

Lyre en DMX et Flowcode

LOÏC JOSSE ^[1]

Et si on commandait une lyre à l'aide du logiciel Flowcode ? Une occasion de développer ses connaissances et compétences dans différents domaines technologiques liés à la programmation. Et de provoquer de l'intérêt chez les élèves de S-SI et STI2D à travers un objet technique festif très répandu dans les plateaux techniques de nos lycées.

Le contexte de la séquence

Dans de nombreux lycées, nous avons fait l'acquisition de lyres DMX que nous avons démontées (ou pas) afin de réaliser des séquences pédagogiques.

À l'occasion d'un séminaire de présentation de travaux pratiques (TP) pour des élèves de S-SI et de STI2D, j'ai voulu réaliser une séquence de travail sur la transmission de l'information, et notamment sur le bus DMX pilotant la lyre de spectacle. Bien sûr, on trouve des interfaces dans le commerce, mais elles limitent les actions à des champs de compétences dédiées à de simples exécutants. Mon idée était que les élèves puissent programmer une séquence de commandes des actionneurs à l'aide d'un logiciel simple à mettre en œuvre. Pour cela, j'ai choisi de travailler sous Flowcode.

Le matériel utilisé

Il ne s'agit pas ici de décrire le fonctionnement de la lyre ni de détailler le protocole DMX 512 que l'on peut facilement trouver sur la Toile. Le but est de montrer comment commander les divers actionneurs de la lyre en utilisant le logiciel Flowcode, dépourvu de composant DMX à la base.

Flowcode est sans doute le logiciel de programmation le plus utilisé en enseignement transversal, car il est facile à prendre en main et, surtout, il s'accorde parfaitement avec la partie 2.3.6 « Comportements informationnels des systèmes » du référentiel. J'ai également voulu démontrer qu'il n'est pas limité à ces composants de base (pour peu que l'on s'intéresse au code) ni aux blocs fonction.

Pour réaliser cette séquence, il faut : un PC muni du logiciel Flowcode V4, V5 ou V6, de la lyre de type Twist 25 (ou encore 150CT pour les modèles testés, mais un autre modèle devrait convenir) et une interface

mots-clés

programmation

de communication. Sur cette interface, nous allons placer le microcontrôleur qui interprètera les commandes Flowcode. J'ai choisi un ECIO28, mais on pourrait connecter une carte de programmation Matrix de type Eblocks, puisque l'interface est pourvue d'un connecteur DB9 **1 2**.

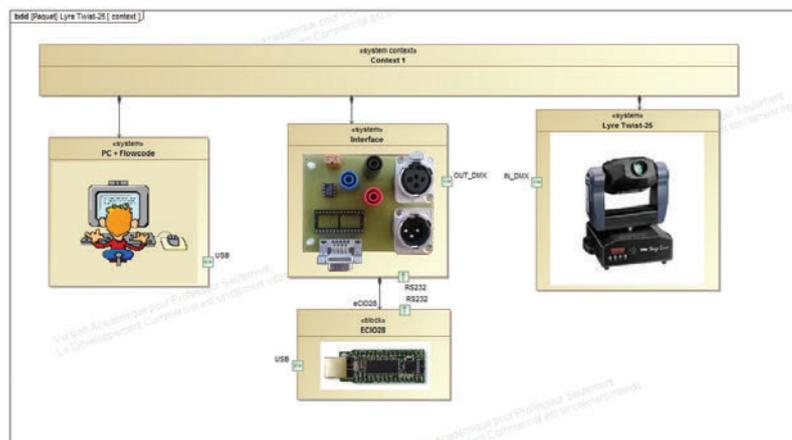
L'interface

Le schéma structurel de l'interface **3** s'articule autour du driver de ligne SN74176, qui convertit la tension du signal de transmission issu du microcontrôleur en deux tensions compatibles DMX. On peut effectuer le câblage sur une platine d'essais ou réaliser une carte à câblage imprimée téléchargeable sur le site académique, au format Proteus.

Pour la réalisation matérielle, il faut avoir conscience que la liaison électrique est de type point à point et que la distance entre deux modules est limitée à 15 m et le nombre de modules limité à 32.

