**Automatisation du banc de test BMS3 V3**

**Intérêt** :

Réaliser l’automatisation des tests de la carte BMS3 (produit de **Vaudoo Audio – CECIELEC** qui s’intègre dans des guitares basses électriques.

En effet, les tests post production, disponible sous :   
***\\CECICLOUD\Projets\Vaudoo-Audio\BMS3 - BMS plug'N'play\production\Procedures\_de\_test\  
210902-BMS3-Procedure\_de\_test.docx***,   
nécessitent entre 6 et 7 minutes par cartes, du fait d’un nombre de manipulation importante et de temps d’attente incompressible.

En automatisant ces tests, la durée des tests passe à environ une minute. De plus, un rapport de test, au format ***xlsx***, est généré automatiquement et met en avant les résultats de test négatifs.

**Pré-requis et informations complémentaires** :

Afin de se soustraire des temps d’attente incompressible, dû aux temps d’affichage de l’état de la batterie et au temps d’attente suite au débranchement du jack, le firmware de la BMS3 intègre à présent un mode de test.

Afin d’activer le mode de test de la BMS3, avant sa mise sous tension, il faut mettre sous **+2V5** l’entrée **JMP\_18V** de la BMS3.

Ensuite l’entrées **JMP\_18V** permet l’activation des LED, lors d’un front descendant, et leur désactivation, lors d’un front montant. Les LED s’allument en suivant le cycle suivant :   
BLEU -> VERT -> ROUGE.

De plus, l’entrée **DEBUG\_RX** permet l’envoie sur la sortie **DEBUX\_TX** de la mesure de la tension de la batterie.

Le banc de test est contrôlé par l’application disponible sous sous :   
***\\CECICLOUD\Projets\Vaudoo-Audio\BMS3*** ***- BMS plug'N'play\Banc de test\Programme (.py etc..)\TestBench\***.

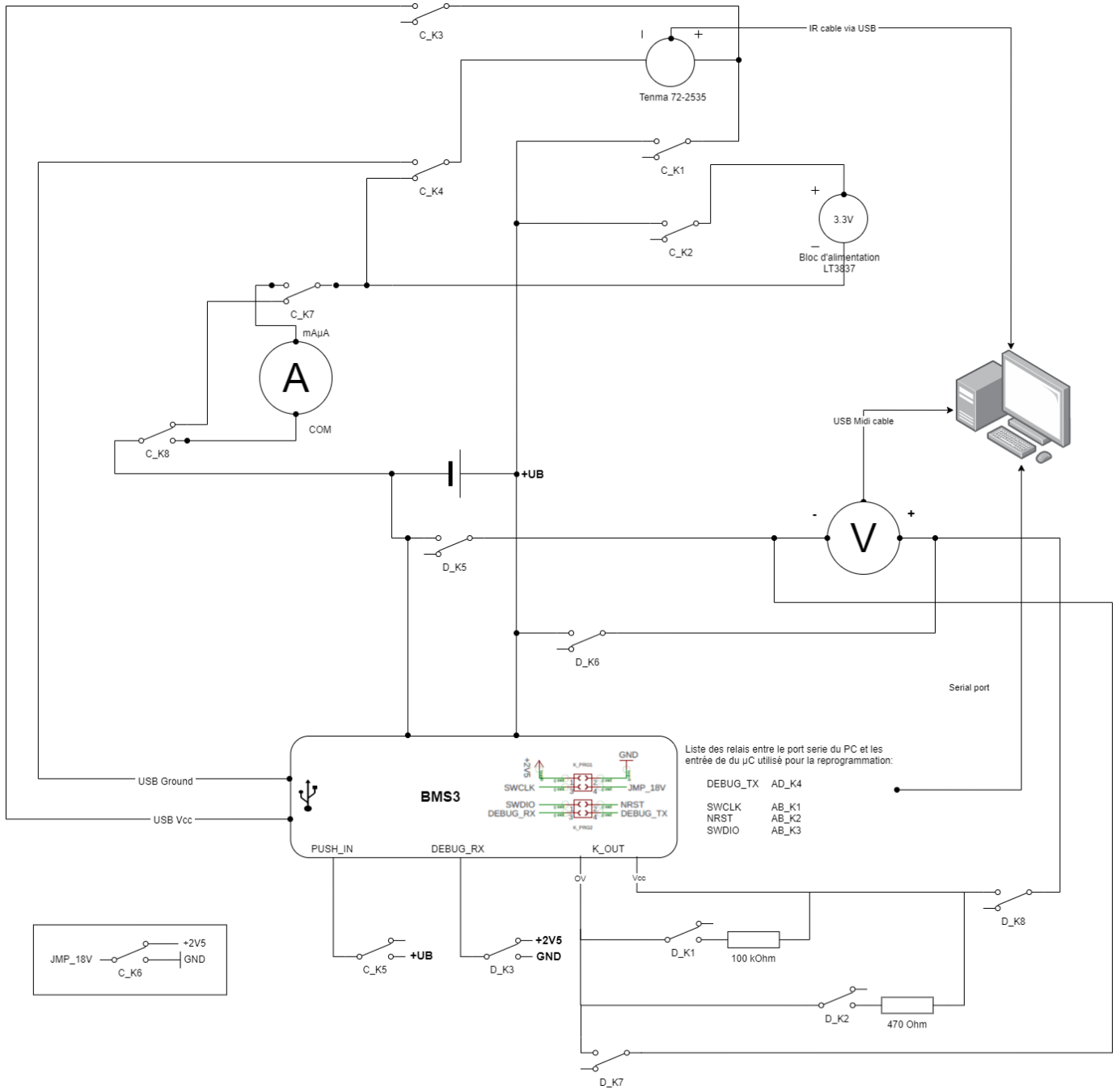
Sous Windows, il peut être lancé via le script **RunProject.bat**. Pour les autres systèmes d’exploitation, les lignes de commandes sont disponibles dans ce même script et doivent être adapté au terminal utilisé.

