F4ToSerial

Envoyer les données de Falcon BMS vers votre cockpit

Documentation (Français)

Documentation écrite par : Myoda

Version du document : 20180727001

Table des matières

1 Le programme F4ToSerial

1.1 - Introduction

Le programme F4ToSerial a été développé avec pour objectif d'inclure dans une seule et même application le contrôle de tous les éléments « actifs » d'un cockpit de simulation.

Par éléments actifs on entend ceux qui reçoivent les informations du simulateur et qui réagissent en fonction : jauges, voyants, afficheurs 7 segments et écrans.

F4ToSerial a été exclusivement développé pour le simulateur Falcon 4 BMS à partir de la version 4.33 ou supérieur.

Les informations proviennent de la mémoire partagée du simulateur, et sont acheminées via le port série vers des cartes contrôleurs capables de piloter ces éléments dits « actifs ».

F4ToSerial est un programme « tout en un » qui vise à apporter une solution alternative à la multiplicité des programmes existant actuellement pour Falcon BMS.

Le programme F4ToSerial possède néanmoins deux contraintes majeures.

- Il ne fonctionne que dans un seul sens (Falcon BMS → Cockpit)
- Il a été développé pour la version « Block 52 » du F16.

L'envoie des données du cockpit vers le simulateur (donc dans l'autre sens) ne représente pas de difficulté et ne sera pas abordé ici, dans la mesure où il s'agit de simuler des touches au clavier dans 90 % des cas.

Le programme F4ToSerial a été développé pour fonctionner avec des cartes électroniques du type Arduino et le code de ces cartes est fourni.

Enfin, notez que ce programme a été développé par moi-même dans le cadre de la création de mon cockpit maison et a été mis en ligne pour la communauté. Il ne saurait être utilisé à des usages professionnels et reste gratuit.

1.1 - Méthodologie

Lors de mes débuts dans la création de mon cockpit de simulateur de F16, j'ai rapidement compris que l'argent investit était élément déterminant dans la qualité du cockpit fini. Certes le temps et l'expertise compte mais un cockpit de haut niveau et fini coûte des dizaines de milliers d'euros… et il ne faut pas se mentir !

N'ayant pas un budget extensible, j'ai dû faire en sorte de consommer le moins de composants possibles pour la création de mon cockpit.

Il existe globalement deux approches pour le cockpit Builder. Dans l'une, le budget n'est pas pris en compte, et dans l'autre on fait attention à chaque euro investit.

Ce principe s'applique aux « lightBits » et aux afficheurs 7 segments par exemple, mais aussi plus généralement à tous les modules du cockpit, Boutons, Switchs, écrans etc. mais aussi l'ACESII, la throttle, etc..

En revanche, plus on simplifie en composant, plus les branchements et la simplification rendent les montages complexes et difficile à développer.

N'étant pas un bricoleur de génie, ni un électronicien né, mon approche se situe à mi-chemin entre les deux méthodes.

C'est pourquoi dans ce guide, j'ai choisi d'utiliser telle ou telle méthode qui parfois simplifient et parfois sont consommatrice à la dépense.

Bien évidement, tout peut être amélioré et je ne prétends pas avoir la solution miracle.

1.2 - Quelle type de carte électronique utiliser avec F4ToSerial ?

Il existe plusieurs types de cartes très utiles pour le constructeur de cockpit de simulateur. Les cartes Photons, Arduino et Pokeys etc. La dernière ne sera pas abordée dans ce document car son utilisation avec le programme F4ToSerial n'est pas encore implémentée.

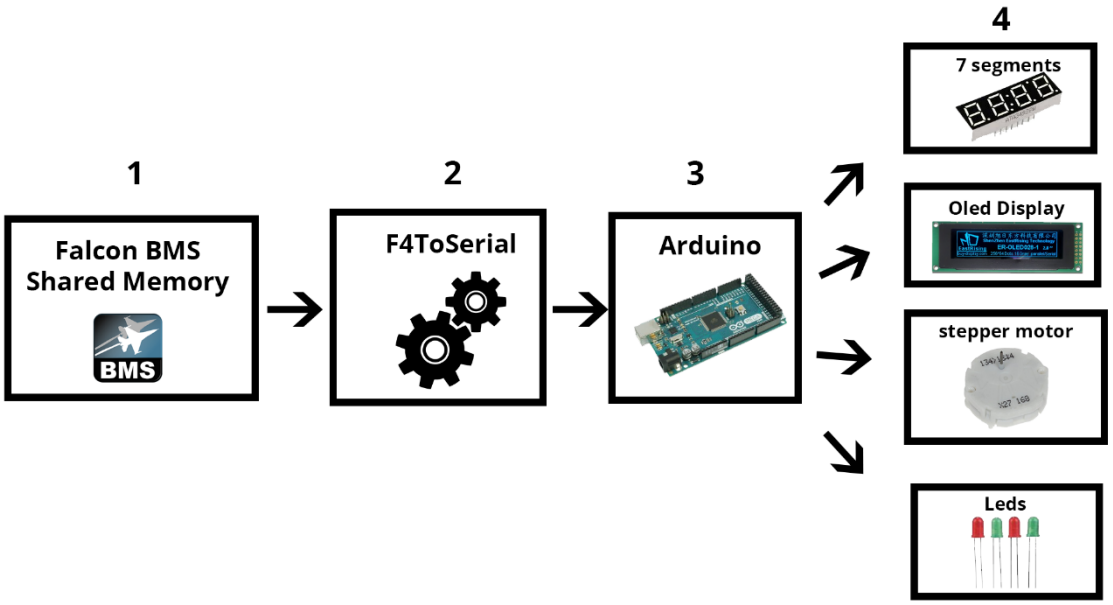
Chacune présente des avantages et des inconvénients et ce document n'a pas pour vocation à faire un comparatif ou vous orienter lors du choix de votre carte.

Notez cependant que le programme F4ToSerial a été développé uniquement pour la carte Arduino Mega.

La carte Arduino UNO est compatible mais en raison de sa faible quantité de mémoire, celle-ci-ne peux pas utiliser une des librairies essentielles du programme C++.

Les autres cartes ne sont pas encore prises en charge et ne ce n'est pas prévu pour le moment.

1.3 - Comment envoyer des données vers un cockpit ?



Avant d'avoir les jauges qui fonctionnent dans un cockpit de simulateur ou encore l'écran DED du F16 qui affiche des informations, un certain nombre d'étapes sont nécessaires. Le schéma précédent décrit ces étapes qui sont détaillées juste après.

Etape 1: Shared Memory

La mémoire partagée est une zone que les développeurs de Falcon BMS ont gentiment implémentée et documentée pour permettre aux constructeurs de cockpit de lire les informations courantes de l'avion et du vol : (altitude, cap, position des manettes etc..).

La première étape consiste donc à accéder au registre des informations de la mémoire partagée. F4ToSerial utilise pour cela la DLL Windows F4SharedMem.dll qui est installée dans le répertoire d'installation du programme F4ToSerial.

Etape 2 : F4ToSerial

L'étape 2 permet de récupérer les données de la mémoire partagée, pour les transformer et les envoyer au cockpit. Dans cette étape, le programme F4ToSerial effectue une boucle permanente sur les adresses mémoire, puis récupère et prépare les informations avant leurs envois sur le port série.

Les données sont transformées par exemple du binaire en une chaine de caractère lisible par la carte Arduino à l'épate suivante.

Exemples de trames de données au format Json.

Configuration d'un moteur pas à pas :

{"SETUP_STEPPER":{"RPM":{"NAME":"RPM","PINS":[46,44,42,40],"PINSCOUNT":4,"STEPS":600,"MAXSPEED":1000,"ACCELERATION":1000}}}

Mise à jours d'une matrice de LEDs :

{"UPDATE_MATRIX":{"A":{"MATRICE":[[1,1,0,0],[1,0,0,1],[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0]]}}}

Affichage des informations de l'écran DED :

{"SET_DISPLAY":{"DED":[" UHF 292.30   STPT $ 8","","," VHF   1   10:56:20","","," M1 3 C 6400 MAN T 75X"]}}

1.3.3 – Etape 3 : Récupération et transformation du signal

Lors de cette étape, la carte Arduino reçois les informations sur le port série puis les transformes en signaux électriques compatible avec éléments actifs (jauges, voyants, afficheurs 7 segments et écrans).

1.3.4 - Etape 4 : activation des éléments du cockpit.

Dans cette dernière étape les éléments actifs évoluent en fonction des signaux reçus (déplacement des jauges, allumage des leds etc.).

Notez que toutes ces étapes fonctionnent toujours dans le sens Falcon BMS vers le cockpit. La mémoire partagée n'étant accessible qu'en lecture seule, toute interaction avec le simulateur se fera à travers des frappes claviers.

2 Télécharger et installer le programme F4ToSerial

2.1 - Téléchargement du programme F4ToSerial

Vous pouvez télécharger le programme F4ToSerial depuis le site internet : http://f4toserial.mini-cube.fr/.

La dernière version de l'application est disponible ici : http://f4toserial.mini-cube.fr/download-latest-version/

Les versions antérieures du programme sont pour l'instant accessible également.

2.2 - Téléchargement du code ++ pour Arduino

2.2.1 – Méthode 1 (Facile) :

1) Télécharger ici un programme qui vous permet de transférer un fichier Hex vers une carte Arduino http://xloader.russemotto.com/
2) Transférer le code Hexa directement sur la carte Arduino (Mega(ATMEGA2560)). http://f4toserial.mini-cube.fr/download-latest-version/

2.2.2 - Méthode 2 (complète) :

Pour faire fonctionner vos modules ou éléments actifs pleinement, vous aurez aussi besoin de télécharger le code C++ compatible avec les cartes Arduino.

