ligne en tant que commentaire s'il n'y a pas de capteur de courant ou si le capteur est connecté à une broche Arduino comme expliqué à la section 6.4
Remplissez une seule valeur; Sélectionnez une valeur parmi ADS_VOLT_1, ADS_VOLT_2, ADS_VOLT_3, ADS_VOLT_4
Note : Lorsqu'oXs calcule un courant basé sur un ads1115, il utilise également 2 paramètres de la section 6.4: MVOLT_AT_ZERO_AMP et MVOLT_PER_AMP

Dans ADS_AIRSPEED_BASED_ON, vous spécifiez quelle mesure de tension (le cas échéant) est utilisée pour calculer la vitesse air
Dé-commentez cette ligne uniquement lorsqu'un capteur de pression différentielle analogique est connecté à l'ads1115
Gardez cette ligne comme un commentaire s'il n'y a pas de capteur de pression connecté à l'ads1115
Remplissez une seule valeur; Sélectionnez une valeur entre ADS_VOLT_1, ADS_VOLT_2, ADS_VOLT_3, ADS_VOLT_4
Note : Un capteur de vitesse typique est le MPXV7002DP qui est disponible sur ebay ou aliexpress.com. Lorsque ce capteur est utilisé, vous devez configurer l'ads1115 de la manière suivante
- Demandez une seule mesure à l'ads1115 et celle-ci doit être une tension différentielle (entre 2 pins de l'ads1115): utilisez donc cette configuration: ADS_MEASURE A0_to_A1, ADS_OFF, ADS_OFF, ADS_OFF
- Connectez une résistance d'environ 10kohm entre ads1115 Vcc et ads1115 A1 et une autre de la même valeur entre ads1115 Ground et ads1115 A1; Donc la broche A1 est à Vcc/2 volt
- Définissez le gain d'ads1115 pour obtenir 2048 mvolt à pleine échelle: utilisez donc: ADC_FULL_SCALE_VOLT MV2048, MV2048, MV2048, MV2048
- Définissez le taux de rafraîchissement de l'ads afin de faire la conversion ADC le plus rapidement possible; utilisez donc: ADS_RATE MS2, MS2, MS2, MS2
- Les autres paramètres ne sont pas critiques (non utilisés pour la vitesse air)

****

#define AN_ADS1115_IS_CONNECTED NO // Sélectionnez parmi YES et NO
#define ADS_MEASURE A1_TO_GND , ADS_OFF , ADS_OFF , ADS_OFF // Si un ads1115 est utilisé, sélectionnez 4 valeurs parmi A0_TO_A1, A0_TO_A3, A1_TO_A3, A2_TO_A3, A0_TO_GND, A1_TO_GND, A2_TO_GND, A3_TO_GND, ADS_OFF
#define ADS_FULL_SCALE_VOLT MV4096, MV4096, MV4096, MV4096 // Sélectionnez parmi MV6144 MV4096 MV2048 MV1024 MV512 MV256
#define ADS_OFFSET 0, 0 , 0 , 0 // Doit être un nombre entier (positif ou négatif)
#define ADS_SCALE 2, 10, 1, 1 // Peut est un float (avec décimales)
#define ADS_RATE MS137 , MS5, MS3 , MS2 // Sélectionnez parmi MS137, MS69, MS35, MS18, MS9, MS5, MS3, MS2
#define ADS_AVERAGING_ON 1 , 10, 50, 50 // Nombre de valeurs utilisées pour la moyenne (doit être compris entre 1 et 254)
#define ADS_CURRENT_BASED_ON ADS_VOLT_1 // Dé-commentez si le courant et la consommation doivent être calculés en fonction de l'une des mesures de tension ADS; Sélectionnez ensuite la tension à utiliser parmi ADS_VOLT_1, ADS_VOLT_2, ADS_VOLT_3, ADS_VOLT_4
#define ADS_AIRSPEED_BASED_ON ADS_VOLT1 // Dé-commentez si la vitesse (et dte) doit être calculée en fonction de l'une des mesures de tension ADS; Sélectionnez ensuite la tension à utiliser parmi ADS_VOLT_1, ADS_VOLT_2, ADS_VOLT_3, ADS_VOLT_4

7 - Configuration du capteur RPM (tour par minute) (optionnel)

Il est possible de mesurer une fréquence de rotation (RPM) à l'aide d'un capteur connecté sur la PIN ICP (= PB0, = broche 8) d'oXs
Ce capteur doit fournir un changement de niveau (0 - Vcc) sur cette broche par exemple chaque fois qu'une pale passe devant le capteur.
Le nombre de pales est un paramètre important à configurer dans le paramètre PULSES_PER_ROTATION.
Toutefois pour le protocole Frsky, ce paramètre est négligé car le paramètre est introduit dans le Tx

Il est également possible de construire une petite carte PCB qui fournira des impulsions lorsqu'elle est connectée à un moteur brushless.
Pour activer cette fonction, mettez YES au lieu de NO dans la ligne #define CALCULATE_RPM .
Note : La broche numérique 8 (PB0 / ICP) est la seule à être utilisée pour mesurer les RPM.
Attention : Pour le protocole Frsky la valeur calculée par oXs est en Hertz (et non en tour par minute) et la conversion est faite dans le Tx

****

#define CALCULATE_RPM NO
#define PULSES_PER_ROTATION 2

8 - Configuration de la mémoire persistante (=non volatile) (optionnel)

Fonction optionnelle.
Si la mémoire persistante est activée, la consommation de courant et la consommation de carburant (+ paramètres de flux) seront stockées dans EEPROM toutes les 30 secondes.
Cette valeur sera restaurée chaque mise sous tension.
Ainsi, vous obtiendrez une consommation continue même si vous éteignez le modèle entre les vols.
Si vous souhaitez enregistrer ces données, vous devez dire OUI dans la ligne "#define SAVE_TO_EEPROM"

Il est également possible de réinitialiser:
- la consommation de courant et de carburant à zéro en appuyant sur un bouton-poussoir connecté à oXs.
Pour cela une broche DIGITAL Arduino doit être connectée à un bouton-poussoir, l'autre broche du bouton-poussoir étant connectée à Gnd (terre).
Pour utiliser cette fonction, vous devez spécifier la broche DIDITAL Arduino utilisée.
Par défaut: 10; Une autre broche numérique peut être utilisée; N'utilisez pas une broche déjà utilisée dans un autre but.
- la consommation de carburant à zéro à partir de l'émetteur en connectant une voie du récepteur à l'Arduino à l'aide de la fonction PPM (voir la section 3 PPM et 11 capteur de débit)

****

#define SAVE_TO_EEPROM NO
#define PIN_PUSHBUTTON 10

9 - GPS (optionnel)