**Présentation du banc de tests :**

Le banc de test est composé :

* D’une alimentation **TENMA 72-2535**, qui sera piloté par l’application, celle-ci doit donc être **connecté au PC via un cordon USB Type B**.
* Du bloc d’alimentation LT3837, qui est utilisé pour la vérification de la charge de la batterie.
* D’une carte FT4232H Mini Module permettant de simuler 4 ports COM et sur laquelle sont reliées 3 cartes à relais ICSE014A (cf. Schéma du banc de test).
* De deux multimètres **TENMA 72-7730A**, l’un utilisé en voltmètre et l’autre en ampèremètre, chacun reliés au PC avec un câble USB **TENMA 72-13125** (cf. Schéma du banc de test pour connaître leur emplacement en fonction des relais permettant leur connection/déconnection). La **fonction SEND** devra être **activé** en restant appuyer sur le bouton ayant la sous-fonction SEND.

Schéma du banc de test



**Réalisation de la carte électronique intégrant la carte** **FT4232H Mini Module :**

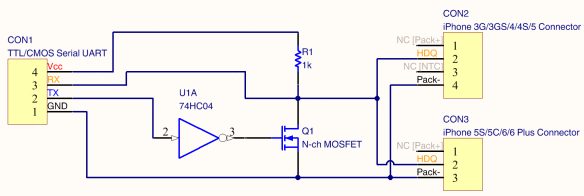
Afin de pouvoir simuler sur le PC au moins 3 ports COM, une carte FT4232H Mini Module est configurée en ce sens. De plus, elle est intégrée sur une carte afin de pouvoir y interfacer les cartes à relais ICSE014A qui seront piloter via ses ports séries. REPRENDRE ICI

Pour une meilleure ergonomie et palier le nombre important de fils devant être connectés sur une plaque de prototypage, il a été réalisé une carte sur plaque à pastilles contenant le mini-module FT4232H, le circuit nécessaire au bus I2C, le montage permettant la conversion du protocole HDQ en UART et les connecteurs pour l’I2C, l’HDQ et les UART des relais.