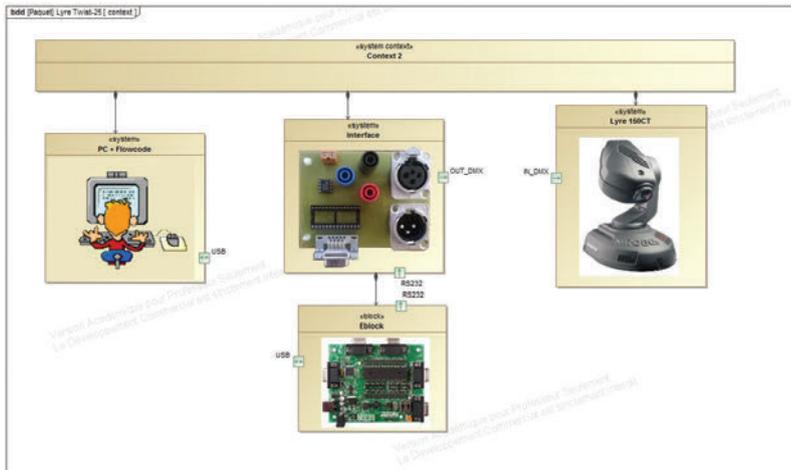
Le DMX sous Flowcode

Depuis 1986, le protocole DMX512 publié par l'USITT (United States Institute for Theater Technology) s'est imposé comme un standard dans la commande des systèmes d'éclairage de scène. Rappelons qu'il utilise une liaison série asynchrone cadencée à 250 kb/s⁻¹ avec 1 bit de start, 8 bits de données et 2 bits de stop. Cette transmission est de type « simplex » ; sa fiabilité réside dans la répétition des trames transmises, car aucun contrôle à la réception n'est effectué. Sous Flowcode, nous pouvons aisément réaliser une liaison



1 Diagramme SysML de définition des blocs lié à l'activité solution avec un ECIO28

[1] Professeur, enseignant SII au lycée Louis-Le-Grand de Paris (75005).



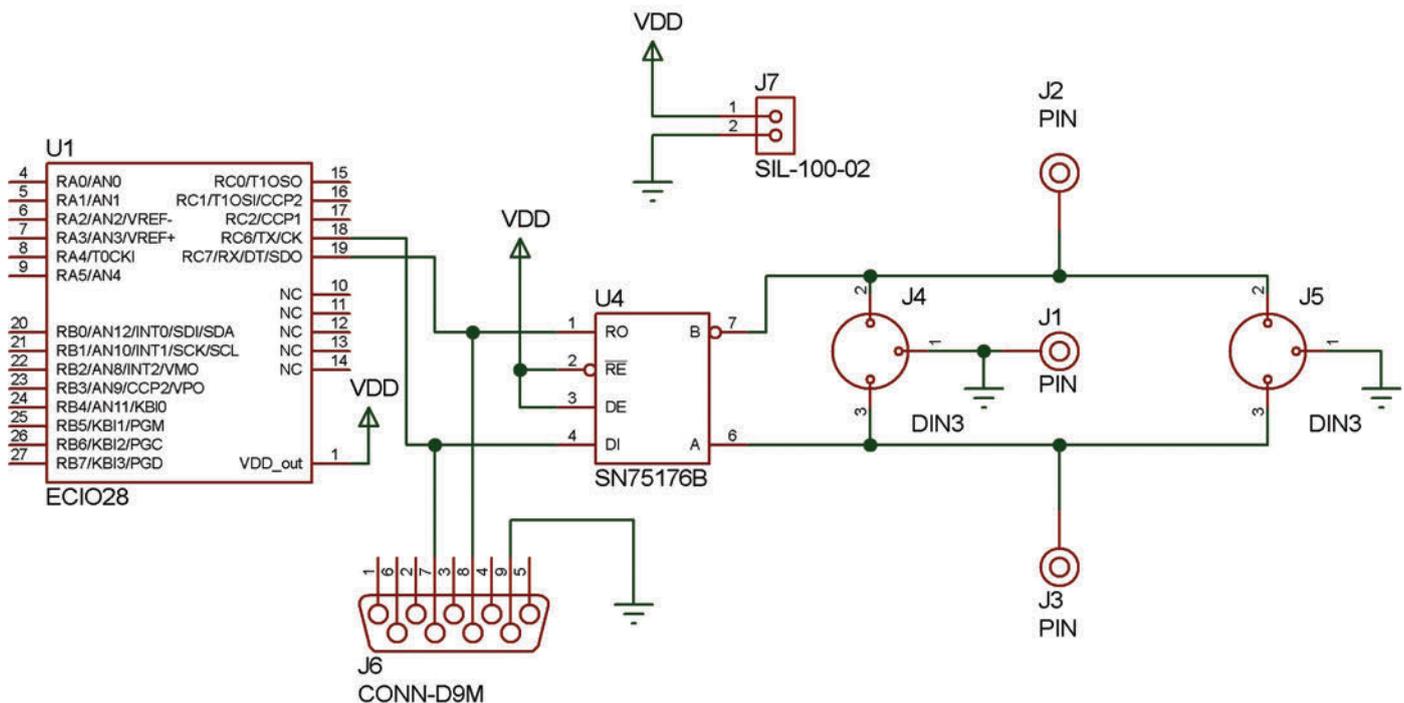
2 Diagramme SysML de définition des blocs lié à l'activité solution avec une carte Matrix de type Eblocks

