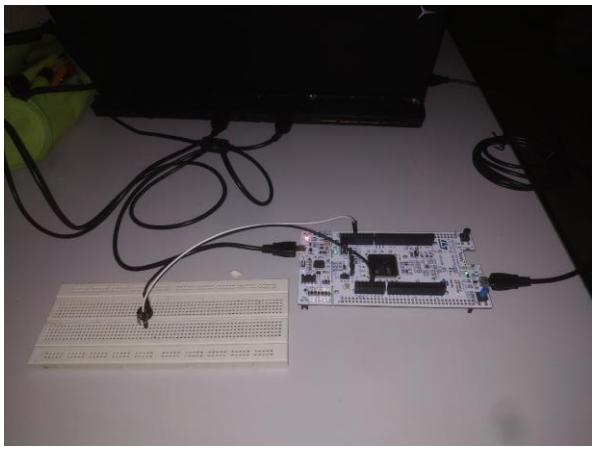
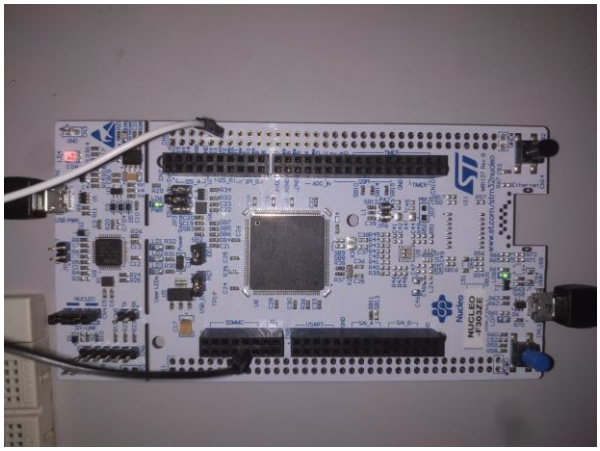
PGE IHM Réalisation des commandes de la carte Nucleo F303ZE

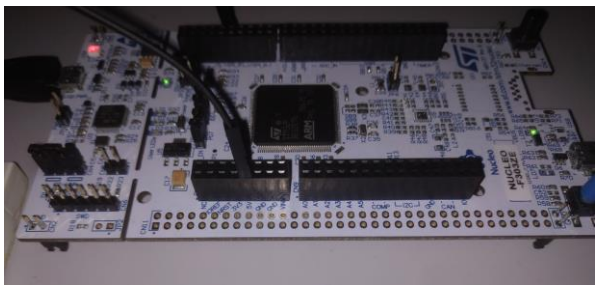
Matériels nécessaires pour piloter la carte :

-2 câbles USB 2.0 A mâle/micro USB mâle
-1 Nucleo F303ZE
-1 PC
-Une résistance de 1.5 kOhms

Branchement :







Vérification :

Ouvrez le gestionnaire de périphérique de Windows et vérifiez que votre PC reconnait la carte nucleo tel que le montre l'image ci-dessous. (Si non, essayer de changer le fil noir sur l'image précédente et mettez-le en 3V3)

Logiciels nécessaires :

-CubeMXIDE STM32CubeIDE - Integrated Development Environment for STM32 - STMicroelectronics
-Hercules Hercules SETUP utility | HW-group.com (hw-group.com)

Descriptions des fonctions :

```
void confirmationCMD(uint8_t cmd, void* str);
```

Cette fonction reçoit un chiffre, lui indiquant quelle commande a tapé l'utilisateur, s'il a tapé genFlux il recevra 1 dans cmd, void* str permet la récupération de n'importe quel type de variable dans le cas d'utilisation de la commande genFlux, il recevra le type de la structure du flux en génération (s_gen_flux), c'est-à-dire la résolution, le type de mire et le standard. Il affichera à l'utilisateur la signification de sa commande.

```
void envoiePCSTM(uint8_t* Buf, uint16_t Len);
```

Cette fonction nous permet d'envoyer des informations du STM32 au PC, il reçoit la chaine de caractère Buf et sa taille Len. Puis il utilise une fonction de la librarie « sbd_cdc_if.h » pour transmettre la chaine de caractère Buf via USB au PC. Un délai y a été ajouté pour que lors d'envoi de plusieurs données rapidement le logiciel de réception Hercules ne plante pas.

```
void s_gen_flux_config(s_gen_flux* gf);
```

Cette fonction sert à paramétrer les informations qu'on veut générer, si un utilisateur saisi genFlux dans le buffer_verif qui est le buffer de toutes les données envoyer depuis Hercules, alors on copie la chaine envoyée dans un buffer secondaire nommé test. Si les paramètres sont saisis correctement c'est-à-dire qu'après un -r ou -symbole on a bien retrouvé la valeur désirée. On continue la vérification de toutes les commandes saisies, si la valeur souhaitée n'a pas été retrouvé on renvoie erreur.

```
void s_gen_bus_config(s_gen_bus* gb);
```

Sous le même principe, cette fonction sert à paramétrer les informations qu'on veut générer, si un utilisateur saisi genBus dans le buffer_verif qui est le buffer de toutes les données envoyer depuis Hercules, alors on copie la chaine envoyée dans un buffer secondaire nommé test. Si les paramètres sont saisis correctement c'est-à-dire qu'après un -r ou -symbole on a bien retrouvé la valeur désirée.