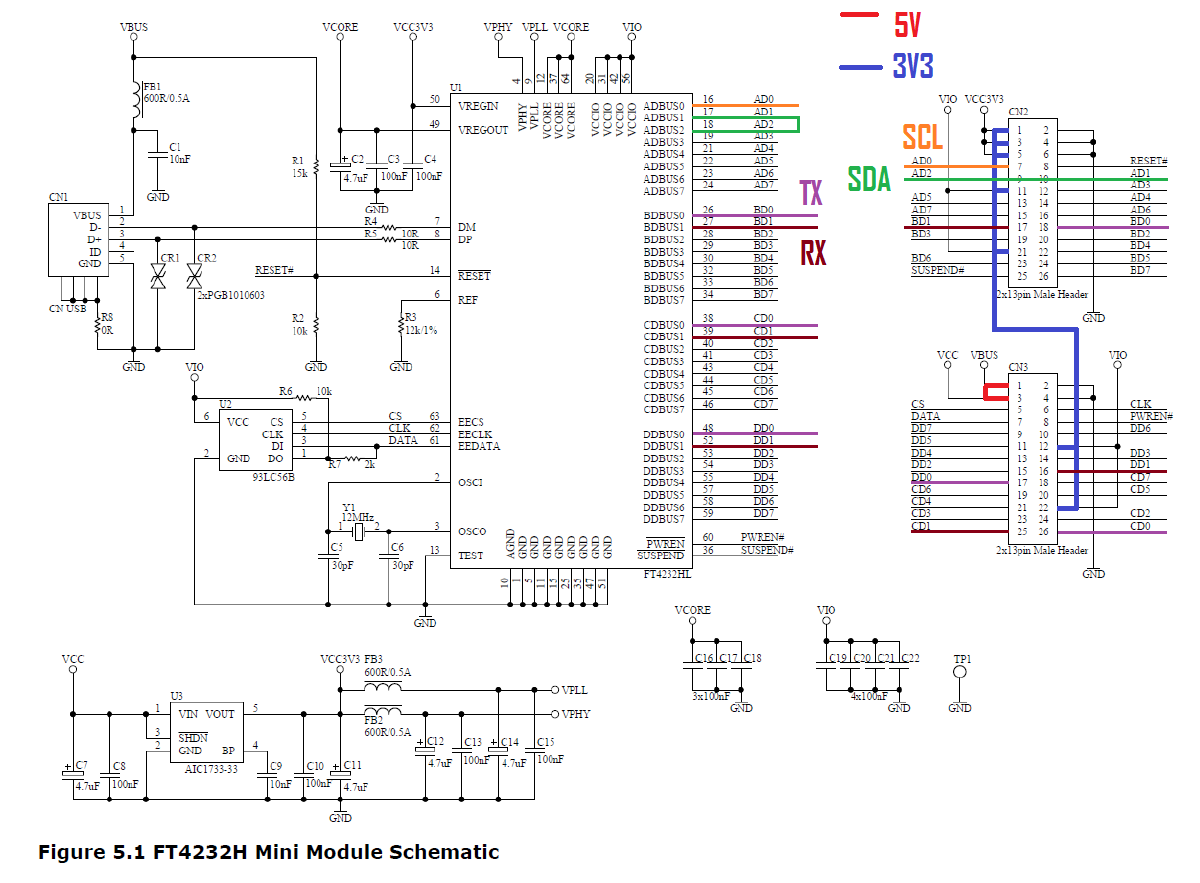
Le FT4232H nécessite un câblage supplémentaire pour l’alimenter.

