Utilisation de VISI-Genie sur 4D workshop

Implémenter sur une électronique embarquée un contrôle-commande avec un écran tactile, associant :

- **Rapidité de développement,**
- **Qualité visuelle,**
- **Evolutivité,**
- **Compatibilité UART avec n'importe quel microcontrôleur.**

Auteur : Gerardo VILLAVICENCIO SANCHEZ – Stagiaire DUT au CEMES Electronique
Juin 2017

Contact : Christian PERTEL – CEMES Electronique

Avec ce logiciel on est capable de programmer un écran LCD 4D Systems par le moyen d'un outil de programmation graphique.

Matériel utilisé :

- Ecran LCD tactile (uLCD – 43PT)
- Carte microntroleur NUCLEO L4
- Carte SSD SanDisk 2GB

Logiciels utilisés :

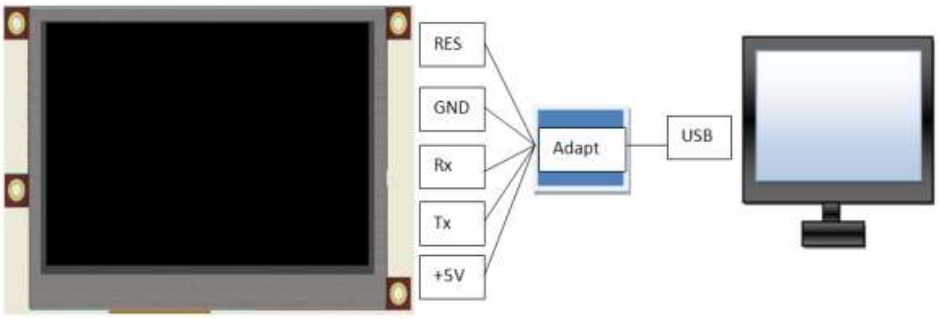
- System Workbench for STM32 (programmation en C)
- STM32CubeMX (Configuration des ports de la carte NUCLEO)
- 4D Workshop

Tout d'abord, avant de rentrer dans le codage en C, il faut programmer l'interface graphique de l'écran LCD dans le but d'avoir une représentation visuelle de ce qu'on veut comme options disponibles au niveau de ce dernier.

Pour ce tutoriel on va voir de quelle façon on peut utiliser VISI-Genie pour, d'une part, modifier le cote esthétique de l'écran et, d'autre part, pour donner des fonctionnalités réelles aux différents éléments présents sur l'interface graphique de l'écran.

Interconnexions STM32-Nucleo avec LCD 4D Systems :

Interconnexions PC avec LCD 4D Systems :



➔ Etape 0 : démarrage du logiciel

Apres avoir téléchargé le logiciel 4D Workshop sur le site officiel de 4D Systems, il faut le démarrer et configurer le projet.

D'abord cliquez sur l'onglet « New », puis choisissez l'écran dont vous disposez et cliquez sur « Next » :

Maintenant il faut ouvrir le logiciel Visi – Genie qui nous permettra de programmer graphiquement l'écran LCD.

Sauvegardez le projet dans le dossier de votre choix. A ce stade, vous êtes prêt à faire de la programmation graphique.

➔ 1ere étape : mise en place des éléments de base

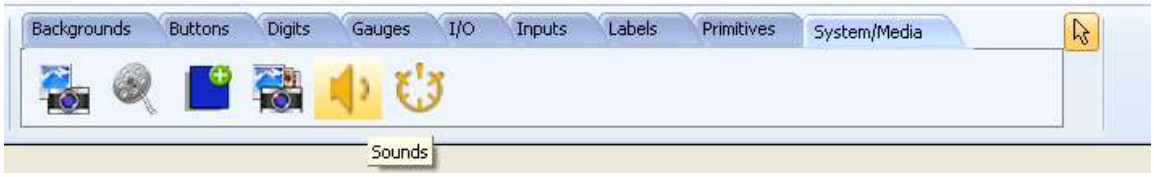
VISI-Genie propose une grande gamme d'options en termes de boutons, jauges, afficheurs, LEDs, sons etc…

REMARQUE : Pour ce tutoriel on va plutôt se concentrer sur les boutons, afficheurs, LEDs et le son et les chaines de caractère !