Il est possible de connecter un module GPS à Arduino. Dans ce cas, oXs transmettra à l'émetteur certaines données générées par le module GPS.
Note : les données sont envoyées uniquement lorsque le GPS a un "fix" (est connecté à plusieurs satellites).
Cela peut donc prendre plusieurs minutes avant que oXs ne commence à envoyer des données GPS.
Si le GPS perd sa connexion avec les satellites, oXs arrête d'envoyer des données GPS jusqu'à ce que le GPS obtienne une nouvelle connexion satellite.
Lorsqu'un récepteur Frsky (SPORT ou HUB) est utilisé, oXs enverra toujours la longitude, la latitude, l'altitude, la vitesse du sol et le parcours.
Cela ne nécessite aucune ligne supplémentaire dans la section "Données à transmettre" (voir 2 ci-dessus)
Lorsqu'un récepteur multiplex est utilisé, l'utilisateur doit définir sous la ligne "#define SETUP_MULTIPLEX_DATA_TO_SEND" les données à envoyer et le numéro de la ligne sur laquelle les données doivent apparaître à l'écran.
Points d'attention concernant le type de matériel:

Modules GPS pris en charge: oXs prend en charge les modules GPS utilisant un GPS UBLOX (facilement disponible sur ebay ou aliexpress) comme Neo6M, Neo7M et Neo8M.
La plupart des modules sont équipés d'un régulateur de tension afin de ramener la tension à 3,3 volts, soit environ le maximum autorisé par le GPS
Ils ont 4 broches disponibles qui doivent être connectés à Arduino
- GPS Ground est connecté à Arduino Ground
- GPS Vcc est normalement connecté à la broche Raw Arduino (qui est normalement connectée à Vcc du récepteur)
Prendre soin que le régulateur de tension sur le module GPS supporte la tension maximale appliquée à la broche Raw. La plupart des modules GPS supportent 6 volts (alors que la broche RAW arduino accepte elle plus de 6V).
Donc, si votre broche RAW reçoit plus de 6 volts, il est prudent d'ajouter un autre régulateur de tension pour réduire la tension GPS Vcc.
- La broche GPS Tx est connectée à la broche Arduino Rx
- La broche GPS Rx est connectée à une résistance (par exemple 10k) et l'autre broche de la résistance est connectée à la broche 6 d'Arduino (broche digitale 6).
Cette résistance est ajoutée (au moins pour un Arduino 5 volts) afin de protéger la broche GPS.
Ceci est nécessaire car l'Arduino génère un signal avec un niveau haut égal à Arduino Vcc (donc normalement 5 volts) alors que le module GPS ne supporte en principe pas plus que 3.3 Volt.
Pour être plus sûr, vous pouvez même ajouter une deuxième résistance entre GPS Rx pin et Ground (valeur = 22k) mais, dans mon cas, cela a fonctionné sans cette deuxième résistance.
Note : Il serait possible d'utiliser une autre broche que la broche 6 d'Arduino, mais il faut modifier certains paramètres dans le fichier oXs_gps.cpp (voir "Configuration du capteur GPS").
/!\ Note importante :
La broche Arduino Rx a 2 utilités: obtenir les données du GPS et télécharger le programme dans l'Arduino (normalement fait une seule fois) à l'aide d'un adaptateur USB/série.
Vous devez éviter d'utiliser le GPS et l'USB en même temps, car lors du téléchargement d'un programme dans Arduino, il y aura des conflits entre les signaux envoyés par les 2 appareils (le PC et le GPS) et la programmation échouera.
Il existe un autre risque: si votre adaptateur USB vers série génère un signal de 5 volts, il pourrait endommager le module GPS.
Donc, lorsque vous connectez l'adaptateur série à l'Arduino, vous devez déconnecter au moins la broche GPS TX de la broche Arduino Rx.
Personnellement, j'utilise un connecteur entre l'Arduino et le module GPS et je peux donc déconnecter complètement le module GPS.
Points d'attention concernant le logiciel:
Le module GPS UBLOX est normalement livré avec une configuration par défaut (générant automatiquement, par exemple, des messages NMEA à 9600 bauds à un taux de rafraîchissement de 1 hz).
oXs suppose que, au démarrage, le GPS fonctionne à 9600 bauds. oXs envoie alors quelques commandes pour :
- Désactiver tous les messages NMEA
- Activer certains messages UBX (format spécifique à UBLOX)
- Augmenter la fréquence de calcul (à 5 Hz au lieu de 1 Hz)
- Configurez le débit à 38400 bauds au lieu de 9600 bauds.
Ces paramètres ne sont pas enregistrés dans le GPS (car certains modules GPS ne le permettent pas). Ainsi, oXs enverra ces commandes à chaque mise sous tension.

Si oXs n'émet pas de données GPS, vérifiez que votre module GPS a encore la configuration par défaut (le plus important est qu'il soit configuré pour recevoir des messages de commande UBX à 9600 bauds).
Un moyen simple de vérifier la configuration GPS est de connecter le module GPS à un FTDI de 3,3 volts (ou à un adaptateur USB/série) et d'utiliser un logiciel gratuit nommé "u-center".
Ce logiciel est disponible sur le site officiel de UBLOX. Plus d'informations sont facilement disponibles sur le Web.
oXs permet de modifier certains paramètres dans le fichier config.h:
- #define A_GPS_IS_CONNECTED NO : Remplacez NO par YES si un GPS est connecté et doit transmettre ses données
- #define GPS_SPEED_IN_KMH : Dé-commentez cette ligne si la vitesse GPS doit être envoyée en km / h au lieu de nœud / h (note: le protocole Frsky exige une valeur en nœud, la conversion en km/h se fait dans l'émetteur)
- #define GPS_SPEED_3D : Décommentez cette ligne si la vitesse GPS doit être la vitesse 3d au lieu de la vitesse 2d (note: 3d est probablement moins précis - à tester)
- #define GPS_REFRESH_RATE 5 // taux de rafraîchissement des données du GPS; sélectionnez parmi 1, 5 or 10 (Hz). Par défaut = 5 Hz; Ublox NEO6 ne supporte pas le 10 Hz

****

#define A_GPS_IS_CONNECTED YES // Sélectionnez parmi YES , NO
//#define GPS_SPEED_IN_KMH // Dé-commentez cette ligne si la vitesse GPS doit être envoyée en km/h au lieu de noeud/h (ne pas utiliser pour les protocoles FRSKY, HOTT et JETI car c'est le TX qui fait la conversion)
#define GPS_SPEED_3D // Décommentez cette ligne si la vitesse GPS doit être la vitesse 3d au lieu de la vitesse 2d (note: 3d est probablement moins précis - à tester)
#define GPS_REFRESH_RATE 5 // taux de rafraîchissement des données du GPS; sélectionnez parmi 1, 5 or 10 (Hz). Par défaut = 5 Hz; Ublox NEO6 ne supporte pas le 10 Hz