série 4 à cette vitesse avec 1 bit de start, 8 bits de données et 1 bit de stop (pas deux !).

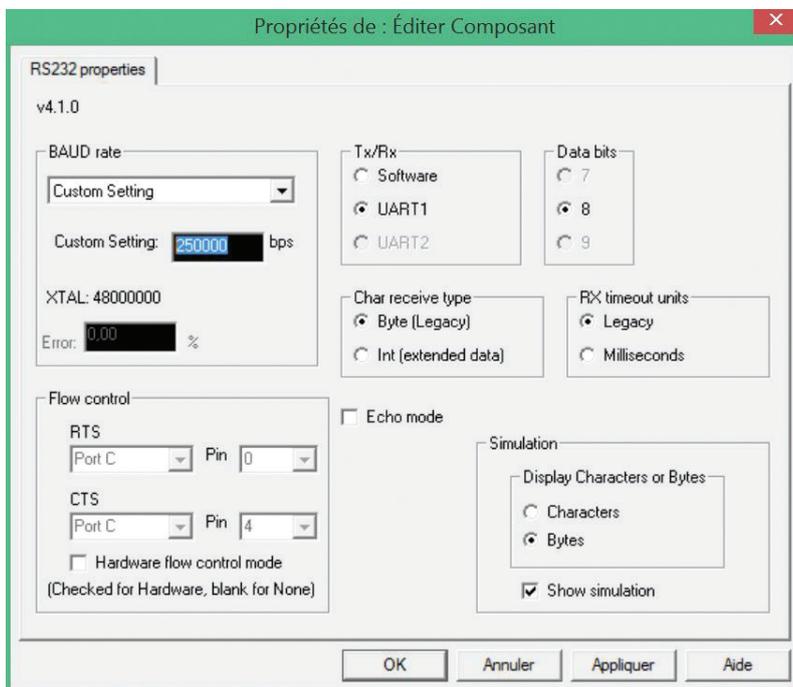
Chaque trame DMX 5 commence par un break (état logique « 0 ») de durée comprise entre 88 µs et 1 s, suivi d'un *mark after break* (état logique « 1 ») de 8 à 1 s, du *start code* (valeur 00h), puis des différents canaux (jusqu'à 512). La trame doit être régulièrement rafraîchie, avec un intervalle entre deux trames (état logique « 1 ») compris entre 0 et 1 s.

Les signaux *break* et *mark after break* permettent aux récepteurs de se synchroniser. Le *start code* est prévu par l'USITT pour d'éventuelles extensions ; la valeur standard – et donc utilisée dans la plupart des récepteurs – est 0x00. Les autres canaux sont fonction du ou des récepteurs utilisés.

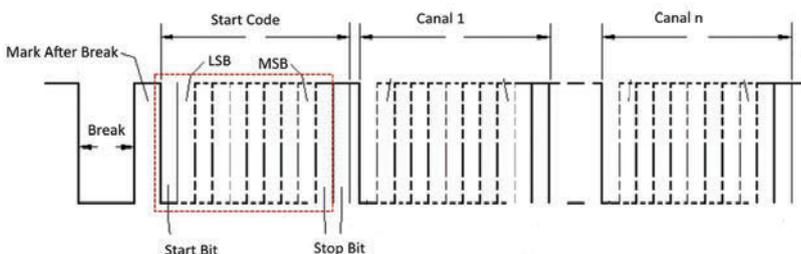
Pour programmer cette partie sous Flowcode, nous avons besoin de créer une petite routine en langage C 6. En effet, nous ne pouvons pas directement contrôler une sortie du microcontrôleur PIC si celle-ci est déjà utilisée par une macro composant. Il n'est alors pas possible de rajouter le *break* et le