- LES BOUTONS

- AFFICHEURS ET LEDS

- LE SON



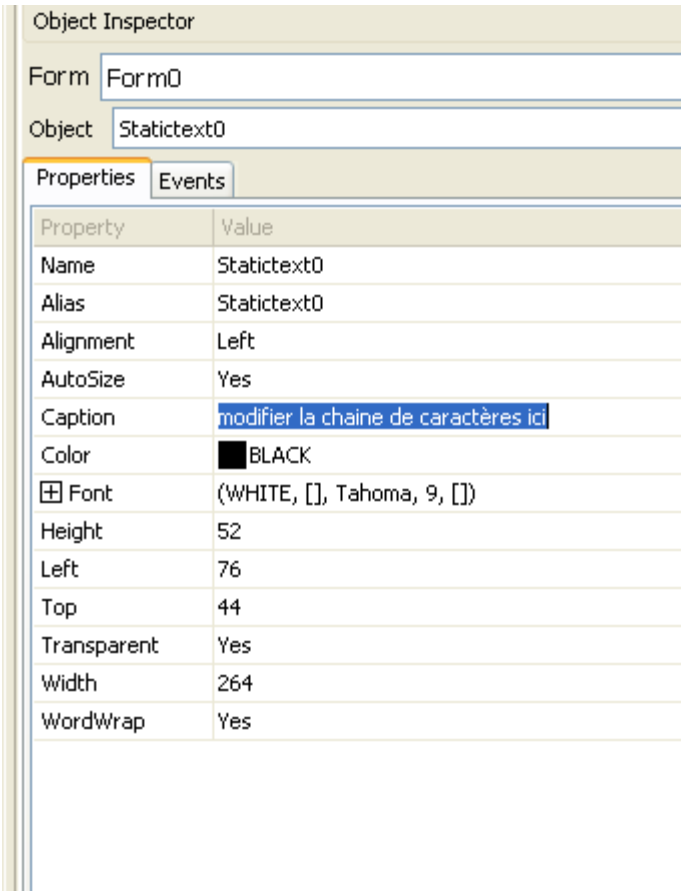
- LES CHAINES DE CARACTERES

Pour mettre en place l'élément souhaité cliquez dessus et cliquez sur l'endroit où vous souhaitez le placer au niveau de l'écran LCD.

EX : Voici le cas de mon projet dont le but principal de l'écran LCD est de piloter une platine moteur.

REMARQUE : Pour le son, on est limité à un seul son par fichier et il vous faudra le fichier .wav de ce son pour pouvoir le mettre dans le projet.

En ce qui concerne les chaines de caractères, il faut juste sélectionner l'option « StaticText » dans l'onglet « labels » et le placer dans l'endroit souhaité de l'écran :



Pour modifier la chaine de caractères il faut juste modifier l'option « Caption » dans l'onglet « Properties » de la chaine sélectionnée.

➔ 2e étape : configurer les éléments de l'écran

Pour configurer un élément il vous faut juste cliquez sur l'élément souhaité et changer ses propriétés :

Les définitions importantes à tenir en compte :