10 - IMU 6050 (capteur accéléromètre/gyroscope) (optionnel) et HMC5883 (magnétomètre))

10.1 - IMU 6050

Il est possible de connecter un capteur IMU (= accéléromètre / gyro) à l'Arduino; C'est optionnel.
Cela permet :
- de réduire le temps de réaction du vario d'environ 0,5 sec (note: un capteur baro doit également être connecté parce que oXs fusionne les données des deux capteurs)
- de transmettre des données sur les accélérations et / ou l'orientation (pitch / roll); dans ce cas, il est important que le dispositif oXs soit monté dans une position correcte et soit aligné avec l'axe de vol de l'avion.
Le seul capteur IMU qui est pris en charge est le: mpu6050.
Ce capteur est facilement disponible sous forme de différents modules. Le meilleur module à utiliser est probablement le GY-86 car il dispose également d'un régulateur de tension (3.3 volts), d'un convertisseur de niveau I2C et d'un capteur baro (MS5611)
5 broches du mpu6050 doivent être connectées à l'Arduino:
- MP6050 ground <---> Arduino ground
- MP6050 Vcc <---> Arduino Vcc
- MP6050 SDA <---> Arduino SDA = Arduino A4
- MP6050 SCL <---> Arduino SCL = Arduino A5
- MP6050 INT <---> Arduino INT0 = Arduino 2 ou Arduino INT1 = **Arduino 3** (n'utilisez pas la même broche pour une autre fonction, comme PPM!)

Pour activer l'IMU, mettez YES dans la ligne
#define A_MPU6050_IS_CONNECTED
Lorsque l'IMU est activée, oXs peut calculer la vitesse verticale d'autre manière en fusionnant l'altitude du capteur baro avec l'accélération verticale de l'IMU (par rapport à la Terre).
Cet autre type de vitesse verticale peut être envoyé comme vitesse verticale en choisissant la valeur "BARO_AND_IMU" dans la ligne
#define VSPEED_SOURCE (voir section 4.2)
Il est également possible de l'affecter à "VARIO_PRIMARY" ou "VARIO_SECONDARY" et donc de basculer entre 2 sources au départ du Tx (en utilisant un canal ppm)
Afin d'obtenir les meilleurs résultats avec l'IMU, il est conseillé de calibrer les "offset" de l'accéléromètre. Pour ce faire, il vous faut:
- Chargez une version d'oXs en veillant à ne pas mettre la ligne #define DISPLAY_ACC_OFFSET en commentaire
- Lancez oXs alors qu'il est connecté au PC (via l'interface série USB = FTDI)
- Ouvrir le terminal IDE Arduino (appuyez simultanément sur CTRL + MAJ + M)
- Veillez à régler le débit du terminal IDE à 115200 bauds (ou 38400 bauds si le GPS est aussi activé)
- Après le démarrage, le terminal doit, toutes les 2 ou 3 s, afficher Acc suivi par 3 nombres respectivement AccX, AccY et AccZ. Notez que ces chiffres changent lorsque le mpu6050 se déplace.
- Assurez-vous que le mpu6050 (GY86) est parfaitement horizontal et ne se déplace pas (par exemple, le mettre sur une table)
- Notez les 2 premiers nombres (= AccX et AccY); Ne tenez pas compte du 3ème nombre car lorsque le capteur est dans cette position, il reflètera la gravité et sera autour de 16384.
- Tournez le mpu6050 afin de mettre l'axe X ou Y parfaitement vertical et ne vous déplacez pas. Maintenant, le 3ème nombre sera beaucoup plus petit (car il ne mesurera plus la gravité)
- Notez le 3ème nombre (= Accz)
- Mettez à jour le fichier oXs_config_advanced.h en remplissant les 3 nombres dans les lignes #define ACC_OFFSET_X, #define ACC_OFFSET_Y et #define ACC_OFFSET_Z
- Mettez la ligne #define DISPLAY_ACC_OFFSET en commentaire (ajoutant "//" à l'avant)
- Télécharger à nouveau le firmware oXs dans Arduino

****
#define A_MPU6050_IS_CONNECTED NO // Remplacez NO by YES si un IMU6050 est connecté et doit transmettre ses données
#define PIN_INT_6050 3 // La broche INT0 de l'IMU6050 doit être connecté à la broche 2 ou 3 de l'Arduino(n'utilisez pas la même broche que pour PPM)
#define DISPLAY_ACC_OFFSET
#define ACC_OFFSET_X 0 // Reportez ici la valeur affichée quand DISPLAY_ACC_OFFSET est activé (eg. -160)
#define ACC_OFFSET_Y 0 // Reportez ici la valeur affichée quand DISPLAY_ACC_OFFSET est activé (eg. -150)
#define ACC_OFFSET_Z 0 // Reportez ici la valeur affichée quand DISPLAY_ACC_OFFSET est activé (eg. -1100)

10.2 - HMC5883

Si vous utilisez un module comme GY-86 et si l'IMU6050 est connecté à l'Arduino (voir 10.1), oXs peut accéder à un magnétomètre HMC5883 afin d'obtenir une mesure Yaw.
Cela ne nécessite pas de câblage supplémentaire.
Pour activer le HMC5883, réglez YES dans la ligne CALCULATE_YAW_WITH_HMC5883
Veillez à obtenir des valeurs fiables du magnétomètre, il est obligatoire de le calibrer.
Il s'agit d'un processus non trivial.