3 Schéma structurel de l'interface de communication



4 Configuration de la liaison série sous Flowcode



5 Trame DMX

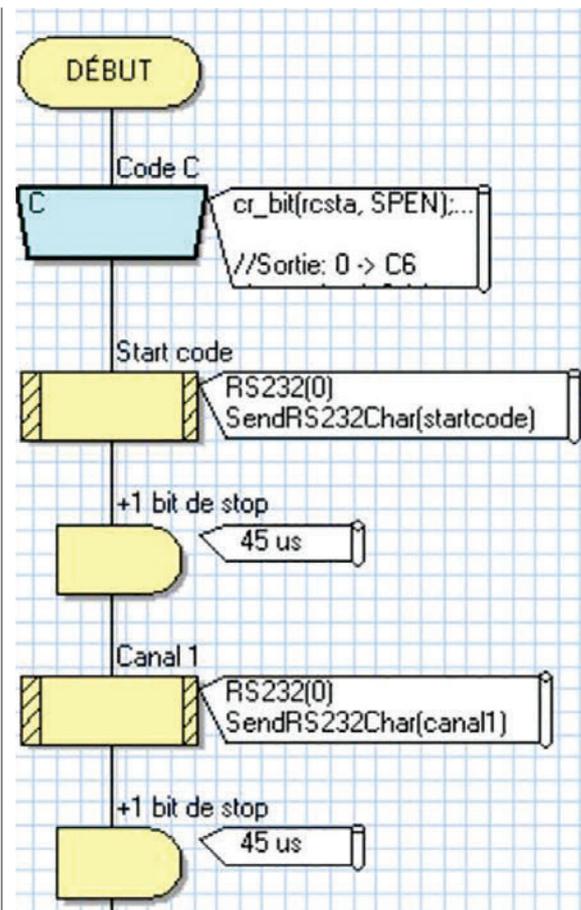
mark after break, ou encore le deuxième bit de stop à la routine RS232.

La solution consiste à interrompre le fonctionnement de la liaison série pour mettre en place le début de chaque trame. Pour désactiver la liaison série, il faut mettre à « 0 » le bit SPEN (Serial Port Enable) du registre RCSTA (Receive Status and Control Register). On valide la liaison série en mettant à « 1 » ce même bit. Ces registres sont identiques pour les différents

```

cr_bit (rcsta, SPEN);           // Stop interface serie
//Sortie : 0 -> C6 (Break)
trisc = trisc & 0xbf;
portc = portc & 0xbf;
delay_us (200);
//Sortie : 1 -> C6 (Mark After Break)
portc = (portc & 0xbf) | 0x40;
delay_us (30);
st_bit (rcsta, SPEN);         // marche interface serie
    
```

6 Contenu de la routine en C



7 Création de la macro nommée « sortie_dmx »

microcontrôleurs PIC utilisés, ce qui n'oblige pas à modifier le code correspondant pour chacun.

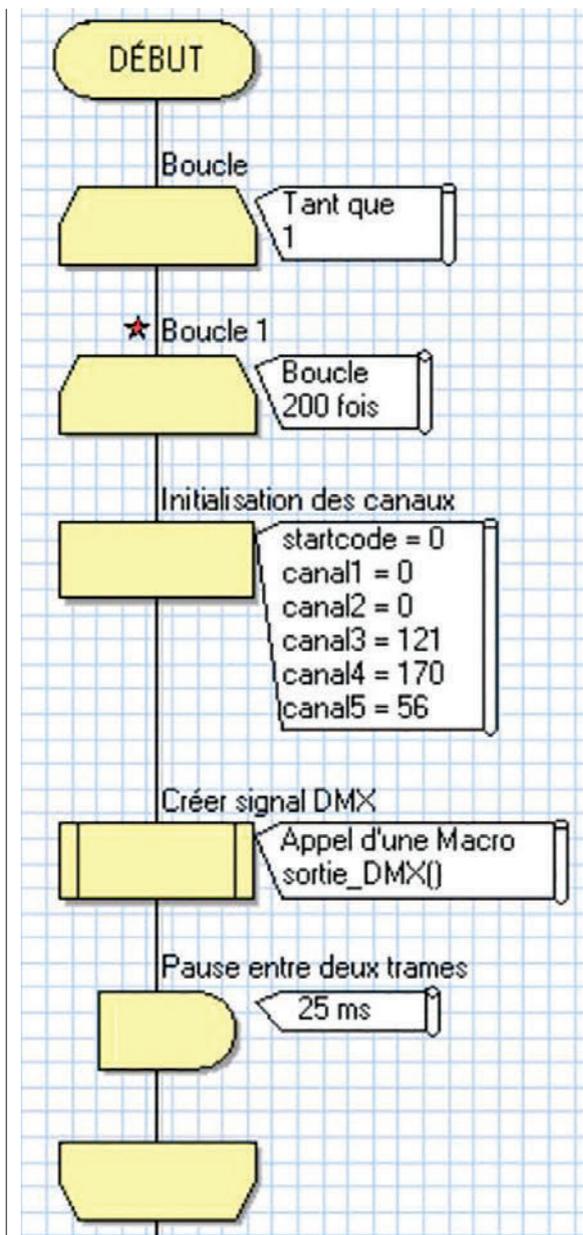
Commençons par créer une macro nommée « sortie_dmx » par exemple. Plaçons-y successivement : la routine C pour créer le début de la trame, l'envoi du start code, une temporisation afin de rajouter le bit de stop manquant, et enfin les différents canaux. Avec une lyre Twist 25, cette macro prend l'allure donnée sur la figure 7.