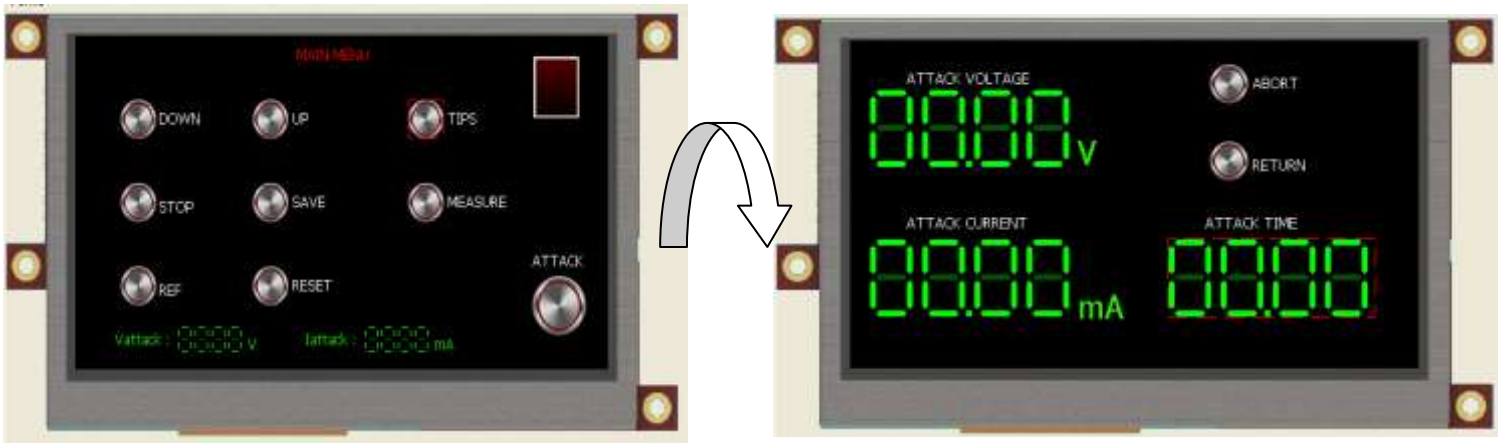
Form : Panel dans lequel se trouve l'élément sélectionné, ce logiciel donne la possibilité d'avoir plusieurs panels qu'on peut afficher (ex : panel Form1 s'affiche lorsqu'on appuie sur le bouton TIPS).

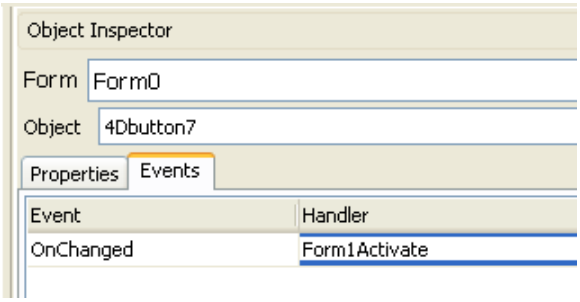
Object : Le nom de l'élément.
Left, Top : Coordonnée X et Y au niveau de l'écran.
Momentary : Pour ce projet j'ai toujours laissé ce paramètre en « Yes ».
Size, Width: Taille de l'élément.

En plus de changer les propriétés graphiques des différents éléments, on peut aussi les configurer de telle façon à ce qu'ils génèrent une action quand on les active en les appuyant (pour le cas d'un bouton).

- **Comment changer de panel affiché au niveau de l'écran LCD en appuyant sur un bouton ?**

Ex : changement de panel en appuyant en appuyant sur MEASURE

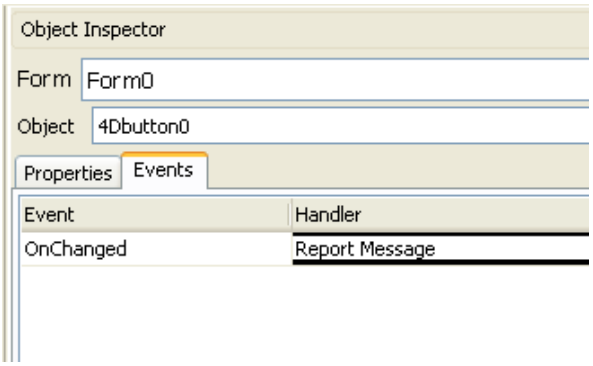




Pour pouvoir faire ça, il faut juste configurer le bouton MEASURE dans l'onglet «event» du panel des propriétés et on met l'action qu'on veut qu'il réalise (dans ce cas c'est Form1Activate).