Le lien de téléchargement du code se trouve ici :

https://bitbucket.org/falconbms/

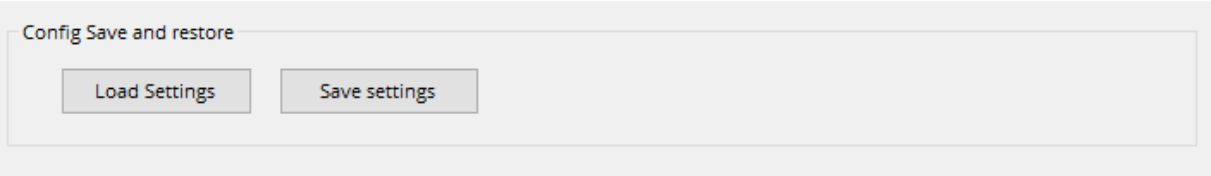
Les fichiers sources C++ pour la carte Arduino sont disponible ici : https://bitbucket.org/falconbms/arduino-commons.git

> *Je vous invite à cloner les fichiers du réporsitory avec un outil du type SourceTree afin de garder à jour les fichiers si de nouvelles versions du code C++ sont mises en ligne.*

3 Manuel utilisateur F4ToSerial

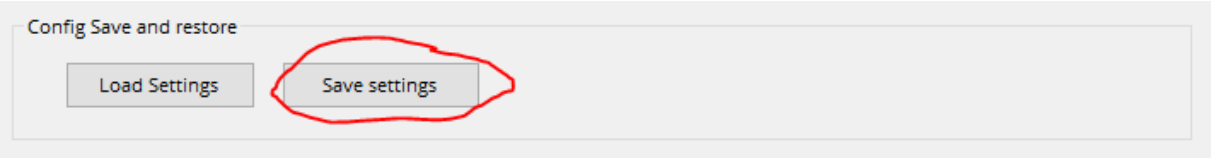
3.1 - Sauvegarde et restauration de la configuration

F4ToSerial permet de sauvegarder la configuration. Cette dernière est fastidieuse à mettre en place la première fois, c'est la raison pour laquelle il est important de bien enregistrer cette dernière à la fin du paramétrage des équipes actifs.



3.1.1 - Sauvegarde de la configuration

La sauvegarde de la configuration se fait en cliquant sur le bouton « Save settings ».



Vous devez ensuite indiquer un emplacement et un nom pour le fichier de configuration. L'extension du fichier de sauvegarde est « .xml ». Il n'est pas nécessaire de le préciser.

Les fichiers de configuration peuvent être modifiées facilement avec un éditeur HTML/XML.

Ci-après un exemple de fichier de configuration :

<?xml version="1.0" encoding="utf-8"?>

<serial_settings>{"7_chaff":"","7_flares":"","7_fuel":"","7_uhf_chan":"","7_uhf_freq":"","aft":"","bcd_decoder_0":"COM1","bcd_decoder_1":"COM1","cabinAlt":"","ded":"","epu":"","ffi":"","ftit":"","fwd":"","hyd_press_a":"","hyd_press_b":"","Matrix_A":"","Matrix_B":"","Matrix_C":"","Matrix_D":"","Matrix_E":"","noz":"","oil":"","oxygen":"","pfl":"","pitch":"","roll":"","rpm":"","yaw":""}</serial_settings>
<steppers_speed_accell>{"aft":[1000,1000],"cabinAlt":[1000,1000],"epu":[1000,1000],"ftit":[1000,1000],"fwd":[1000,1000],"hyd_press_a":[1000,1000],"hyd_press_b":[1000,1000],"noz":[1000,1000],"oil":[1000,1000],"oxygen":[1000,1000],"pitch":[1000,1000],"roll":[1000,1000],"rpm":[1000,1000],"yaw":[1000,1000]}</steppers_speed_accell>

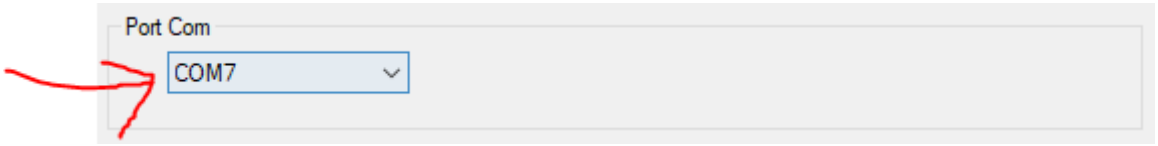
3.1.2 - Restauration de la configuration

Le chargement de la configuration se fait en cliquant sur le bouton « Load settings ».

Toute configuration actuelle sera remplacée par celle du fichier de configuration lors du chargement. Attention donc à ne pas écraser une configuration correcte car l'opération ne peux être annulée.

3.2 - Configurer les éléments actifs

Le principe de paramétrage des éléments actifs ou modules (jauges, voyants, afficheurs 7 segments et écrans) se fait en premier lieu en choisissant le port série dans les paramètres de l'élément concerné.



Chaque carte Arduino connecté à l'ordinateur possède son propre port Série.

Une fois connecté, la liste affiche le nom des ports séries disponible.

> *La liste des ports Com n'est pas rafraichie en temps réel. Pensez à branchez vos équipements avant de lancer le logiciel F4ToSerial*

3.2.1 - Les jauges

Généralités :

J'ai réalisé plusieurs tests avant de prendre la décision d'utiliser des moteurs pas à pas x27.168 pour les jauges car :

- Ils ne sont pas bruyants comparés à des servosmoteurs
- Leur angle de rotation est élevé par rapport à un servomoteur.
- Ils consomment peu de courant (peuvent être banchés sans alimentations externes en direct sur la carte Arduino)
- Ils sont très rapides.
- Le coût d'un x27 est très faible (moins de 2 € pièces).

Les jauges du cockpit utilisables dans F4ToSerial sont les suivantes :

Console centrale :

- Rpm
- Noz
- Oil
- Ftit

Console de droite :

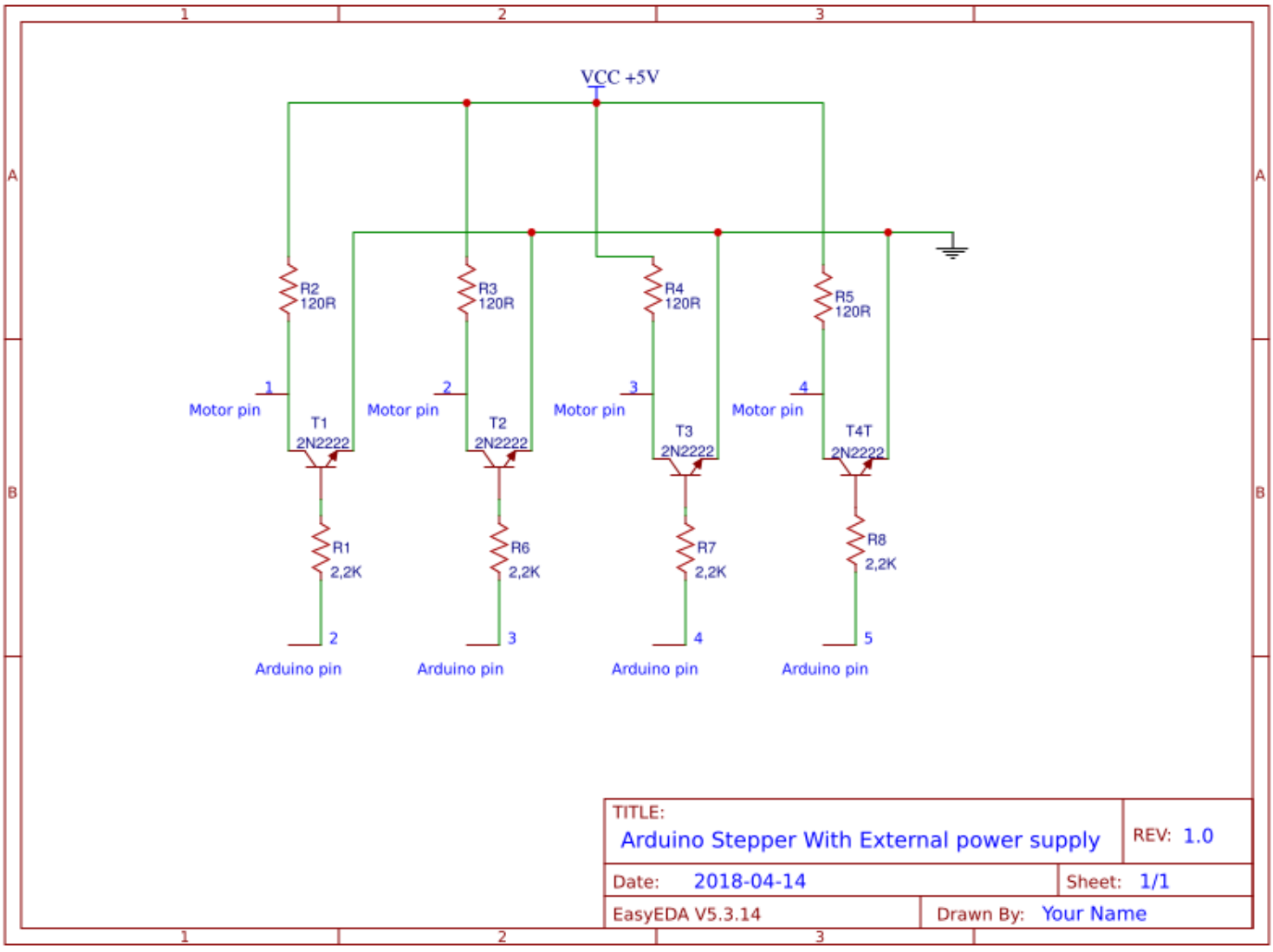
- Epu
- Fwd
- Aft
- Hyd Press A
- Hyd Press B
- Cabin