Le programme principal est quant à lui assez simple, il suffit d'initialiser les divers canaux, puis de les envoyer dans une boucle 8.

La lyre Twist 25

Les valeurs que peuvent prendre les différents canaux de la lyre Twist 25 sont indiquées dans le tableau 9. L'allure des gobos (nom générique donné aux figures lumineuses projetables) est représentée sur la figure 10.

Pour ceux qui préfèrent programmer avec l'environnement Arduino, en spécialité SIN (Système d'information et numérique) par exemple, les éléments nécessaires sont fournis dans l'encadré (page 64).



8 Programme principal

Exploitation pédagogique

Le contrôle de la lumière et des éclairages fait partie des techniques essentielles de la représentation et de la mise en scène d'espaces physiques et d'espaces-temps racontant une histoire ou mettant en valeur une pratique artistique sur scène. Nous proposons une séquence de travaux pratiques à destination des élèves de S-SI ou de STI2D dans le cadre des activités de tronc commun. Cette séquence, d'une durée de 2 heures, se découpe en trois parties : étude du bus DMX, analyse du document constructeur de la lyre et étude d'un cas pratique. Elle s'inscrit dans les référentiels de la façon suivante :

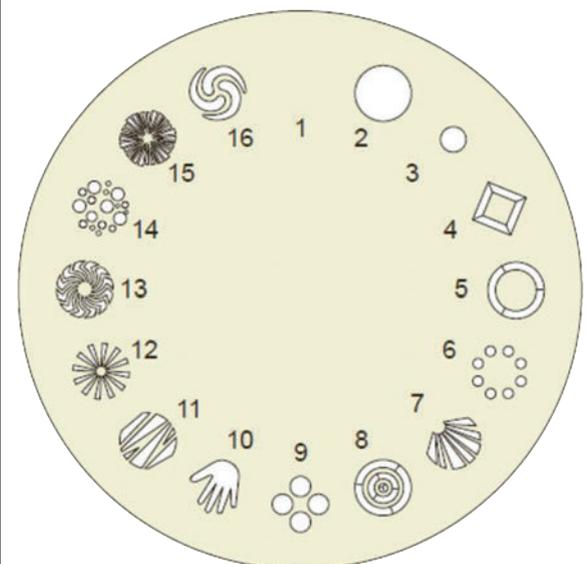
– S-SI : A2. Analyser le système (Notion de trame, liaisons série) ;