- **Comment générer un message par ligne Tx en appuyant sur un bouton ou en changeant de « *form* » ?**

Le logiciel graphique donne aussi la possibilité à l'utilisateur de pouvoir envoyer une trame à travers la liaison Tx (transmission des données) au dos de l'écran. Cette trame nous va être utile par la suite pour la programmation en C au niveau de la carte NUCLEO L4.



Pour que le bouton ou la « form » envoie une trame par liaison Tx il faut configurer son action comme « *Report Message* » et c'est cette propriété qui va permettre à l'écran LCD de gérer la carte nucléo pour piloter d'autres systèmes (ex : un moteur rotatif).

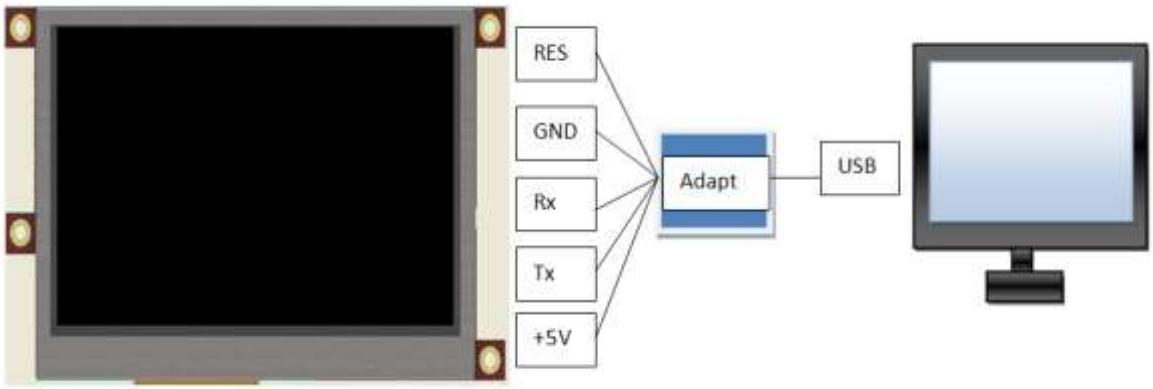
D'autre part, l'écran LCD est aussi équipé d'une ligne Rx (réception des données) qui va lui permettre de recevoir des trames depuis le microcontrôleur. Ces trames vont être interprétées par l'écran LCD et va lui donner des ordres pour faire des actions (ex : changer de « form », allumer une LED, faire un affichage dynamique, etc…)

Voir explications plus détaillées en étape 4

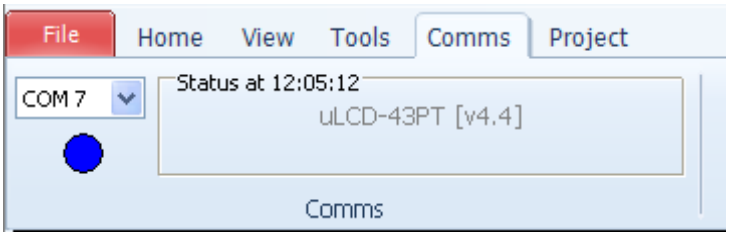
➔ 3e étape : Charger le programme a l'aide d'une carte SSD et test

Maintenant, qu'on a fini de customiser l'écran LCD avec Visi – Genie, il faut charger ce programme à l'aide d'une carte SSD.

- D'abord brancher l'écran (au dos) à l'ordinateur à l'aide du câble USB approprié :



- Vérifier que l'écran a bien été reconnu par le logiciel :

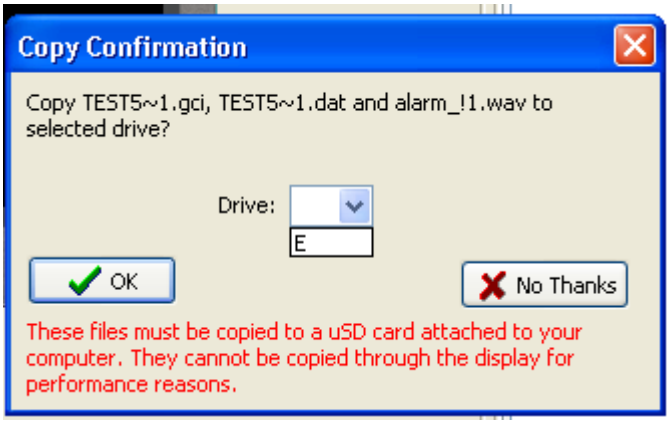


- Mettre la carte SSD au dos de l'écran LCD et vérifiez sa compatibilité en appuyant sur « uSD Tester » :