Console de gauche :

- Roll Trim
- Pitch Trim
- Yaw Trim

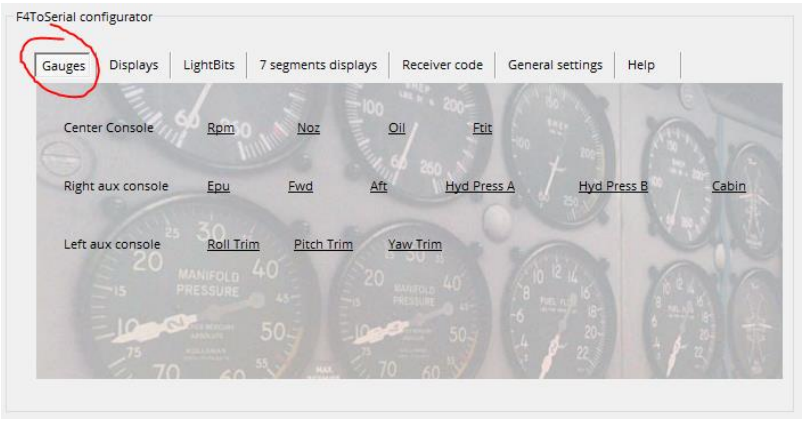
Une particularité existe pour la jauge Hyd Press. Cette dernière utilise 2 aiguilles sur le même axe. Dans ce cas de figure un autre type de moteur pas à pas doit être utilisé : Le x40.168

Le schéma ci-après illustre comment câbler le moteur x27 avec une alimentation externe. Cela permet de brancher tous les moteurs sur la même carte Arduino.



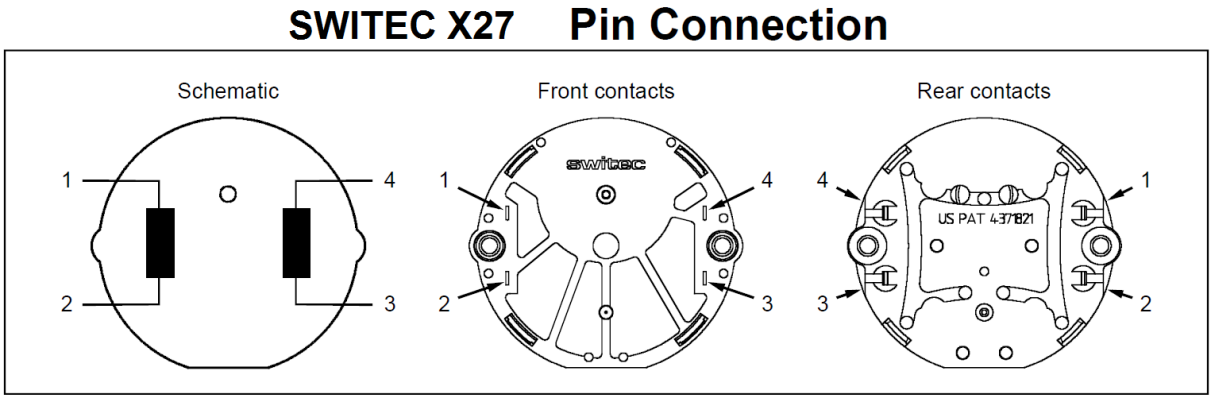
Utilisation dans F4ToSerial.

Dans F4ToSerial, les jauges sont configurées dans le premier onglet.



Cliquez sur le lien du module pour accéder à sa configuration.

Les moteurs pas à pas X27 ont 4 broches :

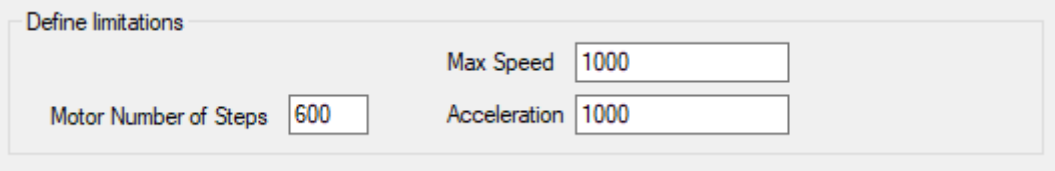


Ces 4 broches sont représentées dans F4ToSerial dans la zone « Define Pins »

Vous devez indiquer ici les numéros de broches correspondant sur votre carte Arduino.

> *Dans cet exemple qui n'est pas correct, la broche 1 du moteur est connectée au pin 0 de la carte Arduino et la broche 4 du moteur est connecté au pin 3 de la carte Arduino.*

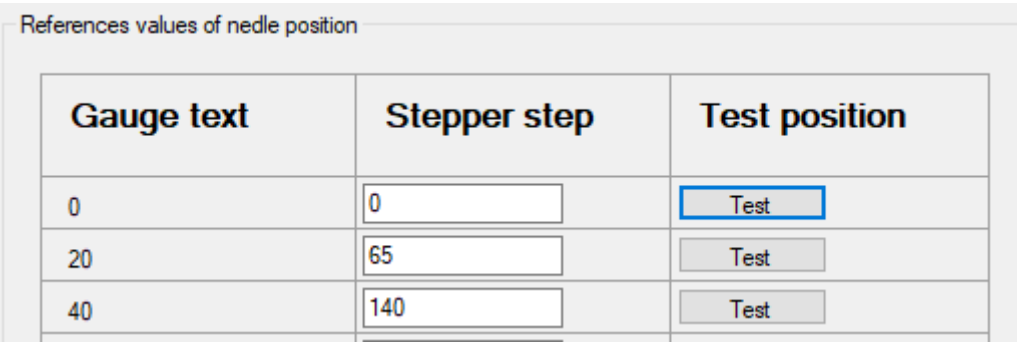
La zone suivante **« Define limitations »** permet de configurer les limites propres aux moteurs X27.



- **Steps** : Les moteurs pas à pas X27.168 sont des moteurs de 600 pas mais vous pouvez changer la valeur ici. Aucun intérêt, sauf si votre moteur est différent d'un x.27.168.
- **Max Speed** : Ce paramètre correspond à la vitesse maximum admissible par le moteur. Elle est calculée sur la base des infos de la librairie AccelStepper : http://www.airspayce.com/mikem/arduino/AccelStepper/
- **Acceleration** : Ce paramètre correspond l'accélération maximum admissible par le moteur. Elle est calculée sur la base des infos de la librairie AccelStepper : http://www.airspayce.com/mikem/arduino/AccelStepper/

> *Je recommande de faire attention si vous changez les valeurs des limitations du moteurs. Vous pouvez détériorer ce dernier ou afficher des valeurs erronées. Les valeurs 600, 1000 et 1000 sont issues de mes nombreux tests.*

La zone « **références values of nedle position »** permet de faire correspondre chaque jauge de vos cockpits aux valeurs des jauges du simulateur Falcon 4 BMS.



Etant donné que chaque cockpit est différent, ce système permet à tous les types de supports de jauges d'être utilisés. Dans mon cas j'ai utilisé un support de jauge qui provient du site internet : http://hispapanels.com/

Mais il existe d'autre support avec d'autres valeurs de jauges. Il est donc important d'avoir un système qui permet de faire correspondre l'ensemble des jauges de tous les cockpits.

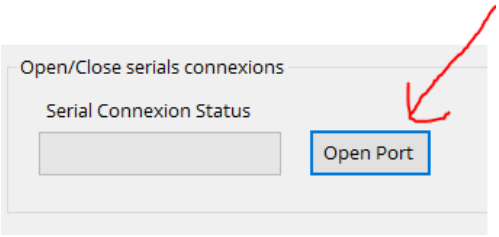
De plus, certaines jauges du F16 ayant des valeurs non linéaires, cette option permet de faire correspondre exactement les valeurs de votre jauge de votre cockpit à celle de Falcon BMS.

Les valeurs de références du simulateur sont indiquées dans la première colonne.

Vous devez faire correspondre vos valeurs de jauges avec celle du simulateur en indiquant un nombre de « pas » dans la colonne centrale.

Cliquez sur le bouton « test » pour essayer de faire coïncider les valeurs entres elles.

Attention, pour essayer de déplacer les aiguilles vous devez correctement bancher votre moteur sur la carte, fermer Falcon BMS et avoir ouvert au préalable les ports séries sur la page principale du programme F4ToSerial.



La photo suivante illustre des repères de correspondance entre une jauge des RPM sur Falcon BMS et une jauge des RPM dans un cockpit maison.

Vous devez pointer votre aiguille sur les valeurs 70 (point rouge), 80 (point vert) et 90 (point jaune) en cliquant sur le bouton test et avoir le même résultat sur votre jauge.