Canal 1 (effets)	Canal 2 (gobos)	Canal 3 (couleurs)	Canal 4 (pan)	Canal 5 (tilt)
0 – 15 : aucun changement, aucun effet 16 – 91 : changement continu de deux modèles (de lent à rapide) 92 – 135 : changement continu de deux couleurs (de lent à rapide) 136 – 195 : changement continu de deux couleurs et deux modèles (de lent à rapide) 196 – 255 : effet tremblement (de lent à rapide) ; l'effet n'est actif que si, pour le deuxième canal, une valeur DMX est réglée entre 8 et 127	0 – 7 : pas de lumière (blackout) 8 – 15 : Gobo n° 2 16 – 23 : Gobo n° 3 24 – 31 : Gobo n° 4 32 – 39 : Gobo n° 5 40 – 47 : Gobo n° 6 48 – 55 : Gobo n° 7 56 – 63 : Gobo n° 8 64 – 71 : Gobo n° 9 72 – 79 : Gobo n° 10 80 – 87 : Gobo n° 11 88 – 95 : Gobo n° 12 96 – 103 : Gobo n° 13 104 – 111 : Gobo n° 14 112 – 119 : Gobo n° 15 120 – 127 : Gobo n° 16 128 – 255 : changement continu des gobos (de lent à rapide)	0 – 10 : blanc 11 – 21 : vert 22 – 32 : orange 33 – 43 : bleu clair 44 – 54 : ambre 55 – 65 : rouge 66 – 76 : violet 77 – 87 : rose 88 – 98 : vert clair 99 – 109 : bleu 110 – 120 : jaune 121 – 127 : magenta 128 – 255 : changement continu des couleurs (de lent à rapide)	0 – 255 pour une rotation de 0° à 540°	0 – 255 pour une rotation de 0° à 270°

9 Valeurs que peuvent prendre les différents canaux pour la lyre Twist 25

– STI2D : 04 – Décoder l'organisation fonctionnelle, structurelle et logicielle d'un système (2.3.6 « Comportements informationnels des systèmes »).

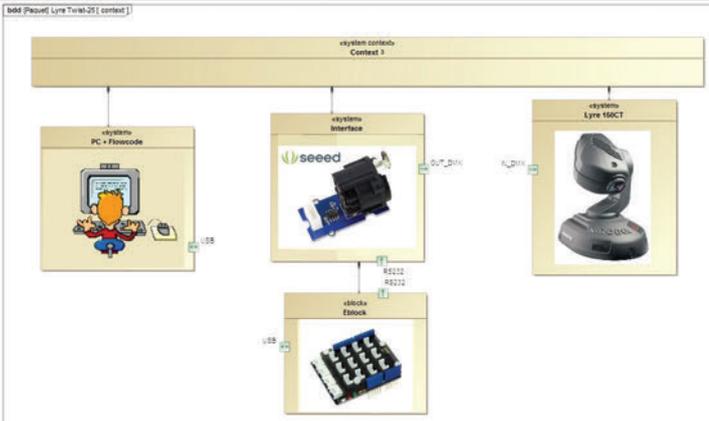
Commençons par analyser le mode de transmission. Nous rappelons aux élèves que le DMX est un protocole 8 bits ; 3 bits supplémentaires sont ajoutés à chacun des mots de commande. Dans la trame du signal, le paquet de données est réactualisé toutes les 22,572 ms. Des pauses sont ménagées en début de signal pour annoncer le début d'une nouvelle trame.



10 Allure des gobos

Programmation avec l'environnement Arduino

Je vous propose l'architecture suivante :