Je recommande de suivre le processus décrit dans ce lien: **http://www.physi.cz/hmc5883l-magnetometer-calibration.html**
Cela signifie que vous devez:
1) Collecter des données d'oXs:
Cela exige que vous:
- Dé-commentiez la ligne #define GENERATE_MAG_CALIBRATION_DATA
- Téléchargiez le programme dans l'Arduino
- Ouvriez le terminal PC Arduino
- Dans le terminal PC, vous obtiendrez quelques lignes de données générales suivies d'une liste avec 3 valeurs par ligne (il s'agit des lectures X, Y et Z du magnétomètre)
- Tournez le capteur lentement sur 360 ° le long de tous les axes afin de collecter plus de 1000 lignes de données (cela peut prendre plusieurs minutes)
2) Copier et coller les lignes du terminal PC vers un fichier TXT (en utilisant un éditeur de texte comme bloc-notes). Supprimez les premières lignes de données générales (avant la liste) et enregistrez ce fichier en tant que fichier TXT
3) Téléchargez le programme Windows dans le lien ci-dessus (voir "Téléchargement exécutable")
Exécutez ce programme, réglez la norme en 1.0, cliquez sur "ouvrir" et sélectionnez le fichier TXT avec vos données d'échantillons, puis cliquez sur "Calibrer"
4) Notez les 3 valeurs "biais combiné" et copiez-les dans XMAG_OFFSET, YMAG_OFFSET et ZMAG_OFFSET (dans oXs_config_advanced.h)
5) Notez les 9 "facteurs d'échelle combinés ..." et ouvrez maintenant un lien vers un site Web qui vous permettra de trouver l'inverse de la matrice des 9 valeurs:
https://www.wolframalpha.com/input/?i=%7B%7B591.0437,-13.1628,-15.0294%7D,%7B-13.1628,596.1147,30.5314%7D,%7B-15.0294,30.5314,552.0759%7D%7D%5E-1
- Remplissez les 9 valeurs dans le premier champ de saisie (respecter le format spécial avec {{,,}, {,,}, {,,}} et cliquez sur le bouton à droite de la ligne d'entrée.
6) Vous obtiendrez une matrice "résultat". Notez les 9 valeurs du résultat et copiez-les dans XXMAG_CORRECTION, XYMAG_CORRECTION, XZMAG_CORRECTION ... ZZMAG_CORRECTION (dans oXs_config_advanced.h)
7) Définissez la ligne #define GENERATE_MAG_CALIBRATION_DATA comme commentaire et téléchargez encore le programme à Arduino

****

```
#define CALCULATE_YAW_WITH_HMC5883 NO // Sélectionnez YES ou NO; YES exige aussi que A_MPU6050_IS_CONNECTED soit à YES (voir ci-dessus)
//#define GENERATE_MAG_CALIBRATION_DATA // Dé-commentez cette ligne lorsque l'étalonnage HMC5883 doit être effectué. Remmettez la ligne en commentaire une fois que les paramètres d'étalonnage ont été introduits
#define XMAG_OFFSET 2 // Doit être un nombre entier (= sans décimales)
#define YMAG_OFFSET -1 // Doit être un nombre entier (= sans décimales)
#define ZMAG_OFFSET 139 // Doit être un nombre entier (= sans décimales)
#define XXMAG_CORRECTION 0.122082 // Peut avoir des décimales
#define XYMAG_CORRECTION -0.00204026
#define XZMAG_CORRECTION 0.00377534
#define YXMAG_CORRECTION -0.00204026
#define YYMAG_CORRECTION 0.130413
#define YZMAG_CORRECTION -0.00491189
```

#define ZXMAG_CORRECTION 0.00377534
#define ZYMAG_CORRECTION -0.00491189
#define ZZMAG_CORRECTION 0.138038

11 - Flow sensor

Si vous utilisez un moteur à essence, vous pouvez connecter un débitmètre à oXs
Ce capteur génère une impulsion chaque fois que quelques milli litres de carburant passent à travers le capteur
oXs peut compter le nombre d'impulsions et calcule 3 données: la consommation courante milli litres / min et, en prenant soin de la capacité du réservoir, le carburant restant en ml et en%.
Il est recommandé d'utiliser le capteur de débit suivant, car il est probablement plus précis que d'autres lorsque le débit est faible
http://www.conrad.be/ce/nl/product/155374/BIO-TECH-eK-FCH-M-Doorstroomsensor-1-stuks-Voedingsspanning-bereik-5-24-VDC-Meetbereik-08-0015-lmin-l-x;jsessionid=EED7B26A7F28BA3F20F0060807E20FD1.ASTPCEN22?ref=searchDetail
Il est prévu pour mesurer de 0,015 à 0,8 ml / min.
Le signal de sortie du débitmètre doit être connecté à la **broche 9 de l'Arduino** (et les 2 autres broches à 5 volts VCC et à Grnd).
Il existe d'autres capteurs de débit bon marché sur ebay ou aliexpress, mais je pense qu'ils n'offrent pas assez de précision lorsque le débit est faible.

Pour activer le débitmètre, vous devez:
- Affecter YES au paramètre A_FLOW_SENSOR_IS_CONNECTED (dans le fichier oXs_config_basic.h)
- Spécifier dans PULSES_PER_ML le nombre d'impulsions générées par le capteur lorsque 1 milli litre de liquide s'écoule à travers
- Spécifier dans TANK_CAPACITY la capacité maximale du réservoir en milli litre
- Spécifiez 8 valeurs utilisées pour calibrer votre capteur dans INIT_FLOW_PARAM
Ces paramètres sont utilisés car le nombre d'impulsions générées par le capteur (pour 1 millilitre de liquide) varie en fonction du débit.
Pour 4 valeurs de débit (les 4 premiers paramètres), oXs vous permet, dans les 4 derniers paramètres, de définir la correction (en%) à appliquer.
Le processus d'étalonnage du capteur peut être le suivant:
Définir les 4 derniers paramètres sur 0 (donc 0% de correction)
Faire fonctionner votre moteur à 4 vitesses différentes (d'une vitesse très basse, à une moyenne et à très élevée) pendant quelques minutes.
Pour chaque régime de fonctionnement,
- Notez le carburant restant (en ml) rapporté par oXs au démarrage (par exemple 1200) et à la fin du test (par exemple 1090)
- Mesurer le carburant restant réellement (en ml) dans le réservoir au démarrage (par exemple 1500) et à la fin du test (par exemple 1380)
- Notez le temps de fonctionnement (en min) entre le début et la fin du test (par exemple, 2 minutes).
- Comparez le carburant en ml consommé (différence entre le carburant restant au début et à la fin) rapporté par oXs à la consommation réelle (par exemple, rapporté = 1200 - 1090 = 110 ml; réel = 1500 - 1380 = 120 ml)
- Calculer le facteur de correction à appliquer (par exemple (120 - 110) / 110 = 9%); La correction des notes pourrait être négative)

- Calculer le débit où cette correction s'applique (= ml consommé rapporté par oXs / temps en conflit = 110 ml / 2 min = 55 ml / min)
Remplissez les 4 premiers paramètres avec les débits calculés (par exemple, 55) et les 4 derniers paramètres avec le pourcentage de correction (par exemple, 9).
Veillez à ce que les 4 premiers paramètres soient en ordre croissant (donc de faible vitesse à grande vitesse).
Note : Lorsque oXs calcule la consommation, il appliquera une interpolation linéaire pour la gamme de valeurs associée.