Retrouver plus d'aide sur ma vidéo ici :

https://youtu.be/dqMFBQkAozs?t=5m55s

Pour vérifier si vos jauges son calibrés vous pouvez lancer le simulateur Falcon BMS et vérifier le fonctionnement de vos jauges. Pour reconfigurer vos jauges vous devez fermer Falcon BMS.

3.2.2 - Les écrans

Généralités :

Il n'existe pas 36.000 solutions pour simuler un écran DED dans votre cockpit. Parmi la plus simple à mettre en œuvre l'utilisation des écrans OLED semble la plus intéressante.

Exemple d'écran OLED (taille 256x64 px) :

Exemple de FFI et DED en utilisant l'excellent programme DEDuino :
https://pit.uriba.org/tag/deduino/

F4ToSerial permet d'obtenir le même niveau de résultat, excepté l'effet de défilement des chiffres sur le FFI. Je développerais surement une mise à jour à ce sujet.

Pour fonctionner avec F4ToSerial, deux formats d'écrans sont acceptés : 256x24 et 128x64.

Ils sont disponibles un peu partout sur internet, mais F4ToSerial utilise exclusivement les modèles SPI plus rapide. Le bus I2C n'est pas implémenté dans le programme C++ ni dans F4ToSerial.

> *Je vous recommande de faire attention lors de l'achat de votre écran, car dans le cas de modèles I2C il faut parfois dessouder une résistance pour basculer le BUS en SPI.*

Dans les exemples liés à mon installation j'utilise deux modèles d'écrans :

Le modèle ER-OLEDM028-1Y dont la datasheet est disponible ici :

https://www.buydisplay.com/download/manual/ER-OLEDM028-1_Series_Datasheet.pdf

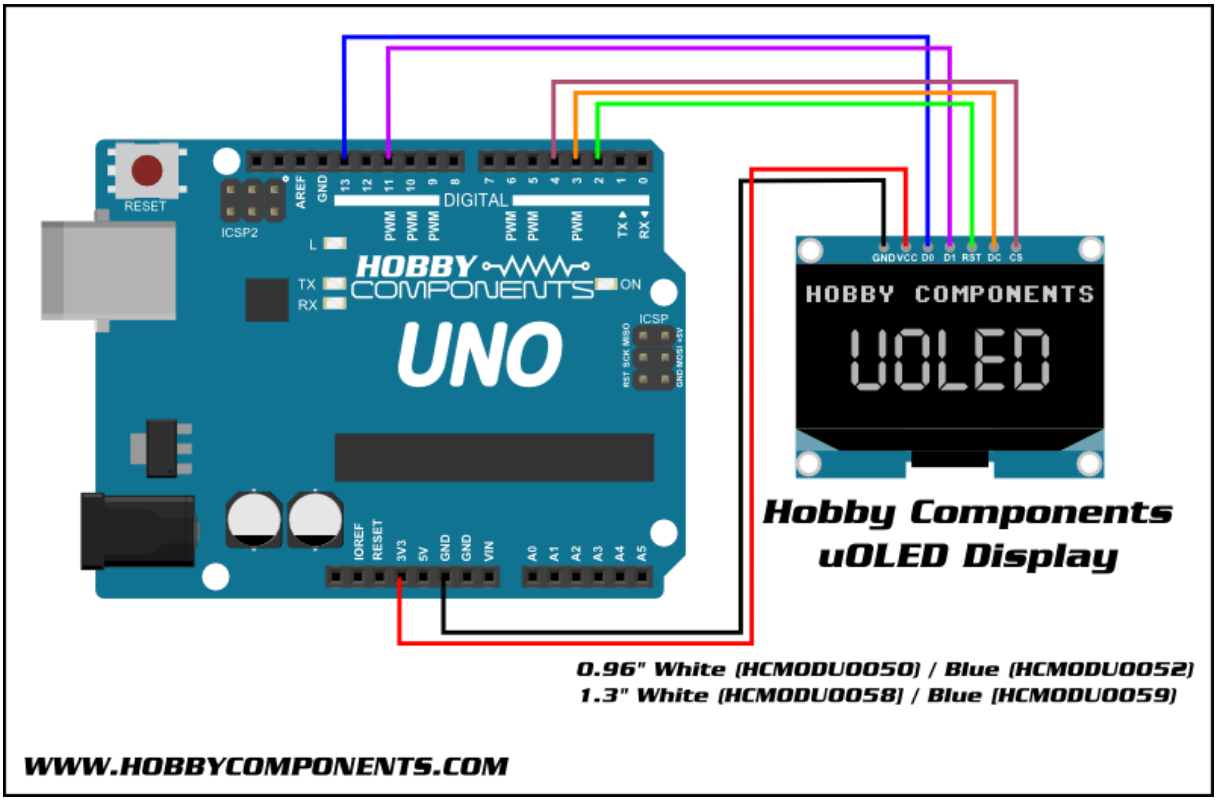
Ainsi qu'un modèle NONAME avec le chipset SSD1322 acheté sur AliExpress ici :

https://fr.aliexpress.com/item/Blue-Color-3-12-3-12inch-OLED-Display-Module-256x64-SPI-Communicate-SSD1322-For-Arduino-STM32/32819511393.html?spm=a2g0s.9042311.0.0.aSjjOi

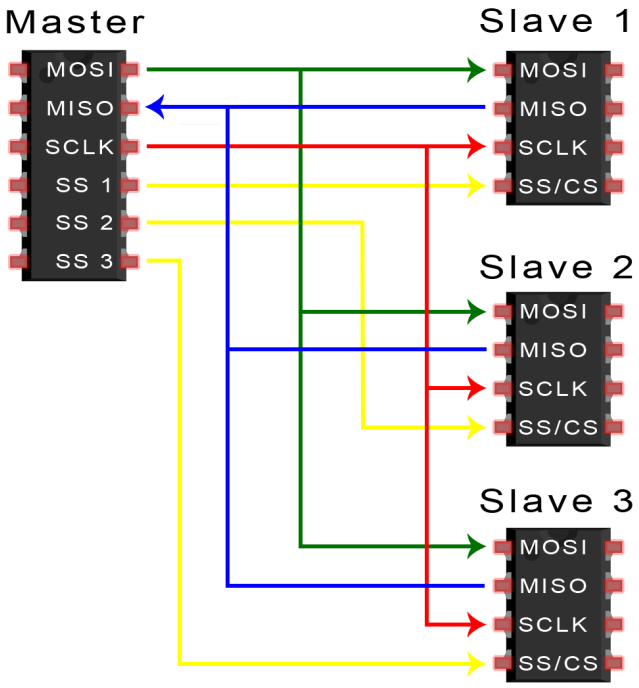
La communication sur le bus SPI utilise 5 fils :

1. Clock
2. Data
3. CS (Cable Select)
4. DC (Data Contrôle Command)
5. Reset

Cet exemple montre le branchement d'un seul écran 128x64 sur une carte Arduino Uno.

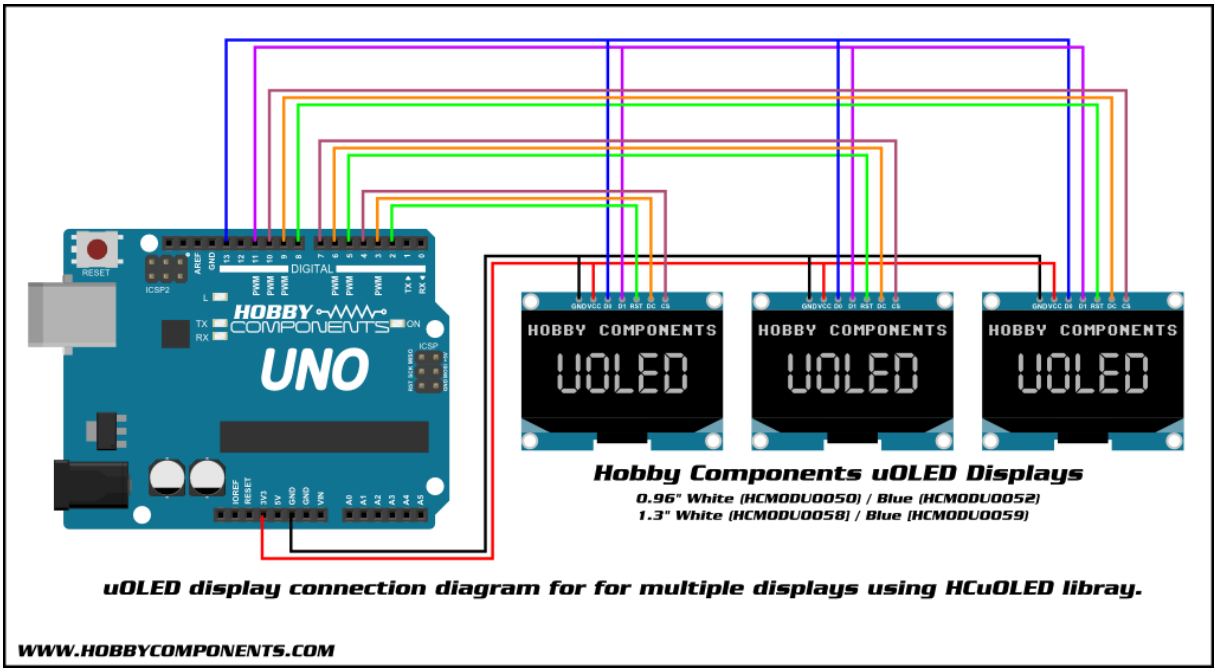


Dans le cas où on utilise plusieurs écran OLED sur le même port Série (la même carte Arduino), il faut relier les horloges (CLOCK/DO) entre-elles ainsi que les données (DATA / M0SI/D1).