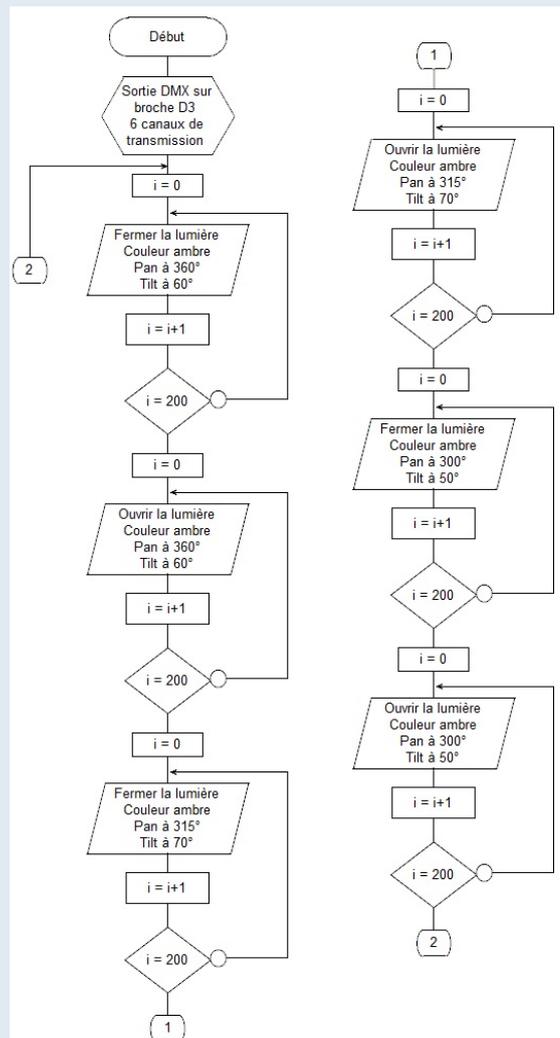


Il suffit d'associer une carte Arduino Uno avec un module d'interface Grove et un module Grove DMX512. Aucun câblage à effectuer, il faut juste assembler ces éléments et connecter la carte Arduino au PC. L'installation d'une bibliothèque DmxSimple dans l'environnement du logiciel facilitera la tâche, puisqu'il suffira d'indiquer dans notre programme le canal à commander et la donnée à transmettre. Pour reprendre l'exercice de programmation, nous donnons l'algorithme sur la figure 11 ainsi que le programme suivant :

Solution

```
#include <DmxSimple.h>
void setup ()
{
  DmxSimple. usePin (3);
  DmxSimple. maxChannel (6);
}
void loop ()
{
  int i;
  for (i=0;i<200;i++)
  {
    DmxSimple. write (2,0);
    DmxSimple. write (3,44);
    DmxSimple. write (4,170);
    DmxSimple. write (5,56);
    delay (25);
  }
  for (i=0;i<200;i++)
  {
    DmxSimple. write (2,8);
    DmxSimple. write (3,44);
    DmxSimple. write (4,170);
    DmxSimple. write (5,56);
    delay (25);
  }
  for (i=0;i<200;i++)
  {
    DmxSimple. write (2,0);
    DmxSimple. write (3,44);
    DmxSimple. write (4,148);
    DmxSimple. write (5,66);
    delay (25);
  }
  for (i=0;i<200;i++)
  {
    DmxSimple. write (2,8);
    DmxSimple. write (3,44);
    DmxSimple. write (4,148);
  }
}
```

```
DmxSimple. write (5,66);
delay (25);
}
for (i=0;i<200;i++)
{
  DmxSimple. write (2,0);
  DmxSimple. write (3,44);
  DmxSimple. write (4,141);
  DmxSimple. write (5,47);
  delay (25);
}
for (i=0;i<200;i++)
{
  DmxSimple. write (2,8);
  DmxSimple. write (3,44);
  DmxSimple. write (4,141);
  DmxSimple. write (5,47);
  delay (25);
}
}
```



11 Algorithme programmation avec l'environnement Arduino

En résumé :

- taux de rafraîchissement de 44,3 Hz ;
- (512 +1) canaux ;
- (8 + 3) bits par canal.

Les élèves peuvent alors calculer la vitesse de transmission (en bits/s ou baud) des informations sur le bus :

$$11 \times 513 \times 44.3 = 250 \text{Kb/s}$$

Afin qu'ils puissent visualiser ces informations, nous leur fournissons un fichier Flowcode « Lyre_dmx_eleve.fcf » dans lequel les canaux ont des valeurs fixes et sont envoyés en continu, par exemple : startcode = 0, canal1 = 0, canal2 = 0, canal3 = 0, canal4 = 42 et canal5 = 80. Après avoir ouvert ce fichier et découvert sa substantifique moelle, ils devront le compiler en hexadécimal afin de l'exécuter sur la carte d'interface présentée ci-dessus ou dans l'environnement Proteus en simulant le fichier « simuDMX.pdsprj ». Ce travail de manipulation et de simulation peut aussi être effectué parallèlement par deux élèves si l'on désire faire une analyse comparative.

Les élèves relèvent les signaux D+ et D- à l'oscilloscope (calibre 100 us et 2v par division). Puis ils mesureront la durée des signaux Break, MAB, d'un bit et entre deux trames :

$$t_{\text{Break}} = 200 \mu\text{s} ; t_{\text{M.A.B.}} = 35 \mu\text{s} ; t_{\text{Bit}} = 4 \mu\text{s} ; \text{durée entre 2 trames} = 15,8 \text{ ms}$$

Avec les réglages proposés, nous obtenons les oscillogrammes sur lesquels un ajustement de la base de temps permet de mesurer aisément la durée d'un bit.

Un troisième élève chargé de rechercher sur Internet ou sur le document relatif à la norme « Normedmx512_r3.pdf » les durées préconisées pour ces signaux peut intervenir afin de justifier les valeurs mesurées, avec ses camarades.