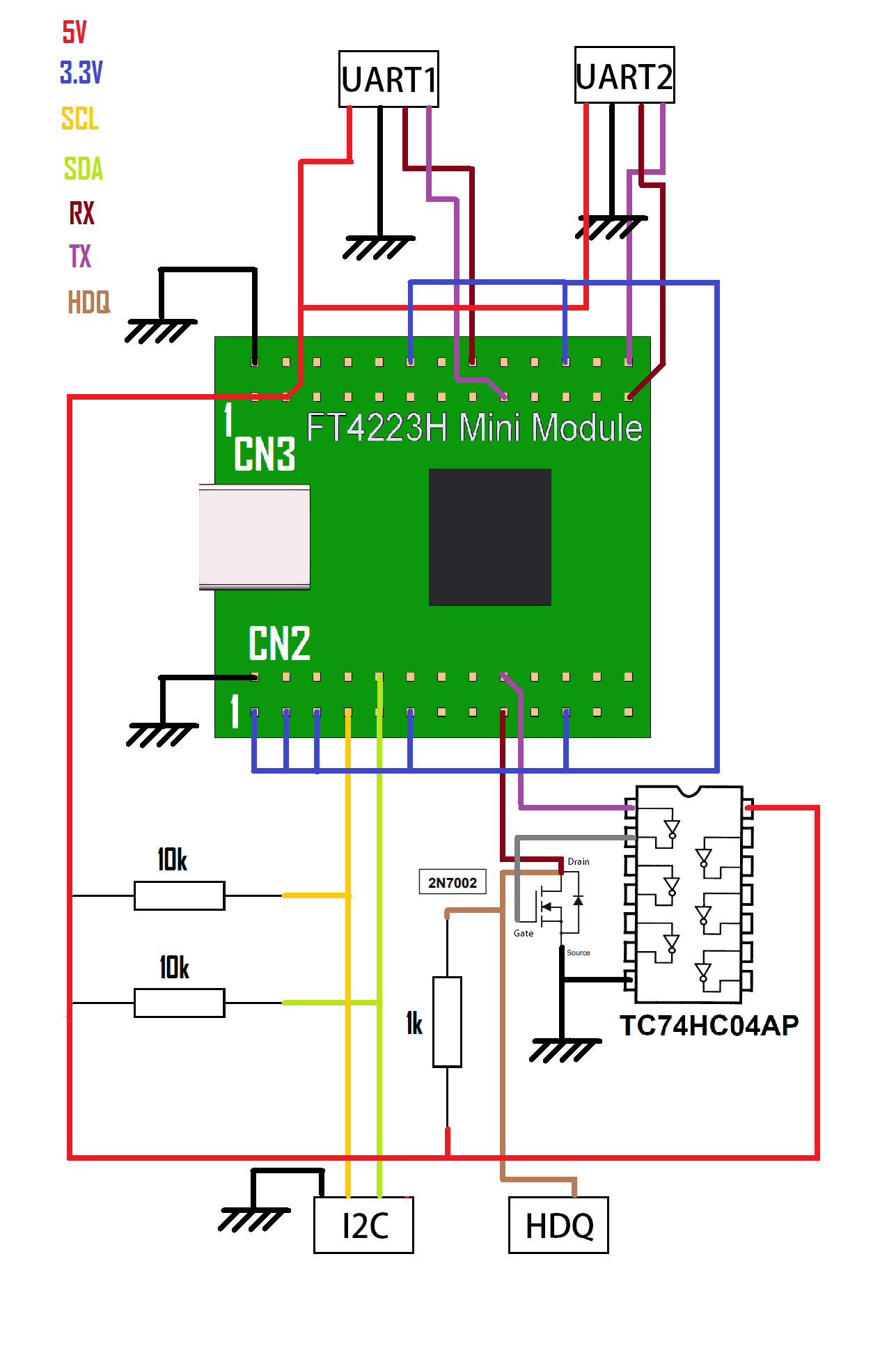
Pour l’I2C, il faut simplement connecter deux résistances (choisies à 10kohm) en pull-up sur les lignes SDA et SCL.

Pour l’HDQ, il faut réaliser un montage de collecteur ouvert.

[](https://ripitapart.files.wordpress.com/2014/09/hdq-adapter1.png)

Le schéma de câblage du FT4232H ainsi que de la carte développée sont disponibles ci-dessous :





**Interfaçage matériel / logiciel :**

Tout le matériel choisi se connectant finalement par USB à l’ordinateur du banc de test CECIPROD, il faut s’assurer que le système Windows puisse s’y connecter.

Pour les multimètres TENMA, la connexion se fait par périphérique HID. Les drivers ont été installés et sont disponibles au téléchargement.[[1]](#footnote-1)

Pour le FT4232H, nous utilisons les drivers sous forme d’un DLL chargé par le programme du banc de test. Les drivers généraux D2XX[[2]](#footnote-2) (pour les protocoles synchrones) et VCP[[3]](#footnote-3) (Virtual Com Port pour permettre au système de détecter les channels UART) sont disponibles au téléchargement, ainsi que le soft FT\_Prog[[4]](#footnote-4) qui permet de configurer le module ainsi que les canaux : le channel A est configuré pour des protocoles synchrones.