RESET, DC et CS sont branchés séparément sur la carte.

Exemple de branchement :

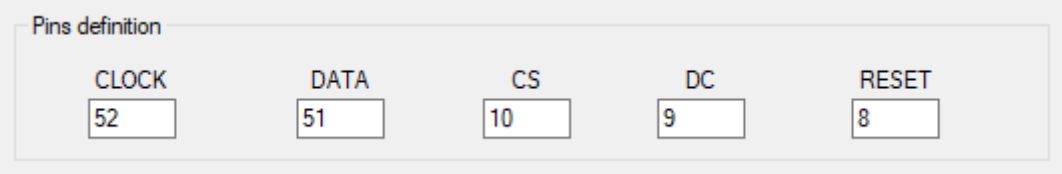


> *Des spécifications existent pour les cartes Arduino. Seuls certaines broches de la carte Arduino sont prévues pour transmettre l'horloge et les données. Veuillez-vous référer à la documentation Arduino.*
> *https://www.arduino.cc/en/Reference/SPI*

Utilisation dans F4ToSerial :

Accessible depuis le menu « Displays » du bloc de configuration, cette option permet de configurer les 3 écrans du cockpit : Le DED, PFL et FFI. Dans Falcon BMS, le dernier (FFI) n'est pas un écran à proprement parler. Mais son affichage reste possible.

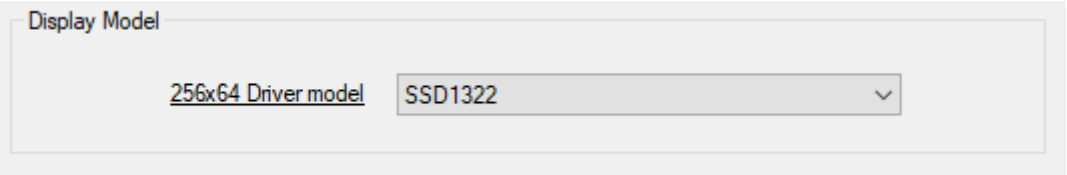
La première étape est la définition du port série puis ensuite des pins de l'écran OLED.



Indiquez ici les broches de votre carte Arduino correspondant au broches SPI de votre écrans OLED.

Par défaut les numéros de broches CLOCK (52) et DATA (51) correspondent à celles utilisées sur une carte Arduino Méga.

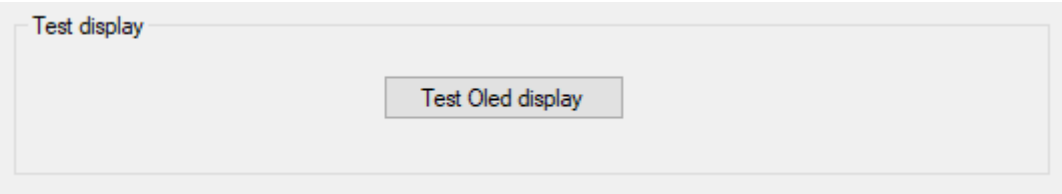
Indiquez ensuite le modèle du microcontroleur de votre écran :



F4ToSerial est compatible avec les chipsets OLED suivant :

- **Format : 256x64**
  - SSD1322 (défaut)
- **Format : 128x64**
  - SSD11306 (défaut)
  - SH1106
  - SSD1309
  - SSD1325
  - ST7565

Enfin, après avoir sauvegardé la configuration, et ouvert les ports Série, il est possible d'effectuer des tests pour s'assurer du bon fonctionnement de vos écrans OLEDS. Cliquez sur le bouton « TEST ».



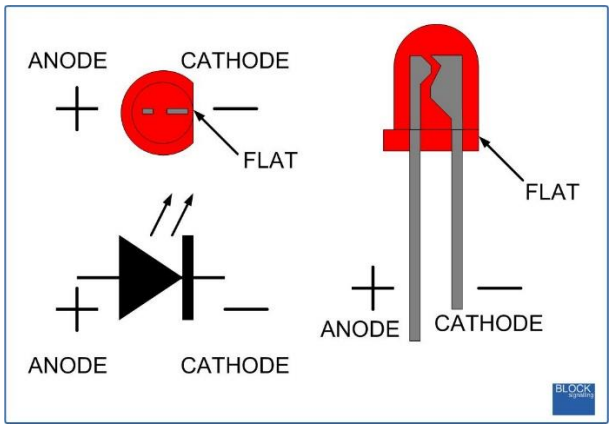
3.2.3 - Les leds ou lightBits

Généralités :

L'utilisation des LED ou DEL dans un cockpit est indispensable pour simuler les nombreux voyants allumés ou éteint dans le simulateur Falcon BMS.

C'est d'ailleurs le même type de composants qui sont utilisés dans tout véritable appareil.

Dans le simulateur Falcon BMS, les voyants sont appelés « LightBits » et leur état (allumé ou éteint) est transformé en « binaire » (O ou 1) dans la mémoire partagée.



Il y a deux approches pour l'utilisation des Lightbits. L'une « économique » et l'autre « on s'en fou ».

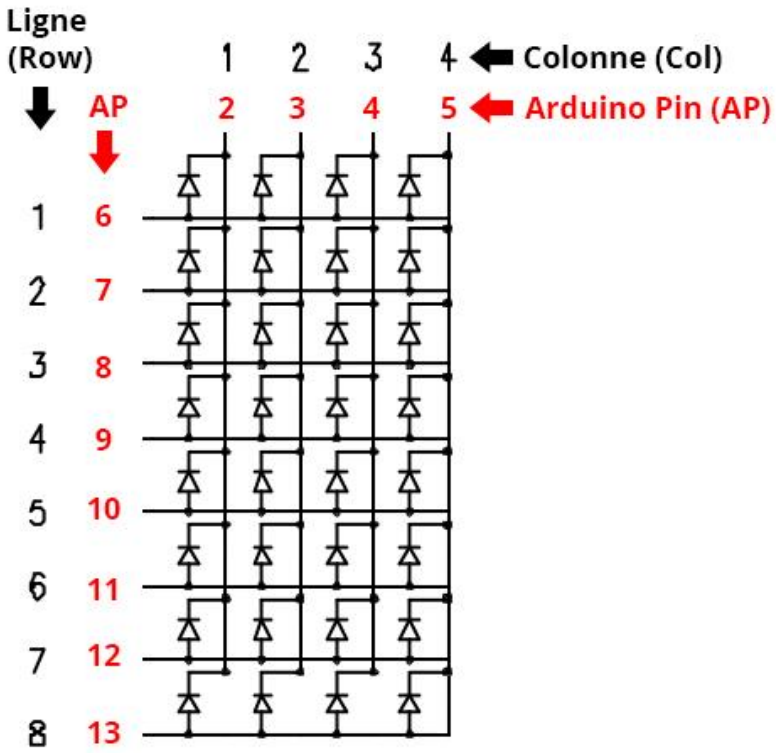
Dans l'approche « on s'en fou », il faut compter 1 broche de la carte Arduino pour chaque lightBits. C'est plus simple à câbler, mais cela occupe rapidement la totalité des broches de votre carte Arduino.

Dans l'approche « économique », 4 broches peuvent alimenter jusqu'à 16 lightbits. C'est ce qu'on appelle des « Matrices » ou « Matrix » en anglais.

Personnellement j'utilise les deux méthodes avec l'application d'une matrice pour certains composants comme le PFL par exemple.

Ci-après, le PFL du F16 Block 52.

Le PFL peut être représenté avec une matrice de ce type :



Bien sûr, il ne faut pas oublier de rajouter les résistances en entrées de chaque colonne.

Je ne vais pas m'étendre sur l'utilisation et le fonctionnement d'une matrice ici car il existe de très bons tutos sur Google en recherchant « LED Matrix » par exemple.

(Français) https://openclassrooms.com/courses/perfectionnez-vous-dans-la-programmation-arduino/concevez-des-matrices-de-led

(Anglais) https://www.arduino.cc/en/Tutorial/RowColumnScanning

La seule chose à savoir c'est que l'utilisation de la matrice ne permet pas d'avoir plusieurs LightBits allumés en même temps. Il faut donc ruser pour allumer plusieurs LED en même temps et pour cela utiliser l'effet de persistance rétinienne.

La bonne nouvelle, c'est que vous n'avez rien à faire, F4ToSerial s'en occupe pour vous !

Dans mon cas pour l'utilisation de LED branchées en direct sur la carte sans passer par une matrice, j'utilise une carte très pratique.

http://www.flightsimparts.eu/shop_ledextension.htm

Cette carte possède 2 avantages

1. Elle a déjà toutes les résistances
2. Elle utilise des transistors pour alimenter les LEDS avec une alimentation externe.

Sur ce dernier point, n'oubliez pas que plus vous branchez de composants (Afficheurs OLEDS, LED, Moteurs etc..) sur votre carte Arduino, plus vous tirez du courant ! Une carte Arduino ne pourra pas à elle seule allumer plus de 5 ou 6 LED !

Je recommande donc vivement l'utilisation d'une alimentation externe !