Si vous déplacez un débitmètre oXs d'un avion à un autre, vous devrez probablement modifier les valeurs définies dans TANK_CAPACITY et / ou INIT_FLOW_PARAM.
Cela nécessite de télécharger un nouveau firmware dans votre oXs, sauf si vous utilisez le protocole JETI ou le protocole SPORT avec openTx 2.2.x (ou suivant).
Pour le protocole JETI, vous pouvez simplement entrer dans la boîte de dialogue JETIBOX, appuyer sur la touche DOWN pour passer à l'élément à modifier et appuyer sur "<" or ">" pour augmenter / diminuer la valeur.
N'oubliez pas d'activer l'option SAVE_TO_EEPROM dans la section 8 afin d'enregistrer les paramètres et de les réutiliser à la prochaine mise sous tension.
Pour le protocole SPORT, vous devez exécuter certains scripts LUA. Dans le dossier oXs (dans le sous-dossier "scripts lua"), vous pouvez trouver 3 scripts LUA à charger dans le dossier SCRIPTS/FUNCTIONS de la carte SD:
- Un (rstful.lua) pour réinitialiser le carburant consommé à 0 (à utiliser lorsque le réservoir est rempli)
- Un (tank.lua) pour définir la capacité maximale du réservoir (avant l'exécution du script, la capacité du réservoir doit être entrée dans GVAR9 pour la phase 9
La valeur dans GVAR9 est le nombre de 50 ml dans le réservoir. Cela signifie, par exemple, que pour un réservoir de 1000 ml, GVAR9 doit être réglé sur 20 (= 1000/50)
- Un (calful.lua) pour définir les 8 (4 X 2) paramètres d'étalonnage (avant d'exécuter le script, les paramètres doivent être entrés dans GVAR9 pour les phases 0 à 8)
Les premières 4 valeurs sont le débit en ml / min (min = 30, max = 800); les valeurs doivent être en ordre croissant
Les 4 dernières valeurs sont la correction en pourcentage à appliquer (min = -100, max = 100) pour chaque flux
Note: Les paramètres enregistrés dans l'eeprom ou chargés par un script LUA ont priorité sur les paramètres définis dans la configuration.

Veuillez noter que la consommation de carburant peut être sauvegardée toutes les 30 secondes dans une mémoire non volatile.
Pour activer cette option, vous devez dire YES dans la ligne #define SAVE_TO_EEPROM de la section 8
Si cette option est activée, à la mise sous tension oXs continue à décompter à partir du carburant qui restait lorsque l'alimentation a été coupée.
Sinon, oXs réinitialise la consommation de carburant et assume un réservoir de 100%.

Une réinitialisation de la consommation de carburant peut être demandée sur le TX. Ceci est vraiment nécessaire lorsque SAVE_TO_EEPROM est activé
Cela peut se faire de plusieurs façons en fonction également du protocole utilisé:
Pour tous les protocoles, il peut être demandé à l'aide d'un signal PPM
Cela nécessite d'activer l'option oXs PPM (voir la section 3) et de connecter un canal Rx à oXs.
La réinitialisation se produira lorsque la valeur absolue du signal PPM dépasse la valeur spécifiée dans FLOW_SENSOR_RESET_AT_PPM (section 11)

En outre:
- Pour le protocole JETI, il peut être effectué avec le JETIBOX en pressant simultanément les touches "<" et ">" lorsque JETIBOX affiche le pourcentage de carburant restant
- Pour le protocole FRSKY SPORT, depuis OpenTX version 2.2.x, il est possible d'activer un script de fonction LUA qui enverra une commande de réinitialisation à oXs

Pour les protocoles JETI, oXs transmet automatiquement le débit actuel, le carburant restant en ml et en%
Pour d'autres protocoles, vous devez demander à oXs de transmettre les données; Donc vous avez:
- A dé-commenter la ligne #define FILL_TEST_1_2_3_WITH_FLOW_SENSOR_CONSUMPTION (dans la section 2.5)
- A spécifier dans lequel des champs de télémétrie, TEST_1 (débit actuel en ml / min), TEST_2 (carburant restant en ml) et TEST_3 (carburant restant en%) sont envoyés (voir section 2.1 / 2.4)
Notez que si vous transmettez TEST_1 ... TEST_3 dans AccX ... AccZ dans le protocole FRSKY, les valeurs sont divisées par 100 par OpenTx. Vous pouvez récupérer les valeurs d'origine si vous configurez une échelle = 255 sur le côté Tx dans les champs de télémétrie

****
#define A_FLOW_SENSOR_IS_CONNECTED NO // Sélectionnez parmi YES , NO
#define PULSES_PER_ML 10.0 // Nombre d'impulsions par milli litre (dépend du capteur); Peut avoir des décimales
#define TANK_CAPACITY 1000 // Capacité du réservoir en ml
#define INIT_FLOW_PARAM 30 , 100 , 500 , 700 , 20 , 10, -5, 15 // Définissez 4 niveaux de débit (en ml / min) (par exemple 30, 100, 500, 700) et 4 paramètres de correction (en%; par exemple 20, 10, -5, 15); Les débits doivent être mis en ordre ascendant.
#define FLOW_SENSOR_RESET_AT_PPM 95 // Lorsque la valeur absolue de ppm est supérieure à celle-ci, la consommation est réinitialisé.

12 - Localisateur

Les oX ont envoyé des données au Tx en utilisant le protocole de télémétrie Rx sur la bande 2.4G tel que défini par FrSky, Hott, Jeti ou Multiplex. Lorsque le modèle est au sol, la portée sur 2.4G est assez limitée, donc, si un modèle est perdu à plus de quelques centaines de mètres, le Tx ne recevra plus de données de télémétrie
oXs permet d'utiliser son propre émetteur afin d'avoir une portée étendue et d'avoir une chance de retrouver un modèle perdu
C'est possible car il utilise une fréquence plus basse, une vitesse de transmission inférieure et un protocole spécial (LORA). L'émetteur est un module LORA SX1276 / RFM95 facilement disponible
Le principe est le suivant
Vous devez construire 2 appareils, un appareil "openXsensor" avec un Arduino, les capteurs que vous voulez (idéalement un GPS et par exemple vario, tensions, courant, ...) et un module SX1276 / RFM95
Un dispositif "récepteur de localisation" avec un Arduino, un module SX1276 / RFM95 et un afficheur
L'écran doit être un 0,96 pouce OLED 128X64 I2C SSD1306. Il est petit et est disponible pour environ 2 €.
Normalement, le récepteur de localisation n'est pas utilisé. L'openXsensor est installé dans le modèle et transmet les données du capteur via la liaison RC Rx / Tx normale.
Le module SX1276 dans oXs est en mode écoute: il envoie des données uniquement lorsqu'il reçoit une requête du "récepteur de localisation"
Lorsqu'un modèle est perdu, le récepteur de localisation "est mis sous tension. Il commence à envoyer des requêtes à openXsensor.