L’I2C s’utilise par l’envoi de commandes précises au FT4232H qui permettent de configurer ce dernier en maître et d’initialiser une communication avec l’esclave BMS.

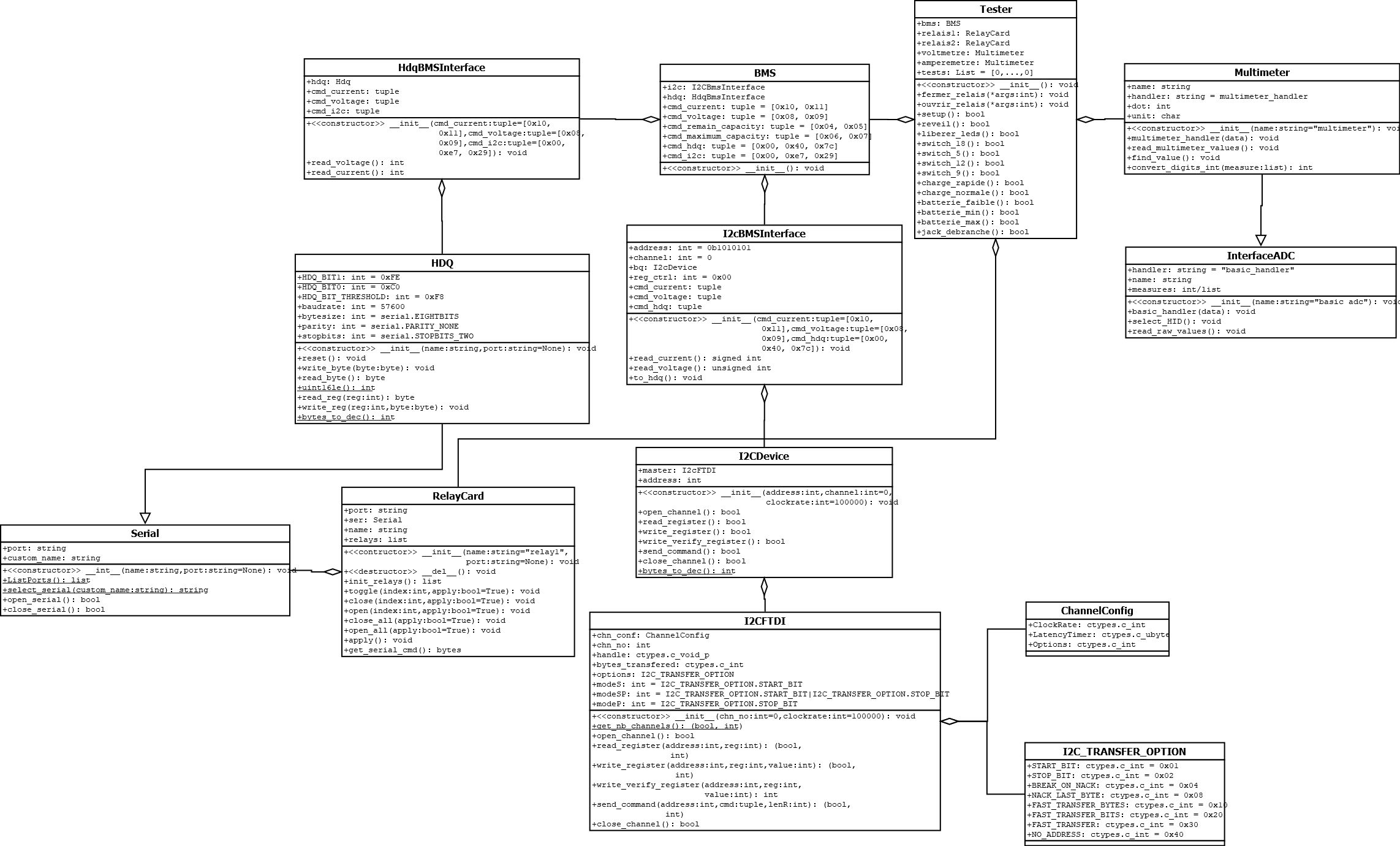
L’HDQ et les relais se pilotent directement via des trames série sur les ports reconnus nativement par les systèmes UNIX.