- Enlever la carte du dos de l'écran et branchez-la à l'ordinateur à l'aide d'un adaptateur pour charger le programme avec l'option « Build / Copy / Load » :

- Après la fin de la compilation, une fenêtre apparaitra automatiquement et vous permettra de télécharger le fichier au niveau de la carte SSD. Choisissez la racine et cliquez sur « OK »:

REMARQUE : N'oubliez pas d'enlever les vieux fichiers de la carte SSD avant d'enregistrer les nouveaux.



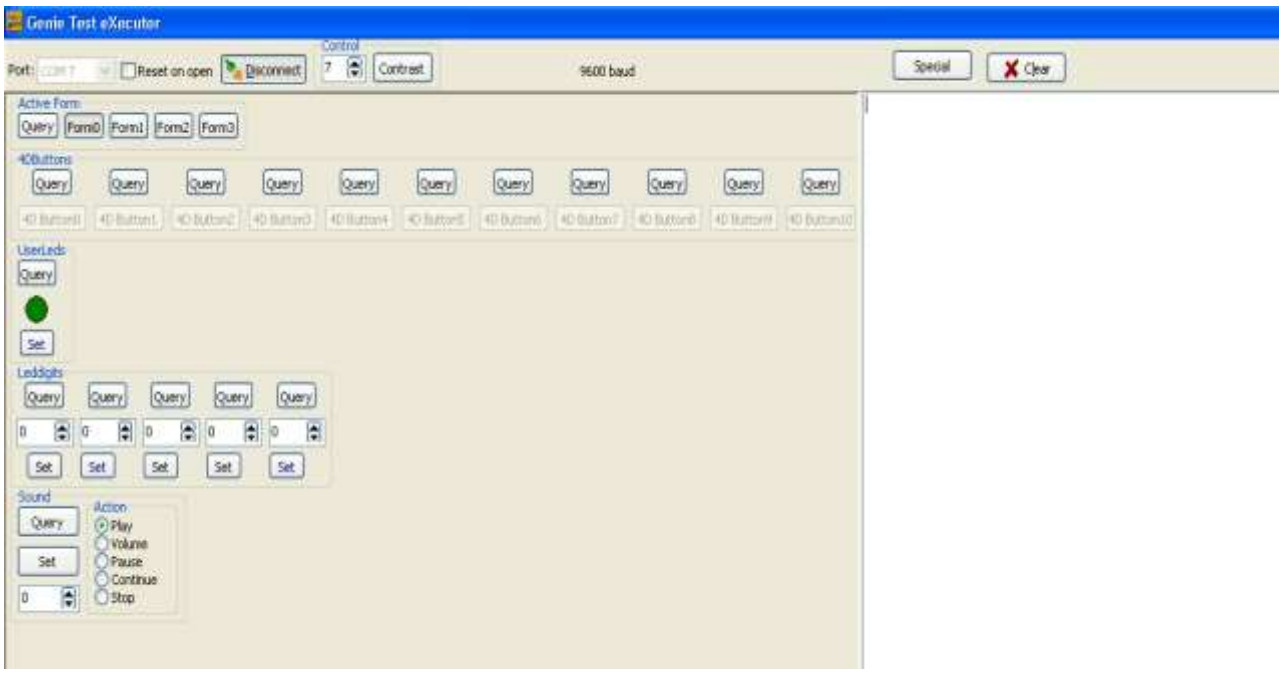
- Une fois, le programme enregistré, rebranchez la carte SSD au dos de l'écran LCD (l'écran étant encore branché a l'ordinateur) et vérifiez le fonctionnement des différents éléments ajoutés avec Visi – Genie.