Lorsque le module SX1276 / RFM95 dans openXsensor reçoit une requête, il répond par un petit message contenant les coordonnées GPS et quelques données sur la qualité du signal de requête
L'affichage sur le récepteur de localisation montre ces données ainsi que la qualité du signal reçu et le temps écoulé depuis le dernier message reçu
Remarque: la plage de communication entre deux modules SX1276 est normalement plusieurs fois plus grande que la liaison RC 2.4G
Malgré tout, quand openXsensor et le récepteur de localisation sont tous les deux au sol, il se peut qu'ils soient trop éloignés pour communiquer
Il existe 2 façons d'étendre la portée

- utiliser une antenne directionnelle sur le récepteur de localisation
    - L'avantage de cette solution est que, si vous obtenez une communication, vous pouvez utiliser le système comme un goniomètre (en regardant la qualité du signal) pour connaître la direction du modèle perdu. Cela fonctionne même si vous n'avez pas de GPS connecté à openXsensor
    - L'inconvénient est qu'une antenne directionnelle n'est pas aussi petite qu'un simple fil
- placez le récepteur de localisation (qui est encore un petit appareil) sur un autre modèle et survolez la zone perdue attendue.
    - Dans ce cas, la portée peut être supérieure à 10 km et les chances sont très élevées qu'une communication puisse être établie entre les 2 modules

Même si la communication est interrompue lorsque le modèle utilisé pour la recherche revient au sol, vous connaîtrez l'emplacement du modèle perdu car l'écran affichera toujours les dernières coordonnées GPS reçues
Un appareil openXsensor avec un SX1276 / RFM95 ne perturbe pas la liaison 2.4G et ne consomme que quelques milliAmp car il reste normalement en mode d'écoute (et lors de l'envoi, il ne s'agit que de quelques pourcentage du temps).
Ainsi, afin d'augmenter la fiabilité du système, il est possible d'alimenter openXsensor avec une batterie lipo 1S séparée de par ex. 200 mAh

Câblage: Le module SX1276 / RFM95 doit être connecté à l'Arduino de la manière suivante (ceci est valable aussi bien pour le dispositif openXsensor que pour le dispositif récepteur de localisation):
Broche numérique Arduino 10 <=> NSS du module
Broche numérique Arduino 11 <=> MOSI du module
Broche numérique Arduino 12 <=> MISO du module
Broche numérique Arduino 13 <=> SCK du module
Arduino GRND <=> GRND du module
Externe (ou Arduino?) 3.3V <=> 3.3V du module
Attention : le module doit être alimenté en 3,3 volts et non en 5 volts. L'Arduino doit également être une version 3,3 V (donc, avec une horloge de 8 MHz)
A vérifier: il faut peut-être utiliser un régulateur de tension supplémentaire (coût inférieur à 1 €) pour obtenir le 3,3 V, car il n'est pas sûr que le régulateur de tension arduino puisse fournir suffisamment de courant lorsque le module transmet (pour juste un petit temps)
Du côté récepteur du localisateur, il n'est pas nécessaire de connecter le capteur mais de souder l'écran. Il utilise 4 fils:
Broche Arduino A4 <=> Affichage SDA
Broche Arduino A5 <=> Affichage SCL
Écran Arduino GRND <=> GRND
Affichage Arduino 3.3V <=> Vcc

Le programme à télécharger dans le récepteur de localisation Arduino (nano ou pro mini) est disponible sur github dans le répertoire locator_receiver.
Pour utiliser un localisateur, vous devez changer une ligne dans oXs_config_basic.h

****

#define A_LOCATOR_IS_CONNECTED NO // choisir entre OUI, NON
* Dans le fichier oXs_config_advanced.h, vous pouvez sélectionner la fréquence à utiliser entre les 2 modules SX1276 (par exemple pour éviter les perturbations lorsque plusieurs appareils sont utilisés simultanément)
Remarque: si vous modifiez la fréquence, veillez à l'utiliser sur les appareils récepteurs openXsensor et locator

13 - Qualité du lien Rf

oXs peut vérifier la qualité de la liaison Rf en surveillant le signal PWM sur un canal normalement inutilisé
Le principe est le suivant :
Le Tx génère sur un canal un signal qui change continuellement (comme lors de l'utilisation d'un servo-testeur); le plus simple est probablement d'utiliser une forme d'onde triangulaire
Le Rx reçoit ce signal et génère une impulsion à intervalle régulier (par exemple toutes les 18 ms pour un récepteur Frsky X8R)
La largeur de l'impulsion varie avec le signal (normalement entre 1000 usec et 2000 usec
Si le Rx ne reçoit pas le signal du Tx, le Rx générera toujours une impulsion mais la largeur sera la même que le dernier reçu (ou après un certain temps une valeur prédéfinie définie comme sécurité intégrée)
oXs peut mesurer la largeur de chaque impulsion et la comparer avec la précédente. Obtenir 2 impulsions identiques indique une perte de signal Rf.
oXs calcule le pourcentage de perte de signal sur un certain temps (par exemple sur la base de 50 signaux) et le nombre de pertes de signal consécutives
Ce principe pourrait normalement être utilisé avec tous les protocoles Rf (FRSKY, JETI, HOTT, MULTIPLEX) si le TX peut être programmé afin de générer un signal variable
Encore actuellement les 2 mesures sont stockées en interne dans TEST_1 et TEST_2 et seul le protocole FRSKY permet de sélectionner des champs de télémétrie pour les transmettre
Pour utiliser cette fonctionnalité, vous disposez:
- Laissez le Tx génère un par ex. signal d'onde triangulaire sur un canal libre.
- Avec openTx, cela nécessite 2 configurations:
- Créer un commutateur logique (par ex. L01) couplé au canal libre (par ex. canal 8): régler par ex. L01 avec "a <x" "Channel 8" "0"
Créer un mixage pour un canal (par exemple 8): Source = le commutateur logique L01, Delay up and down = 1.0, Slow up and down = 2.0
Dans le fichier oXs_config_basic.h: dans la section 2.1 (pour le protocole FRSKY), définir dans quels champs les 2 mesures TEST_1 et TEST_2 doivent être transmises (par exemple dans TEMP1 et TEMP2)

****

définir MEASURE_RF_LINK_QUALITY sur OUI

Dans le fichier oXs_config_advanced
- Dans la section 2.5, décommentez #define FILL_TEST_1_2_WITH_LQ afin de laisser TEST_1 et TEST_2 être remplis
- Dans la section 3 (réglage PP), définissez PIN_PPM avec la broche numérique arduino (devrait être 2 ou 3) qui sera connectée au canal RX
- Dans la section 13, définissez les paramètres suivants

PULSE_INTERVAL_MIN et PULSE_INTERVAL_MAX sont utilisés pour rejeter une mesure d'impulsion PWM si le paramètre. Prenez soin de laisser une certaine tolérance; par exemple. 17000 et 19000 semblent corrects pour un FRSKY Rx générant une impulsion toutes les 18000 micro sec. Le délai entre 2 impulsions n'est pas dans l'intervalle attendu.
LQ_COUNT_MAX définit le nombre d'impulsions PWM utilisées pour calculer les 2 paramètres de qualité Rf; 50 signifie que vous obtenez la mesure environ une fois par seconde (par exemple 18 ms * 50)
WIDTH_ERROR_MAX définit la différence de largeur maximale entre 2 impulsions consécutives
Si la différence de largeur est inférieure ou égale à ce paramètre, oXs considère que le signal n'a pas suffisamment de variation et donc qu'une trame Rf n'a pas été reçue.
Normalement, ce paramètre doit être réglé sur 1 ou 2.
- Pour de meilleurs résultats, Rx doit être configuré avec un fail safe en "Hold".