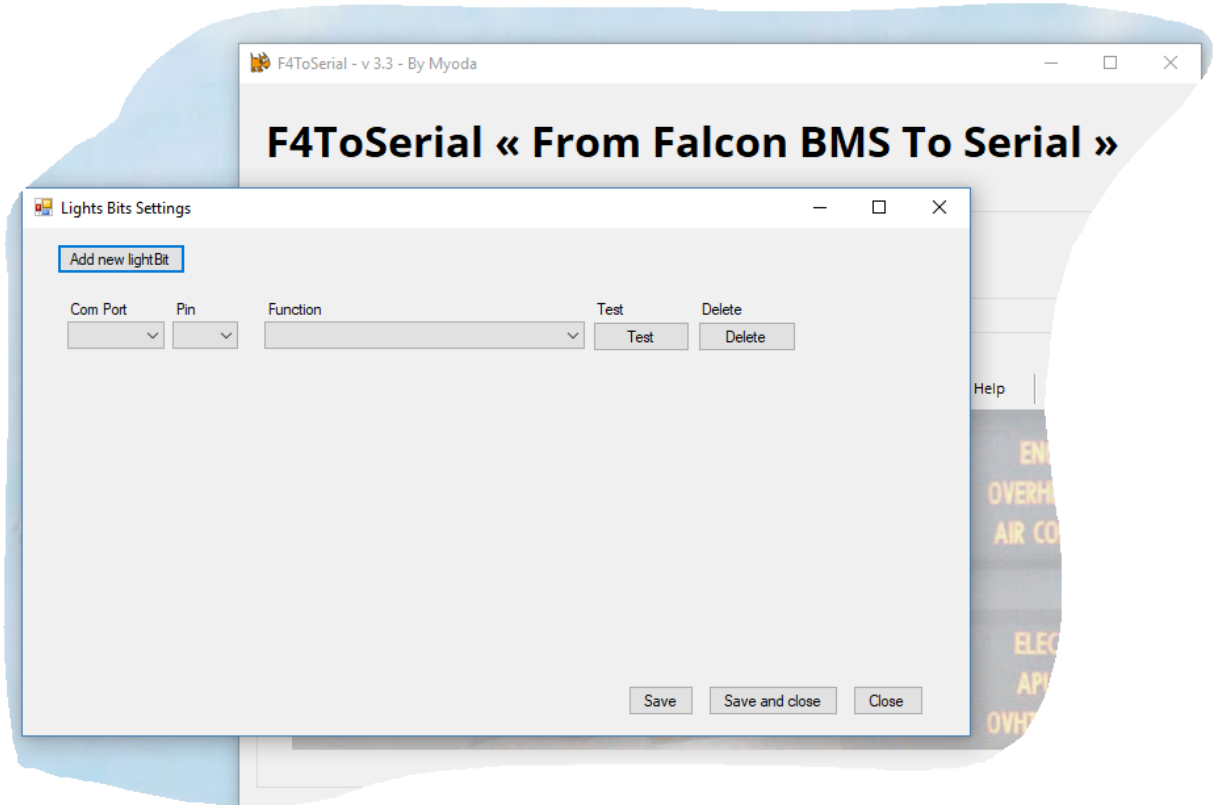
Dans le cas d'une matrice, seule une seule LED est allumée, ce qui ne consomme que peu de courant !

Utilisation dans F4ToSerial :

Pour configurer les LightBits dans F4toSerial commencez par cliquer sur l'onglet « LightBits »

Il y a 3 liens dans cette rubrique. Le premier lien « LightBits Digital output » sert à connecter les LEDS directement sur la carte. Les deux autres servent à créer et configurer une matrice.

Création d'un LightBit directement sur la carte « LightBits Digital output »



En cliquant sur « Add New LightBit », on ajoute une nouvelle « entrée » à notre tableau de LightBits.

Comme toujours il faut définir dans un premier temps le port série (Com Port).

Ensuite indiquez le « pin » ou « broche » de sortie de la carte Arduino correspondant à votre LightBit.

Enfin, définissez sa fonction :

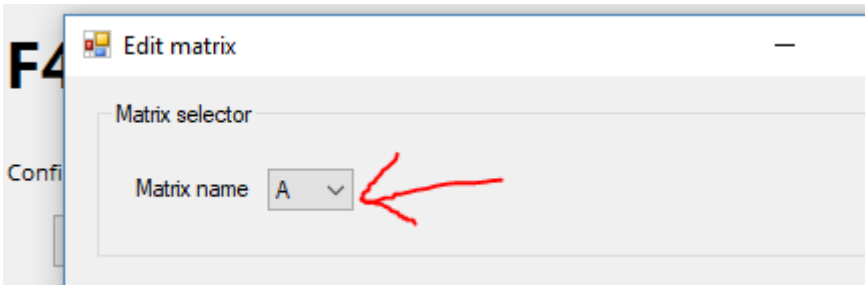
La colonne « Function » correspond à l'a fonction du lightBit dans le simulateur Falcon BMS. Je vous invite pour les tests à utiliser une fonction simple comme le « Crochet » ou le verrouillage du siège qui allume immédiatement le PFL.

Vous pouvez toujours également faire un test en appuyant sur (MAL & IND LTS) pour tout allumer.

<u>Création d'une Matrice de LightBits « Matrix configurator » et « LightBits matrix output »</u>

Cliquez sur « Matrix configurator »

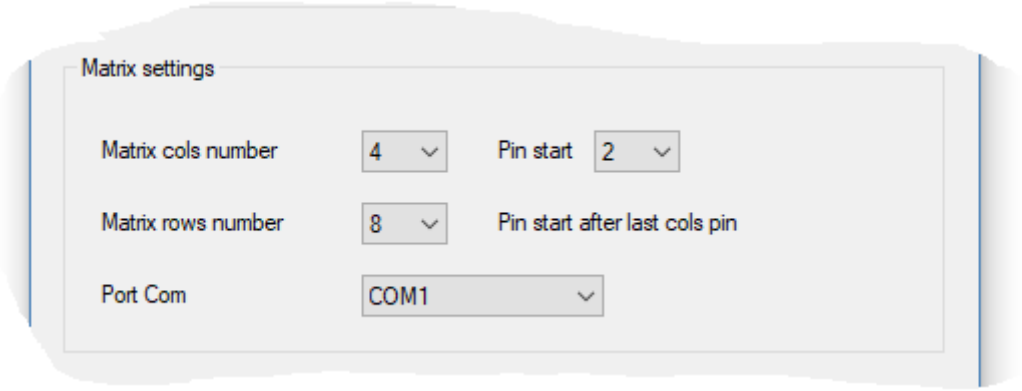
Puis commencez par choisir le nom de votre Matrice



F4ToSerial vous permet de créer jusqu'à 5 Matrices 8x8 (8 lignes et 8 colonnes).

Elles sont nommées A, B, C, D ou E.

Indiquez ensuite les 4 paramètres clés de la matrice :



1. Le nombre de colonne (Matrix cols number)
2. Le nombre de ligne (Matrix rows number)
3. Le port série (Port Com)
4. La broche de départ (Pin Start).

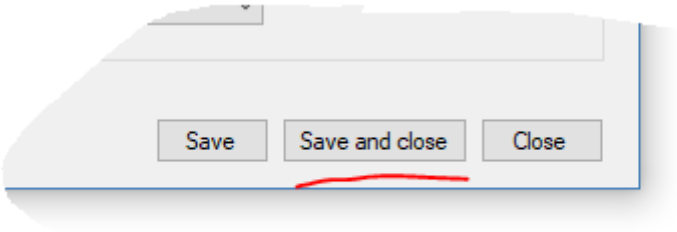
Le dernier paramètre (PinStart) est très simple à comprendre. Il indique à votre carte Arduino quelle est la première broche de votre matrice sachant qu'elles sont toutes dans l'ordre.

Vous devez toujours câbler votre matrice sur la carte Arduino en commençant par les colonnes avec le programme F4ToSerial.

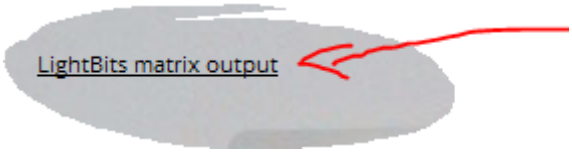
Dans l'exemple ci-dessus, la matrice PFL 4x8 (4 colonnes et 8 lignes) est envoyée au port Com 1, et les broches sont câblées à partir de la 2 jusqu'à la 13.

2,3,4 et 5 (4 colonnes) + 6,7,8,9,10,11,12 et13 (8 lignes).

Vous pouvez ensuite sauvegarder la configuration.



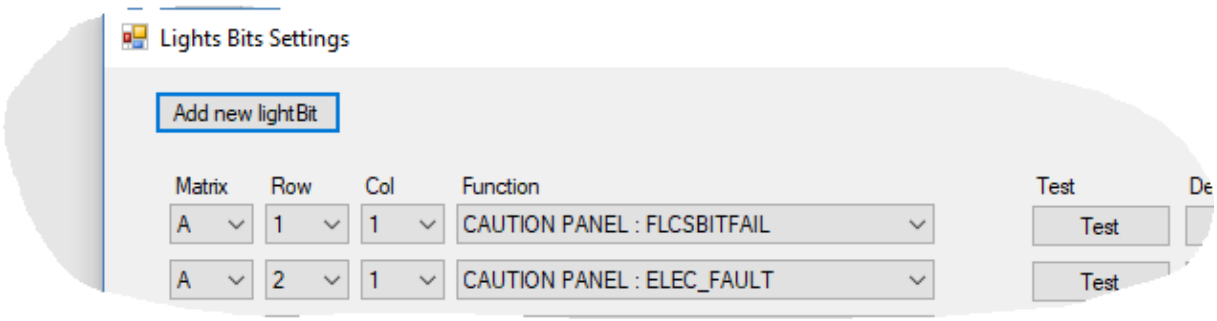
Il faut maintenant définir les LightBits et leurs fonctions dans votre matrice. Pour cela, cliquez sur « LightBits matrix output » dans la fenêtre principale.