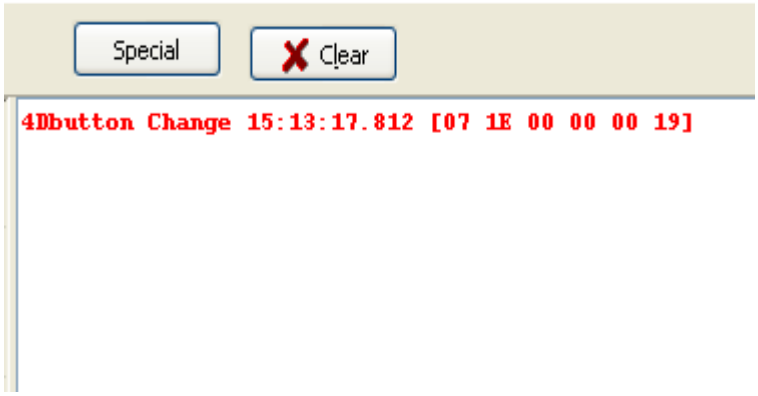
➔ 4e étape : définir les trames envoyées/reçues

Pour savoir la trame qu'on va recevoir en appuyant les boutons ou en changeant de « form »(côté microcontrôleur) et les trames qu'il faut envoyer à l'écran LCD pour qu'il génère des actions au niveau de l'interface graphique (ex : activer une LED, activer un son, changer de « form », etc…) il faut ouvrir la console GTx se trouvant dans le menu « tools ».

Avant de pouvoir utiliser cette console, il faut enregistrer l'interface graphique réalisée dans l'étape précédente et il faut la charger au niveau la carte SSD qu'on mettra au dos de l'écran LCD.

Une fois le programme chargé dans l'écran LCD, on le connecte avec le connecteur USB à l'ordinateur et on ouvre la console GTx :





En appuyant sur un des boutons (au niveau de l'écran LCD) qu'on a mis en mode « Report Message » on a les messages suivants qui s'affichent. Cela veut dire qu'en appuyant sur ce bouton on enverra une trame de la forme 07 1E 00 00 00 19 (6 octets) par la liaison Tx de l'écran LCD à la carte NUCLEO.

REMARQUES :

- Tous ces trames sont déjà définies dans l'écran LCD.

-Tous les éléments que j'ai placés dans l'écran LCD (boutons, LEDS, afficheurs, son, « forms ») utilisent des trames pour communiquer avec des systèmes extérieures, il s'agit juste de trouver les trames en utilisant la console GTx pour pouvoir établir une communication effective.

- **Voici quelques trames types pour les différents éléments graphiques de mon projet :**

Code couleur : - Partie constante, - Partie variable

A) Trame envoyée à la NUCLEO lorsqu'on appuie sur un bouton en mode « report message » :

⇨ 07 1E NB 00 00 CS (6 octets)

NB : Numéro du bouton (qu'on peut voir dans ces propriétés, voir étape 2)

Checksum : opération logique en XOR de tous les octets de la trame

Ex : pour une trame 07 1E 00 00 00 CS on a CS = 07^1E^00^00^00

B) Trame envoyée à la NUCLEO lorsqu'on change de « form » en mode « report message » :

⇨ 01 0A NF 00 00 CS (6 octets)

NF : Numéro de la « form » (ex : 00 pour form0, 01 pour form1, etc…)

C) Trame à envoyer à l'écran LCD pour allumer ou éteindre une LED présente sur son interface graphique :

⇨ 01 13 NL 00 EL CS (6 octets)

NL : Numéro de la LED

EL : Etat de la LED (00 pour l'éteindre, 01 pour l'allumer)

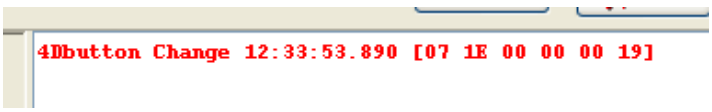
D) Trame à envoyer à l'écran LCD pour activer un son :

⇨ 01 16 NS 00 00 CS (6 octets)