Si vous attribuez une valeur PWM prédéfinie pour la sauvegarde en échec dans votre Rx, les oX ne compteront pas comme une erreur la première trame après que le Rx entre en mode de sécurité intégrée car la valeur PWM sera très probablement différente de la précédente.
Donc, dans ce cas, la mesure de la qualité des liens rapportée par oXs sera trop optimiste.
Pourtant, normalement un Rx n'entre pas en mode de sécurité intégrée si seulement quelques rf se produisent. Ainsi, la mesure oXs peut toujours donner une indication acceptable de la qualité rf (du moins si le Rx n'entre pas trop vite en mode de sauvegarde en cas d'échec)
Remarque: sur certains Tx (par exemple ceux avec openTx), vous pouvez demander au Tx de calculer et d'afficher le pourcentage de qualité le plus bas et la valeur la plus élevée d'erreurs LQ consécutives afin de surveiller toute la session.

****
#define PULSE_INTERVAL_MIN 17000 // délai minimum (micro seconde) entre 2 impulsions PWM générées par le Rx
#define PULSE_INTERVAL_MAX 19000 // délai maximum (micro seconde) entre 2 impulsions PWM générées par le Rx
#define LQ_COUNT_MAX 50 // nombre d'impulsions PWM utilisées pour calculer les 2 paramètres de qualité Rf; 50 signifie que vous obtenez la mesure une fois par seconde (par exemple 18 ms * 50)
define WIDTH_ERROR_MAX 1 // L'impulsion PWM est considérée comme incorrecte si la largeur de 2 impulsions consécutives diffère de ce paramètre ou moins. Normalement, ce paramètre doit être réglé sur 1 ou 2

20 - Séquenceur (ON/OFF) pour certaines sorties digitales

oXs vous permet de contrôler (HIGH / LOW) jusqu'à 6 sorties Arduino dans différentes séquences.
Chaque séquence est composée d'une ou plusieurs étapes; Chaque étape définit pour combien de temps (= une durée) les sorties sont HAUT et puis BAS.
oXs détermine normalement la séquence à lire en fonction du signal reçu sur un canal PPM (voir la section 3 pour configurer un signal PPM).
Il y a encore 2 exceptions:
À la mise sous tension ou lorsque aucune chaîne PPM n'est configurée / reçue, oXs générera par défaut la séquence définie dans la ligne #define SEQUENCE_m100 (voir ci-dessous)
Lorsqu'une alarme de basse tension est configurée (voir ci-dessous) et si la tension devient faible, oXs générera la séquence définie dans la ligne #define SEQUENCE_LOW (voir ci-dessous) tant que la tension reste faible

Lorsqu'une séquence a été exécutée, oXs peut soit la répéter soit attendre l'ordre d'exécuter une nouvelle séquence. La différence est faite dans la configuration de la séquence.
Chaque fois qu'un nouveau signal PPM (= un signal différent) est reçu, oXs démarre immédiatement la séquence correspondante (même si la séquence actuelle n'est pas complètement exécutée)
- Pour utiliser la fonctionnalité du séquenceur, vous devez d'abord définir quelles broches Arduino doivent être contrôlées par le séquenceur.
Les broches Arduino qui peuvent être contrôlées sont les broches 13, 12, 11, 10, 9 et 8.
Cette configuration est obtenue par une ligne comme: #define SEQUENCE_OUTPUTS 0b100100
Chaque bit (1 ou 0 après le "b") représente une sortie; Le bit le moins important correspond à la broche 8, le bit à gauche la broche 9, etc., jusqu'à la broche 13
Mettre 1 lorsque la broche doit être contrôlée par le séquenceur, 0 sinon; Dans cet exemple, cela signifie que seules les broches 13 et 10 seraient contrôlées.
Remarque: si la ligne #define SEQUENCE_OUTPUTS xxxx est omise ou mise comme commentaire, le séquenceur n'est pas actif du tout.
Veillez à ne pas utiliser la même broche pour le séquenceur et pour une autre fonctionnalité oXs (par exemple comme broche Tx, pour bouton poussoir, pour PPM, pour RPM, ...)
Si une broche est configurée avec 0 (= non contrôlée par oXs), elle ne sera jamais forcée à HIGH ou LOW par le séquenceur même si un 1 ou 0 est configuré dans une séquence.
Lorsque le séquenceur est activé (SEQUENCE_OUTPUTS est défini) Le signal PPM est automatiquement utilisé SEULEMENT pour contrôler la séquence (donc PPM ne peut plus contrôler la sensibilité du vario ...)
Le passage de courant à travers les broches numériques d'Arduino ne doit pas dépasser 40mA par broche (et 200 mA pour l'ensemble des broches).
Dans le cas où vous souhaitez avoir un courant plus élevé (ce qui est le cas pour la plupart des LED à haute puissance et des bandes LED), vous devez ajouter un transistor. Le schéma de connexion peut facilement être trouvé dans Google.
- Ensuite, vous pouvez (optionnel) spécifier les unités utilisées pour définir les durées
Par défaut, les durées sont exprimées en 10 millièmes de seconde. Un paramètre vous permet d'augmenter l'unité en multiple de 10 millièmes de seconde;
Exemple: avec une ligne comme #define SEQUENCE_UNIT 50, les durées deviennent des multiples de 500 millièmes de seconde (= 50 10)
Remarque: ce paramètre doit être un nombre entier entre 1 et 1000. Donc il est impossible d'obtenir une durée inférieure à 10 ms.
Si cette ligne est omise (ou comme commentaire), la valeur par défaut (1 = 10 ms) sera appliquée.
- Ensuite, vous devez définir les séquences utilisées pour chaque valeur du canal PPM
Vous pouvez définir jusqu'à 9 séquences différentes.
Une séquence est définie par une ligne comme: #define SEQUENCE_m50 200, 0b100000, 300, 0b000000, 100, 0b100100
Chaque séquence est identifiée par la valeur du signal en ppm qui l'active; Le suffixe m100 (m = moins) signifie qu'il doit être activé lorsque la valeur de ppm est d'environ -100, m75 est pour ppm = -75, 75 est pour ppm = +75, etc.
Le suffixe de séquence est en multiple de 25; Les seuls suffixes de séquence valides sont : m100, m75, m50, m25, 0, 25, 50, 75 et 100
Chaque séquence se compose de plusieurs étapes (de 1 à 126 étapes ou même plus) (séparées par ",")
Chaque étape est composée de 2 paramètres (également séparés par ","): une durée et une combinaison (LOW / HIGH) des sorties
- Une durée peut être n'importe quelle valeur entre 0 et 255.