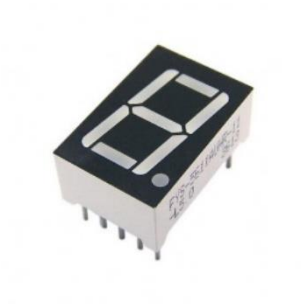
Le principe est le même que pour un lightBit connecté sur la carte Arduino directement sauf que maintenant vous devez indiquer

- La matrice concernée (Matrix)
- La colonne (Row)
- La ligne (col)
- La fonction (Function)

Votre lightBit se trouve à l'intersection de la ligne et de la colonne.

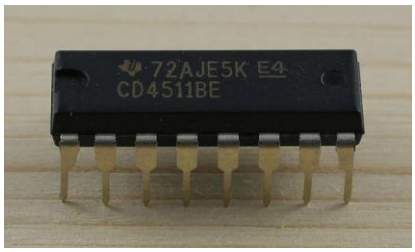


3.2.4 - Les afficheurs 7 Segments



Les afficheurs 7 segments existent depuis très longtemps et sont utilisés dans de nombreux appareils dont le F16 Falcon Block 52.

Ce composant, relativement simple à néanmoins un défaut : il est gourmand en câblage ! En effet, chaque « LED » de l'afficheur est une led à part entière et à besoin d'être alimentée pour fonctionner, tout comme un lightBit.

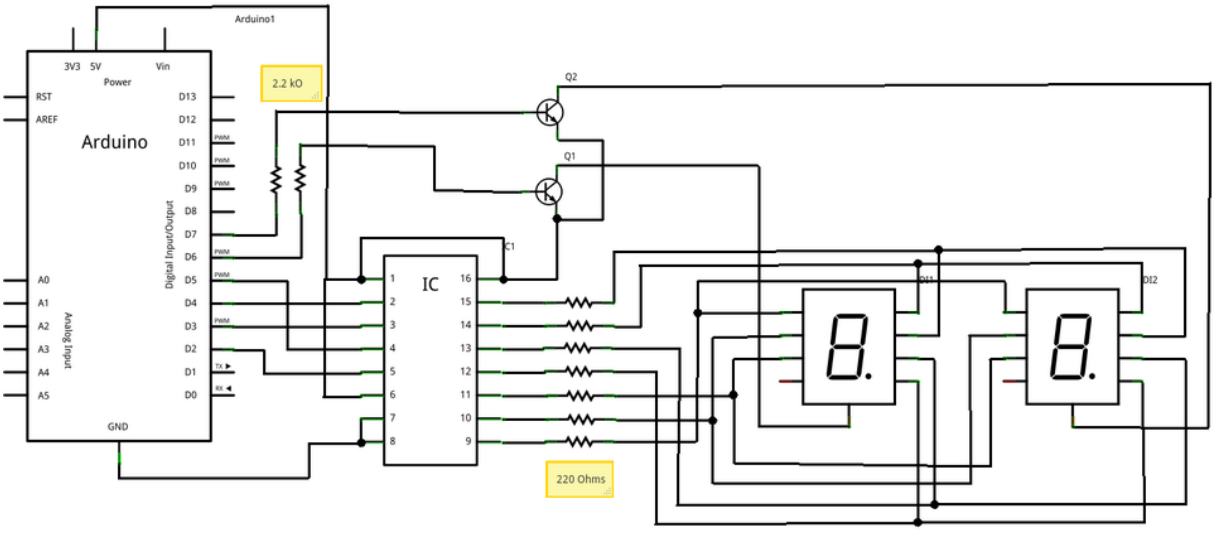


Pour éviter d'utiliser trop de broches sur la carte Arduino, il est donc fortement recommandé de passer par un décodeur BCD.

Encore une fois, je ne rentrerai pas dans les détails du fonctionnement du décodeur BCD, car de très bonnes documentations existent sur Google avec quelques recherches.

https://www.carnetdumaker.net/articles/utiliser-un-afficheur-7-segments-avec-une-carte-arduino-genuino/#question-piege-cathode-ou-anode-commune

Ci-après un exemple de branchement d'une carte Arduino vers un décodeur BCD et deux afficheurs 7 segments.



Notez que sur l'exemple ci-dessus, nous utilisons deux transistors bipolaire NPN. Ils sont utilisés pour commuter l'alimentation des afficheurs pour jouer sur l'effet de persistance rétinienne. Car tout comme pour un lightBits, quand on banche plusieurs afficheurs sur le même décodeur BCD, un seul peut être utilisé à la fois.

Les décodeurs BCD ne peuvent afficher que les valeurs de 0 à 9. Je vous invite à lire ce tutoriel pour bien comprendre le principe de fonctionnement :

http://eskimon.fr/tuto-arduino-205-afficheurs-7-segments

Dans F4ToSerial, pour utiliser un afficheur 7 segments, il faut passer par un décodeur BCD.

Les afficheurs 7 segments utilisable avec F4ToSerial sont :

- UHF FREQ
- UHF CHAN
- FUEL (Gauge)
- CHAFF/FLARES (CMDS pannel)

Attention :

Pour les CHAFF et FLARES, quand la quantité de capsule de CHAFF et FLARES atteind le niveau 0 l'indication « Lo » pour « Low » (minium) apparait devant le nombre de Chaff/Flares restant.

Visible également ici (sauve que 01 et 02 ne sont pas utilisé sur le F16)

Malheureusement les décodeur BCD ne peuvent afficher que les valeurs de 0 à 9.

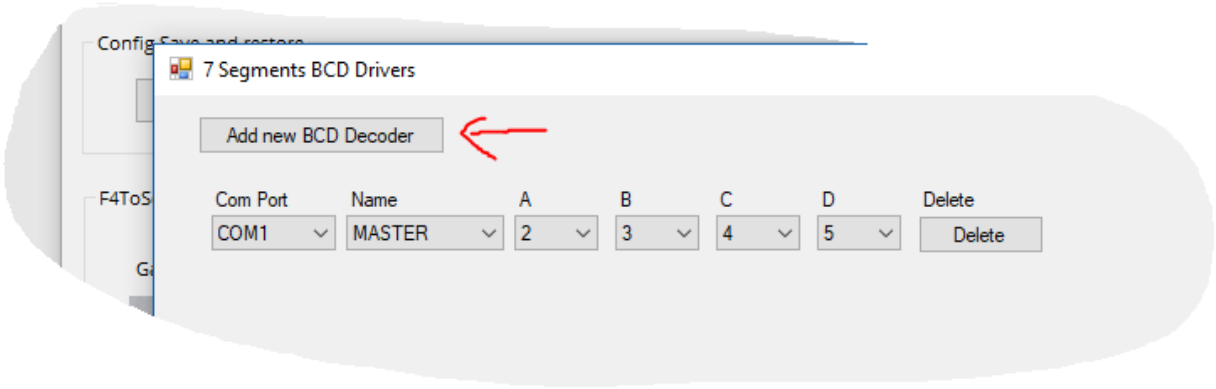
Il n'est pas prévu pour l'instant de mise à jour à ce sujet car cela ne gêne pas la compréhension du panneau CMDS et ce n'est pas une contrainte dite « forte ».

Utilisation dans F4ToSerial :

Commencez par cliquer sur l'onglet « 7 segments displays ».

Cliquez ensuite sur « BCD DECODER » pour définir et ajouter un nouveau décoder BCD.

Cliquez sur le bouton « Add new BCD décoder » pour ajouter une nouvelle ligne.



Pour chaque ligne vous devez définir les paramètres suivants :

- Le port série (Com Port)
- Le nom du décodeur (Name)
- La broche d'entrée A du décodeur BCD
- La broche d'entrée B du décodeur BCD
- La broche d'entrée C du décodeur BCD
- La broche d'entrée D du décodeur BCD

Etant donné qu'il est possible de répartir les décodeurs BCD sur plusieurs cartes Arduino, vous pouvez ajouter plusieurs lignes et vous devrez pour chacune choisir un nom à votre décodeur.

Le nom permet simplement de définir la fonction du décodeur. En choisissant par exemple « Master », vous définissez un décodeur qui aura plusieurs fonctions comme UHF, CAN, FLARES etc.

Ce nom sera utilisé par la carte Arduino pour identifier quel décodeur recevra les informations du simulateur Falcon BMS.

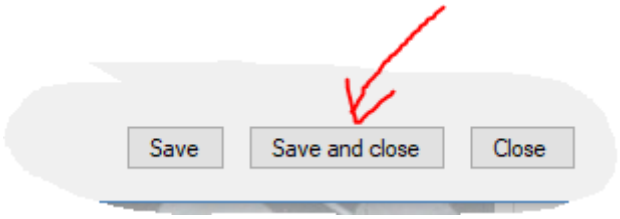
Exemple :

Sur une carte Arduino connectée sur le port COM2 vous utilisez un décodeur pour les afficheurs des CHAFF, FLARES et FUEL vous pouvez alors définir comme nom « Master ».

Sur une autre carte Arduino sur le port COM3 vous utilisez un décodeur pour les afficheurs de l' UHF CHAN et l' UHF FREQ vous pouvez alors définir « UHF » ou « UHFCHAN » ou « UHFFREQ » comme nom.