NS : Numéro du son

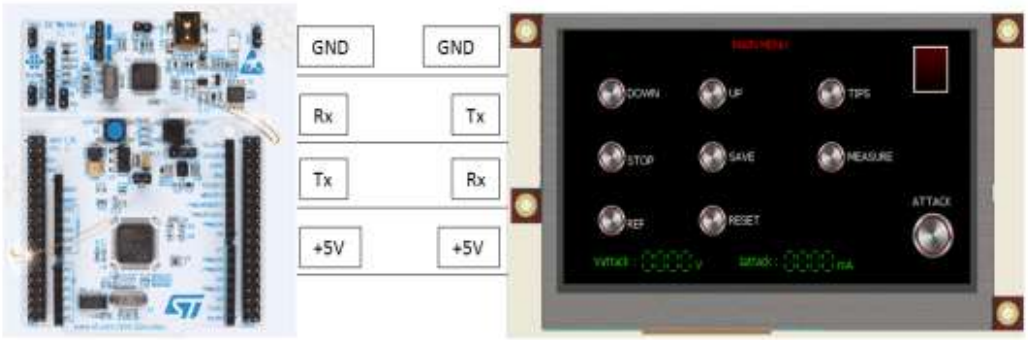
Exemple :

- On appuie sur le bouton DOWN. D'après Visi Genie, en utilisant la console GTx, on a la trame suivante qui est envoyée par la liaison Tx (côté écran LCD) :



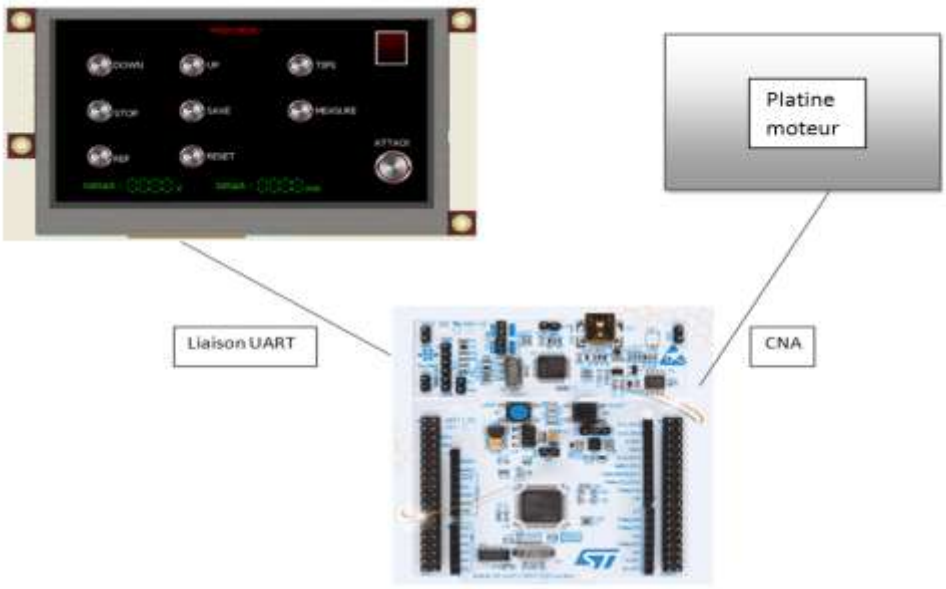
- On utilise cette trame au niveau du code en C (côte carte NUCLEO) et on indique à un convertisseur numérique – analogique qu'on a programmé (Voir 4e étape) d'envoyer une tension permettant la descente du moteur lorsqu'on recevra la trame 07 1E 00 00 00 19 par la liaison Rx (côte carte NUCLEO).