Pour l’HDQ, il faut cependant configurer la connexion série de manière très précise : Baudrate à 57600, pas de bit de parité et deux bits de stop. Enfin, chaque mot de 8bits transmis par une interface UART sera représentatif d’un bit de protocole HDQ : 0xFE pour 1 et 0xC0 pour 0. La note de TI plus complète explique en détail ce changement de protocole[[5]](#footnote-5).

Pour la communication avec les relais, le baudrate à respecter est de 19600.

**Identification des sous-systèmes :**

Avant de se lancer dans la programmation, il convient d’identifier les différents éléments du système et ses objets, ainsi que les fonctionnalités associées. Un diagramme de classes UML a été réalisé sur le logiciel DIA :



**Programmation du banc de test :**

Le langage choisi pour ce banc de test est Python 3.6. Il a été choisi de travailler en POO, avec des classes représentant chaque matériel et interface de communication et respectant la classification précédente.

Libraries notables utilisées :

* Pyserial : classe dont hérite notre classe Serial pour la communication série (UART et HDQ)
* Ctypes : utilisation des types et appel de la librairie C MPSSE pour le FTDI « *libMPSSE.dll* »
* Pywinusb : utilisation du module HID pour la communication HID avec les multimètres

Le programme dispose d’un main qui (après avoir salué l’opérateur) effectue pour le moment l’ensemble des tests de la classe Tester. Celle-ci comporte des fonctions de test dont chacune a été pensée et réalisée afin que l’environnement nécessaire au début du test ne dépend pas de manipulations précédentes. Chaque intervention de l’opérateur nécessaire (changement des valeurs du DIP-SWITCH et validation des LEDS) est indiquée sur le terminal.

Les classes I2CBmsInterface et HdqBmsInterface sont celles utilisées directement par le Testeur et implémentent des méthodes transparentes pour la lecture des registres nécessaires ou l’envoi de commandes spécifiques.

**Utilisation :**

Il faut premièrement connecter les ports I2C et HDQ de la BMS à la carte du banc de test, qui est à brancher par USB à l’ordinateur de test.

Il faut également connecter les deux modules de relais aux connecteurs UART de la carte.

Il est nécessaire de connecter les deux TENMA par USB, de les placer sur le bon calibre ainsi que de maintenir le bouton « SEND » appuyé pour configurer l’envoi infrarouge automatique. Il faut également changer de RANGE pour se placer à 200.

Lancer le soft 72\_7730 V3.00 à chaque connexion de multimètre. Il peut être nécessaire de ne brancher qu’une des deux multimètres, de l’initialiser avec le soft, puis de connecter le deuxième pour faire de même.

**Reste à faire :**

Implémenter le passage de I2C à HDQ ( 0x00 0x40 0x7C : écriture de 0x7C40 dans 0x00)

Programmer la flash par I2C (UNSEAL : 0x36 0x72 0x04 0x14)

Récupérer la valeur de CHEM\_ID (0x08) pour valider la communication I2C a début du test

Test à ajouter : Envoyer la commande de lecture les registres de charge restante et de charge maximale (commandes 0x04 et 0x06)

**Améliorations à apporter par la suite :**

Menu pour choisir le test à effectuer si test d’une carte réparée dont il est nécessaire que de tester certaines fonctionnalités.

Export des résultats de tests de Tester dans le fichier Excel.

Utilisation de tkinter

1. <https://www.element14.com/community/docs/DOC-28371/l/tenma-72-7730-software-file-data-transfer-instructions-file> [↑](#footnote-ref-1)
2. <http://www.ftdichip.com/Drivers/D2XX.htm> [↑](#footnote-ref-2)
3. <http://www.ftdichip.com/Drivers/VCP.htm> [↑](#footnote-ref-3)
4. <http://www.ftdichip.com/Support/Documents/AppNotes/AN_124_User_Guide_For_FT_PROG.pdf>

   <http://www.ftdichip.com/Support/Utilities.htm#FT_PROG> [↑](#footnote-ref-4)
5. <http://www.ti.com/lit/an/slua408a/slua408a.pdf>

   <http://www.ti.com/lit/an/slva101/slva101.pdf> [↑](#footnote-ref-5)