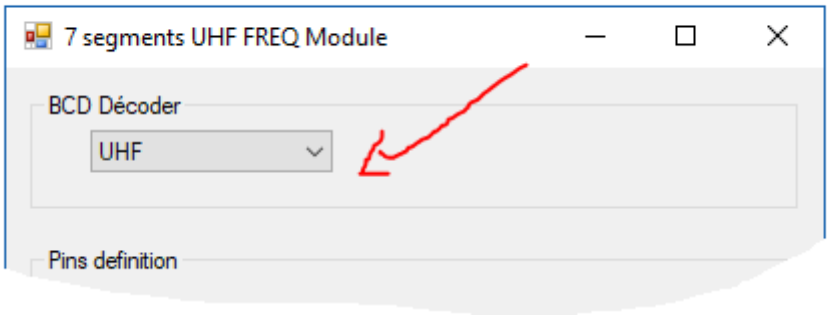
Les broches A, B, C et D du décodeur BCD sont à définir également et je vous invite à rechercher la documentation de votre décodeur pour plus d'informations si vous ne comprenez pas cette partie.

Une fois que vous avez fini les paramètres et enregistré la configuration :



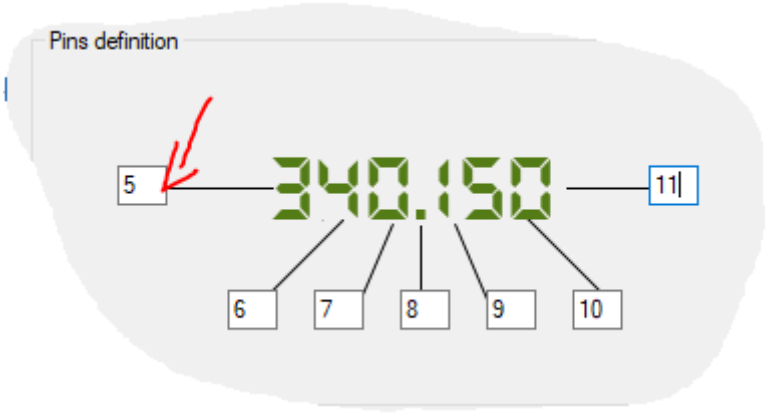
Cliquez sur le l'afficheur dans la fenêtre principale.

Puis sélectionnez le décodeur BCD pour cet afficheur.

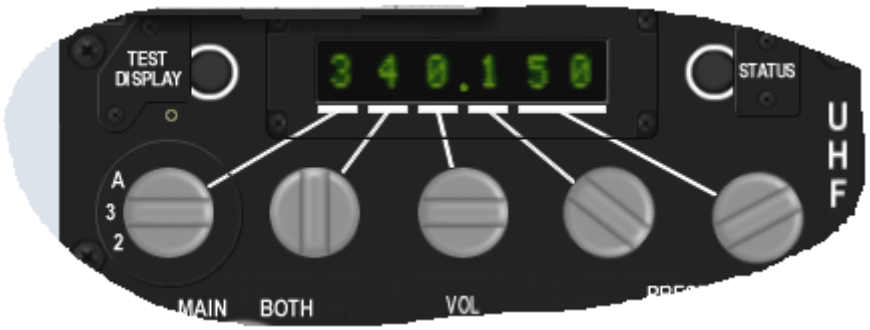


Une fois le décodeur sélectionné indiquez ensuite pour chaque afficheur (représenté ici par un chiffre en vert) le numéro de broche correspondant.

Dans cet exemple il s'agit des 6 chiffres qui composent l'UHF FREQ (7 avec le point).

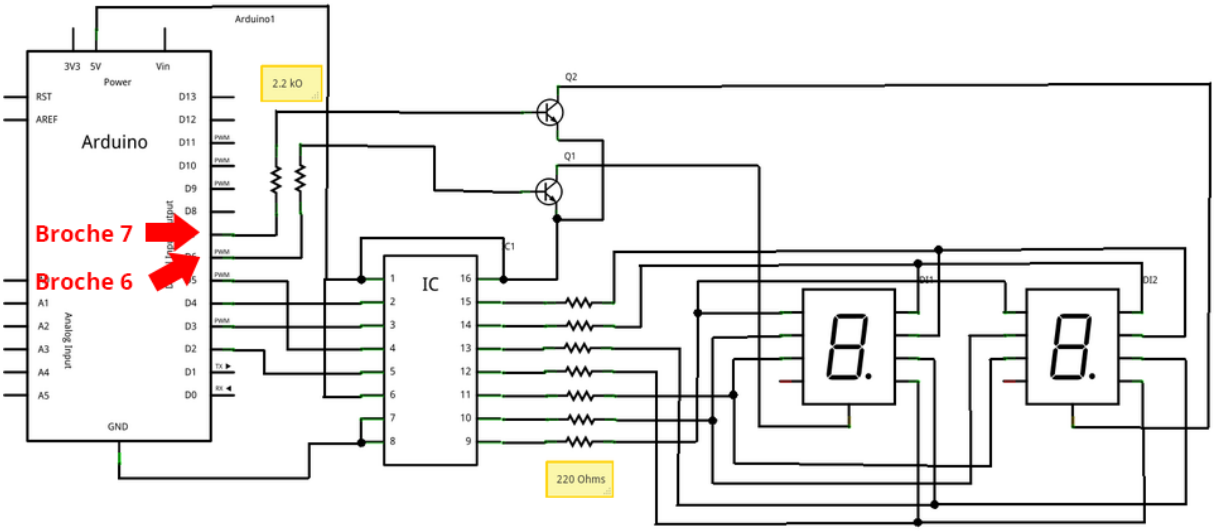


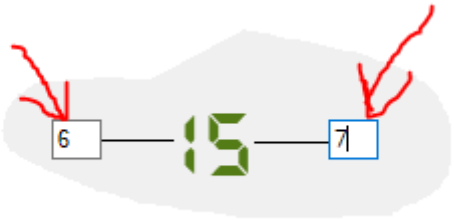
Cette fenêtre représente ce panneau.



Pour mieux comprendre comment définir les broches qui permettent de choisir l'afficheur je vous invite à vous représenter ce schéma.

Exemple :





3.3 - Paramètres généraux

3.3.1 - Télécharger la dernière version

Télécharger la dernière version est important pour garder le logiciel à jour.

Pour cela cliquez sur le bouton « Check version ».

3.3.2 - Taux de rafraichissement

Le taux de rafraichissement correspond à la durée écoulée entre chaque nouvel envoi d'information à votre carte Arduino à partir du programme F4ToSerial.

Attention, cette vitesse est calculée et dépends de la vitesse de transfert (Baud Rate).

- Plus vous réduisez ce nombre plus vous envoyez des informations rapidement à votre carte Arduino. **Le risque à la saturation du « buffer » de votre carte Arduino.**
- Plus vous augmentez ce nombre plus le temps s'écoule entre les informations. **Le risque est de constater un retard sur les éléments de votre cockpit par rapport à Falcon BMS (exemple jauge qui ne suivent pas).**

3.3.3 - Vitesse du port série

La vitesse du port série correspond à la vitesse à laquelle chaque « bit » de données est transféré du programme F4ToSerial vers la carte Arduino.

Arduino supporte nativement les fréquences suivantes.

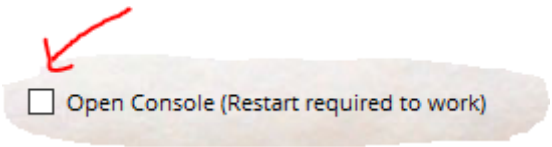
300, 600, 1200, 2400, 4800, 9600, 14400, 19200, 28800, 38400, 57600, or 115200

> *Mon expérience m'a montré que la carte Arduino peut supporter bien au-delà de 115.200 bauds. En pratique, je vous invite à utiliser une vitesse supérieure pour éviter tout décrochage des éléments actifs notamment des afficheurs OLEDS.*
>
> *Je recommande l'utilisation de 1.000.000 comme vitesse minimale de transfert.*

3.3.4 – Afficher la console de débug

La console est un mode très utile pour le débogage de votre application. Quand vous utilisez F4ToSerial pour la première fois, ou que vous êtes en cours de configuration d'un module.

Elle permet d'afficher les données qui transite sur la carte et vous alerte en cas de problème dans le programme.



Lorsque vous cliquez sur « Open Console » vous devez relancer F4ToSerial.

4 Bugs et problèmes possibles

Rien ne se passe quand j'ouvre les ports séries pourquoi ?

Si rien ne se passe quand vous ouvrez les ports série et que vous cliquez sur les boutons de tests des éléments (lightbits, moteurs etc.)

Commencez par regarder dans la console si vous avez des informations qui sont bien envoyées.

Si rien ne se passe, vous envoyez probablement les données vers une carte à la mauvaise vitesse (bauds). Pour cela, vérifiez que le programme de votre carte Arduino communique à la même vitesse que le programme F4ToSerial.

Dès que j'ajoute plus de 6 LEDS ou plus de 2 jauges ça ne fonctionne plus pourquoi ?

Si vous tirez trop de courant sur votre carte Arduino cette dernière ne pourra pas fonctionner correctement. Vous devez alimenter vos éléments avec une source externe.

D'une manière générale vous devez branchez tous vos équipements sur une source externe autre que la carte Arduino, elle-même alimentée par la prise USB de votre ordinateur.