On continue la vérification de toutes les commandes saisies, si la valeur souhaitée n'a pas été retrouvé on renvoie erreur.

`void s_rec_bus_config(s_rec_bus* rb);`
Cette fonction sert à indiquer qu'on veut recevoir, la fréquence, l'octet et le mot binaire du bus de communication. On copie la chaine de caractère du buffer_verif dans dr(une chaine de caractère d'observation), et si un utilisateur saisi recBus, cela signifie qu'il veut les informations de la structure passé en paramètre. La fonction appel confirmationCMD pour exécuter l'ordre associé à cette commande recBus, qui est l'affichage des paramètres de la structure s_rec_bus* rb.

`void s_rec_flux_config(s_rec_flux* rf);`
Cette fonction sert à indiquer qu'on veut recevoir, la résolution (largeur et hauteur), le blanking (horizontal et vertical) ainsi que le framerate du flux vidéo. On copie la chaine de caractère du buffer_verif dans dfl(une chaine de caractère d'observation), et si un utilisateur saisi recFlux, cela signifie qu'il veut les informations de la structure passé en paramètre. La fonction appel confirmationCMD pour exécuter l'ordre associé à cette commande recFlux, qui est l'affichage des paramètres de la structure s_rec_flux* rf.

`void help(void);`
Cette fonction envoie au PC la description de toutes les commandes, si dans le buffer_verif(dans Hercules), un utilisateur a tapé help.

`void clear_buffer(uint8_t * buffer_verif);`
Cette fonction affecte la valeur de base '\0' à notre buffer_verif en faisant ceci il vide le buffer_verif.

Vos variables :

Déclarer dans le « main » comme ci-dessous les structures s_rec_bus et s_rec_flux contient les valeurs à modifier.

```
s_rec_bus rb = s_rec_bus_init();  s_rec_flux rf = s_rec_flux_init();
```

Les paramètres qui changeront sont ceux des structures s_rec (rec pour réception soit les valeurs qu'on s'attend à recevoir) elles sont définies comme ceci :

```
typedef struct {
    int freq;
    int octet;
    int motbinaire;
} s_rec_bus;

typedef struct {
    int width;
    int height;
    int blankingH;
    int blankingV;
    int framerate;
} s_rec_flux;
```

Bien que vous le sachiez sûrement un petit rappel pour la modification des variables d'une structure.

```
Coordonnees monPoint;

Coordonnees *pointeur = &monPoint;
```

monPoint.x = 10; // On travaille sur une variable, on utilise le "point"

pointeur->x = 10; // On travaille sur un pointeur, on utilise la flèche

Normalement vous n'aurez qu'a modifier les pointeurs, des structures ci-dessous qu'on attend.

Comment savoir que l'on souhaite la récupération des données ?

Dans la fonction s_rec_bus_config et s_rec_bus_config il faudra ajouter votre variable vous indiquant une demande de changement des champs de s_rec_bus pour s_rec_bus_config et s_rec_flux pour s_rec_flux_config.

```
void s_rec_bus_config(s_rec_bus* rb)
{
        uint8_t dr[8];
        sscanf((char *) buffer_verif, "%s", (char *) dr);

        if(memcmp(dr, "recBus", strlen("recBus")) == 0)
        {
            confirmationCMD(RECEPTION_BUS_CMD, (void*) rb);
            //ajouter la variable vous indiquant de modifier les champs des structures attendues
        }
}

void s_rec_flux_config(s_rec_flux* rf)
{
        uint8_t dfl[8];
        sscanf((char *) buffer_verif, "%s", (char *) dfl);

        if(memcmp(dfl, "recFlux", strlen("recFlux")) == 0)
        {
            confirmationCMD(RECEPTION_FLUX_CMD, (void*) rf);
            //ajouter la variable vous indiquant de modifier les champs des structures attendues
        }
}
```

Puis lorsque vos les champs sont bien complétés le notifier par **vos_donnes_est_pretes**

if ((cmd == RECEPTION_BUS_CMD)&& (**vos_donnes_est_pretes** == 1)))

if ((cmd == RECEPTION_FLUX_CMD)&& (**vos_donnes_est_pretes** == 1)))

Les lignes 265 et 276 voir ci-dessous de la fonction confirmationCMD pourront être modifiés

```
265     if(cmd == RECEPTION_BUS_CMD) {
266         s_rec_bus* rb = (s_rec_bus*) str;
267
268         sprintf((char *) tab, "%sFrequence : %d \n", (char *) tab, rb->freq);
269         sprintf((char *) tab, "%sOctet : %d \n", (char *) tab, rb->octet);
270         sprintf((char *) tab, "%sMot binaire : %d \n", (char *) tab, rb->motbinaire);
271
272         envoiePCSTM(tab, strlen((char *) tab));
273     }
274
275     sprintf((char *) dat, "\nRECEPTION FLUX VIDEO\n");
276     if(cmd == RECEPTION_FLUX_CMD) {
277         s_rec_flux* rf = (s_rec_flux*) str;
278
279         sprintf((char *) dat, "%sResolution : %dx%d\n", (char *) dat, rf->width, rf->height);
280         sprintf((char *) dat, "%sBlanking : %dx%d \n", (char *) dat, rf->blankingH, rf->blankingV);
281         sprintf((char *) dat, "%sFramerate : %d \n", (char *) dat, rf->framerate);
282
283         envoiePCSTM(dat, strlen((char *) dat));
284     }
```