On demandera de repérer, sur le document constructeur « Lyre_TWIST25fr.pdf », l'actionneur commandé par chaque canal :

5 canaux et le start code sont transmis dans une trame : canal1 = shutter, canal2 = gobos, canal3 = couleurs, canal4 = pan et canal5 = tilt.

Il leur faudra également indiquer les plages de déplacement en degré des moteurs Pan et Tilt. Et d'en déduire le pas de déplacement en degré des moteurs Pan et Tilt, sachant que chaque canal est codé sur 8 bits.

$$\text{Pan} : 540^\circ \rightarrow \text{pas} = 540/255 = 2.11^\circ ;$$

$$\text{Tilt} : 270^\circ \rightarrow \text{pas} = 270/255 = 1.05^\circ$$

À travers cette exploitation pédagogique, les élèves découvrent de façon analytique et expérimentale la transmission de niveau 1 (modèle OSI) d'une liaison DMX. Il est important qu'ils puissent visualiser la manière dont transitent les informations sur la couche physique.

Étude de cas

Pour les besoins d'une exposition, la lyre doit éclairer successivement trois objets pendant 3 secondes. L'éclairage doit être de couleur ambre et l'objet N° 1

situé suivant les coordonnées 360° Pan et 60° Tilt ; l'objet N° 2 à 315° Pan et 70° Tilt ; l'objet N° 3 à 300° Pan et 50° Tilt. Le faisceau lumineux est de forme ronde diamètre maximum. On obscurcit le faisceau lumineux entre deux objets le temps de se repositionner (on estime cette durée à 3 secondes également).

Pour éclairer un objet, il faut alors définir les valeurs des canaux, envoyer la trame DMX, attendre 25 ms, puis recommencer 200 fois pour atteindre les 3 secondes demandées .

Ensuite, on obscurcit le faisceau et on donne les nouvelles coordonnées en Pan et Tilt pour atteindre le prochain objet, toujours dans une boucle 200 fois. Il y aura donc 6 boucles dans une boucle sans fin.

On recherchera les valeurs à envoyer, ainsi que le canal à utiliser pour allumer et éteindre le faisceau de la lyre, puis pour obtenir le filtre de couleur ambre :

Pour éteindre le faisceau : canal2 = 0

Pour l'allumer : canal2 = 8

Couleur ambre : canal3 = 44

On traduira les différentes coordonnées des objets afin d'obtenir les valeurs décimales à envoyer sur Pan et Tilt.

Objet 1 : 360° Pan → 170 60° Tilt → 56

Objet 1 : 315° Pan → 148 70° Tilt → 66

Objet 1 : 300° Pan → 141 50° Tilt → 47

Les élèves pourront ensuite réaliser le programme en complétant le fichier Flowcode donné précédemment. Une fois le programme compilé et envoyé vers l'interface, ils constateront le bon fonctionnement.

Les valeurs des canaux pour les six boucles sont les suivantes :

Boucle 1 : startcode = 0 canal1 = 0 canal2 = 0

canal3 = 121 canal4 = 170 canal5 = 56

Boucle 2 : startcode = 0 canal1 = 0 canal2 = 8

canal3 = 121 canal4 = 170 canal5 = 56

Boucle 3 : startcode = 0 canal1 = 0 canal2 = 0

canal3 = 121 canal4 = 148 canal5 = 66

Boucle 4 : startcode = 0 canal1 = 0 canal2 = 8

canal3 = 121 canal4 = 148 canal5 = 66

Boucle 5 : startcode = 0 canal1 = 0 canal2 = 0

canal3 = 121 canal4 = 141 canal5 = 47

Boucle 6 : startcode = 0 canal1 = 0 canal2 = 8

canal3 = 121 canal4 = 141 canal5 = 47

Conclusion

Cette séquence intervient après le cours sur les réseaux et les bus de terrain en cycle terminal. Nous la réalisons en parallèle avec l'étude du bus CAN effectuée sur le pilote automatique de bateau. On a constaté qu'elle avait un impact positif sur les élèves, en particulier ceux de SIN qui ont voulu poursuivre l'expérience en réalisant une commande de la lyre par Bluetooth et par IHM (Interface homme-machine) sur PC.

Il est possible de récupérer le fichier source Flowcode « Lyre_dmx_profECIOv4.fcf » sur le site académique de Paris. Bonne programmation ! ■