➔ 4e étape : branchement de la NUCLEO au LCD et programmation en C



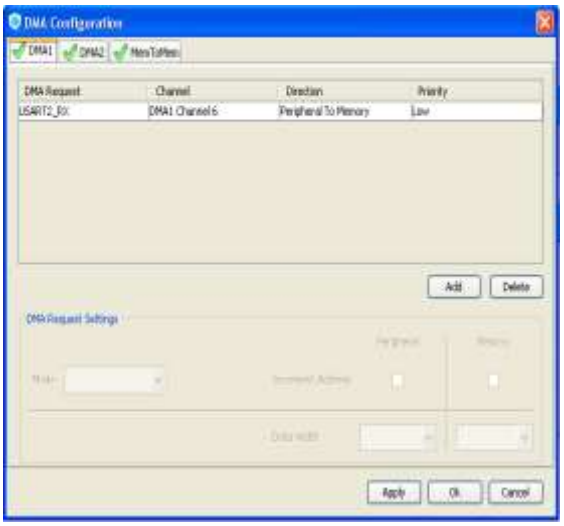
Apres avoir configuré l'interface graphique de l'écran LCD tout en notant les différentes trames qu'il va falloir utiliser, il s'agit maintenant de programmer en C au niveau du microcontrôleur de la carte NUCLEO.

Exemple d'utilisation : commande d'un moteur rotatif



Le principe de cette application est d'envoyer des commandes (boutons sur LCD) sur un moteur pas à pas, piloté par un contrôleur (TRINAMIC TMCM1140). Selon une valeur analogique (0-10V), le programme embarqué dans le TMCM1140 effectue des mouvements moteurs définis dans le code (ex : montée, descente, selon une vélocité ou une position définie, etc…). Par conséquent, le LCD va envoyer une trame selon telle action sur un bouton ; le STM32 va interpréter cette trame et convertir une valeur définie en une tension analogique envoyé sur le TMCM1140.

D'abord il faut programmer les ports qu'on veut utiliser au niveau de la carte NUCLEO (à l'aide de STM32CubeMx). Pour ne pas trop rentrer dans les détails de mon projet, pour ce cas on n'utilisera qu'un convertisseur numérique - analogique pour générer des tensions de commande (qu'on va envoyer au moteur) en fonction du bouton appuyé à l'écran LCD.



Apres avoir configuré le convertisseur numérique – analogique avec CubeMx, il faut configurer l'UART Tx et l'UART Rx. Il faut faire attention à bien configurer l'UART Rx en mode DMA circulaire qu'on va remplir avec les différentes trames reçues depuis l'écran LCD.

- **<u>Voici le squelette du code (très simplifié) :</u>**

```
#define BUFFER_SIZE 20 // On définit la taille du buffer
uint8_t UART_buffer[BUFFER_SIZE]; // On déclare le buffer pour le DMA
HAL_UART_Receive_DMA(&huart2, UART_buffer, BUFFER_SIZE); // On remplit le buffer avec les trames reçues
/*programme*/
if( trame dans le buffer = Trame du bouton DOWN)
{
HAL_DAC_SetValue(&hdac1, DAC_CHANNEL_1, DAC_ALIGN_12B_R, DOWN); //On met la valeur correspondante à la tension nécessaire pour faire descendre le moteur
        HAL_DAC_Start(&hdac1, DAC_CHANNEL_1); // On applique cette valeur au CNA
}
```

Si on veut, par exemple, changer la « form » affichée au niveau de l'écran LCD a partir du programme, il faut juste envoyé une trame a travers la liaison Tx de la carte NUCLEO.

- **Voici le squelette du code (très simplifié) :**

//d'abord on remplit un tableau avec le contenu de la trame à envoyer, pour ce cas on demande au LCD de changer vers la « form1 »

```
frame[0] = 0x01;
frame[1] = 0x0A;
frame[2] = 0x01;
frame[3] = 0x00;
frame[4] = 0x00;
frame[5] = 0x0A;
HAL_UART_Transmit(&huart2, frame, 6, 100);  // On transmet à travers Tx vers l'écran LCD
```

Remerciements à Arnauld BIGANZOLI, pour nous avoir présenté ce type d'écran et nous avoir prêté un µLCD43 pour que nous évaluions cette solution, validée par la suite. Ces échanges techniques doivent se poursuivre dans l'intérêt des projets scientifiques.