La valeur fixe la durée minimale qu'une combinaison de sorties doit être appliquée. La durée (en ms) est égale à la valeur indiquée SEQUENCE_UNIT 10
Ainsi, une valeur = 2 signifie une durée de 1 seconde si SEQUENCE_UNIT = 50.
Une valeur égale à 0 a une signification particulière. Lorsque oXs atteint une durée = 0, il applique la combinaison correspondante de sorties et la conserve pendant un temps non limité.
Cela permet de forcer les sorties à rester avec une combinaison spécifique après avoir joué la séquence.
Si la durée = 0 est utilisée, elle devrait être dans la dernière étape de la séquence (parce qu'oXs n'appliquera jamais les étapes suivantes).
Si la durée est 0 dans la première étape, oXs appliquera directement la combinaison spécifique des sorties et la gardera.
Si la durée = 0 n'est pas utilisée dans une séquence, oXs répétera automatiquement toute la séquence après avoir atteint la dernière étape.
Note: Si vous avez besoin d'une durée supérieure à la durée maximale (= 255 SEQUENCE_UNIT 10 ms), vous pouvez définir plusieurs étapes successives avec la même combinaison de sorties.
- Une combinaison (LOW / HIGH) des sorties définit les broches qui doivent être forcées sur LOW et celles sur HIGH
Une combinaison peut être définie en format binaire pour définir six 1 (HIGH) et / ou 0 (LOW) juste après "0b" (par exemple 0b100100)
Le bit le moins important correspond à la broche 8, le bit à gauche la broche 9, etc., jusqu'à la broche 13.
Donc, si SEQUENCE_OUTPUTS = 0b110111, alors 0b100100 signifie que:
- les broches 13 et 10 doivent être HIGH,
- Les broches 9 et 8 doivent être LOW
- Les autres (broches 12 et 11) ne sont pas contrôlées par une séquence en raison de la valeur attribuée à SEQUENCE_OUTPUTS = 0b100111
Ainsi, #define SEQUENCE_m50 2, 0b100000, 3, 0b000000, 1, 0b100100 signifie (en supposant que SEQUENCE_OUTPUTS = 0b100100 et SEQUENCE_UNIT = 50,):
- Mettre la broche 13 sur HIGH et la broche 10 à 0 (= 0b100000) lorsque le signal PPM devient -50
- Attendre au moins 2 50 10 = 1000 ms = 1 sec avant de changer les sorties
- Après 1 sec, mettre la broche 13 (et la broche 10) sur LOW (= 0b000000) pendant une durée de 1,5 sec (3 50 10)
- Après 1,5 sec, mettre les broches 13 et 10 sur HIGH pendant une durée de 0,5 sec (1 50 10)
- Après 0,5 seconde, répéter la première étape (broche 13 HIGH pendant 1 sec)
- Continuez avec les prochaines étapes
Note : lorsqu'une séquence n'est pas définie, oXs la gère comme s'il elle était définie avec 0, 0b000000 (donc pas de répétition, toutes les sorties LOW)
- Enfin, vous pouvez (mais ce n'est pas obligatoire) configurer la (les) condition (s) pour détecter une tension trop basse. Lorsqu'une tension devient trop faible, oXs démarre automatiquement la SEQUENCE_LOW (et néglige les informations venant par le canal PPM)
Une condition basse tension peut être configurée en fonction de 1 ou 2 tension (s):
- La tension sur la broche Arduino définie par le 6ème paramètre PIN_VOLTAGE; Cette configuration est obtenue par une ligne comme: #define SEQUENCE_MIN_VOLT_6 6000 où 6000 est la tension "basse" en mVolt.
Note : Si vous utilisez cette option, n'oubliez pas d'assigner un numéro de broche au 6ème paramètre dans #define PIN_VOLTAGE et de remplir (si demandé) les autres paramètres de tension pour ce 6ème élément.
La broche définie dans le 6ème paramètre de PIN_VOLTAGE peut être identique à un autre paramètre dans PIN_VOLTAGE; Cela peut être utile si vous souhaitez également configurer des paramètres de basse tension.

- La plus faible tension de toutes les cellules lipo; Cette configuration est obtenue par une ligne comme: #define SEQUENCE_MIN_CELL 3000 où 3000 est la tension "basse" en mVolt.
Note: Si vous utilisez cette option, n'oubliez pas de définir les autres paramètres de tension PIN_VOLTAGE, etc ... et NUMBEROFCELLS
Note : Lorsqu'aucun paramètre de basse tension n'est défini, oXs ne démarre pas automatiquement SEQUENCE_LOW.
Lorsque les deux paramètres de tension sont définis, oXs démarrera automatiquement SEQUENCE_LOW dès que l'une des 2 tensions devient faible.
Si vous souhaitez que oXs notifie une détection de basse tension, n'oubliez pas de définir SEQUENCE_LOW (sinon, oXs définira toutes les broches de sortie sur LOW)
Si vous avez la télémétrie, vous pouvez également effectuer une configuration sur le Tx pour détecter une tension trop faible, puis envoyer une valeur spécifique sur le canal ppm.
Dans ce cas, vous ne devez pas définir la configuration dans oXs et le même appareil peut être utilisé sur plusieurs modèles.

xx - Reservé au développeur

DEBUG doit être activé ici lorsque vous souhaitez déboguer une ou plusieurs fonctions dans d'autres fichiers.
L'activation de DEBUG permettra d'utiliser l'Arduino Serial Monitor à 115200 baud (ou 38600 lorsque le GPS est activé) pour voir les données d'initialisation et certaines valeurs de capteur en direct
Vous pouvez ensuite sélectionner les parties que vous souhaitez déboguer en dé-commentant les paramètres DEBUG spécifiques que vous souhaitez dans chaque fichier cpp.
Note : oXs permet aussi de transmettre 3 champs appelés TEST_1, TEST_2, TEST_3. Vous pouvez remplir ces champs avec ce que vous voulez, si vous voulez transmettre des données supplémentaires au Tx.
Il suffit de remplir test1.value (ou 2, 3) avec un int32_t et test1.available (ou 2, 3) avec true et ajouter TEST_1, (2 ,3) dans la section qui précise les données à envoyer.

****

```
//#define DEBUG
```