18 juillet 2016

ESIEE ENGINEERING

Rapport de stage :

UNLV

Driving simulator UNLV

Vanhaesebroucke Geoffrey

I – Contexte du projet

L'University of Nevada Las Vegas (UNLV) a en sa possession depuis 7 années maintenant un simulateur de conduite. L'environnement 3D est réalisé avec le logiciel de simulation Unity.

Le but de mon projet était de réussir à activer la motion du simulateur en fonction de l'accélération de la voiture et de l'inclinaison de la route. Tout ceci avec le logiciel Unity.

Le simulateur est devenu compatible avec Unity depuis l'année dernière seulement. Le constructeur « Simcraft » a donc développé un SDK pour pouvoir faire fonctionner la « motion » avec Unity. Cependant, après avoir pris contact avec eux, ils nous ont proposé de nous aider en souscrivant à leur support de deux ans pour la maudite somme de 10 000$, ce qui était impensable pour notre superviseur. Il a donc fallu développer notre propre plugin.

II – Activation de la motion

A – Refferences des Hardwares

La structure du simulateur vient de la société « Simcraft » et le modèle du simulateur est de la gamme APEX 3.

Lien : http://www.simcraft.com/apex-sc830.aspx

La structure du simulateur est articulée sur 3 axes qui correspondent à 3 pistons (Actuators).

Les « actuators » sont pilotés par une « motion box » contenant des serveaux moteurs et des amplificateurs. Ceux-ci sont sous traités chez la société « Diysim ».

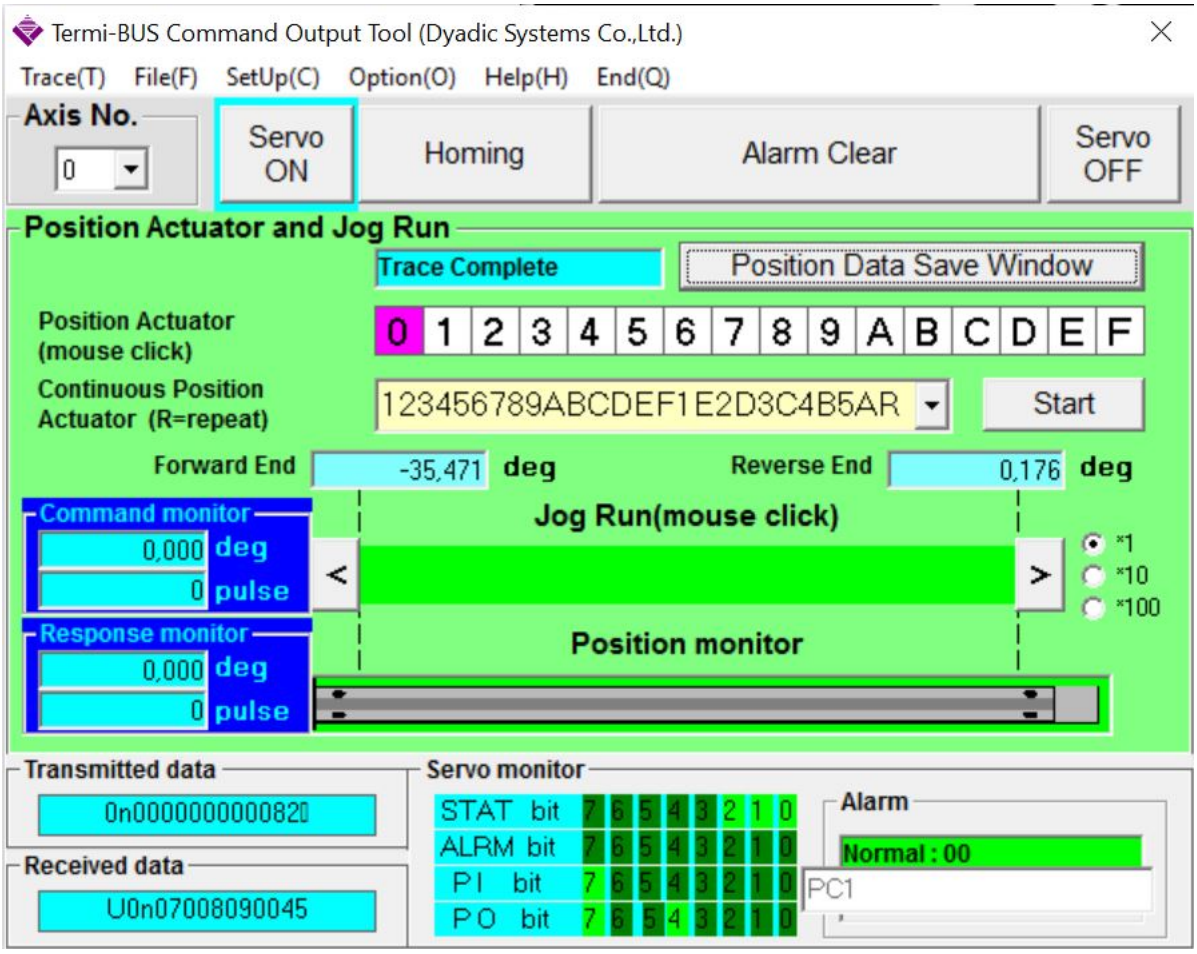
Lien : http://diysim.com/product-category/motion-systems/

Plus précisément, les « actuators » sont les modèles SCN6_040.

Les serveaux moteurs des « actuators » peuvent être pilotés en leur envoyant des trames en « string » par le biais d'une liaison série RS232. Sur le boîtier de motion du simulateur, il y a un convertisseur USB vers RS232 pour le piloter depuis un PC. Pour activer la liaison série sur votre PC, il est nécessaire d'installer certains drivers disponibles de le dossier « driver_serial_port » du .zip.

B – Le logiciel de pilotage TBVST

Ce logiciel de chez « Dyadic » est une interface permettant de piloter le simulateur sur ses trois axes 0, 1 et 2. Concrètement avec ce logiciel il est possible d'initialiser la connexion avec chacun des trois axes (Homing) et d'enregistrer jusqu'à 15 positions par axe pour ensuite jouer des séquences.



Lien de téléchargement :

http://www.miraiintertech.com/pctool452.zip

Ce logiciel m'a aidé à comprendre l'encodage des données envoyer aux serveaux moteurs. Les tests que j'ai effectués consistaient à cliquer sur des positions tout en écoutant ce qui passait par le port série pour identifier quelle commande correspondait à quelle trame en « string ».

Pour écouter le port série j'utilisais le logiciel « Serial port monitor » de chez Eltima (cependant la version de démo n'est disponible que pendant 14 jours).

Lien de télechargement :

http://www.eltima.com/fr/products/serial-port-monitor/

Ce logiciel ne peut pas être dans la solution définitive du projet, puisque le but est de pouvoir envoyer des trames par le biais de Unity directement.

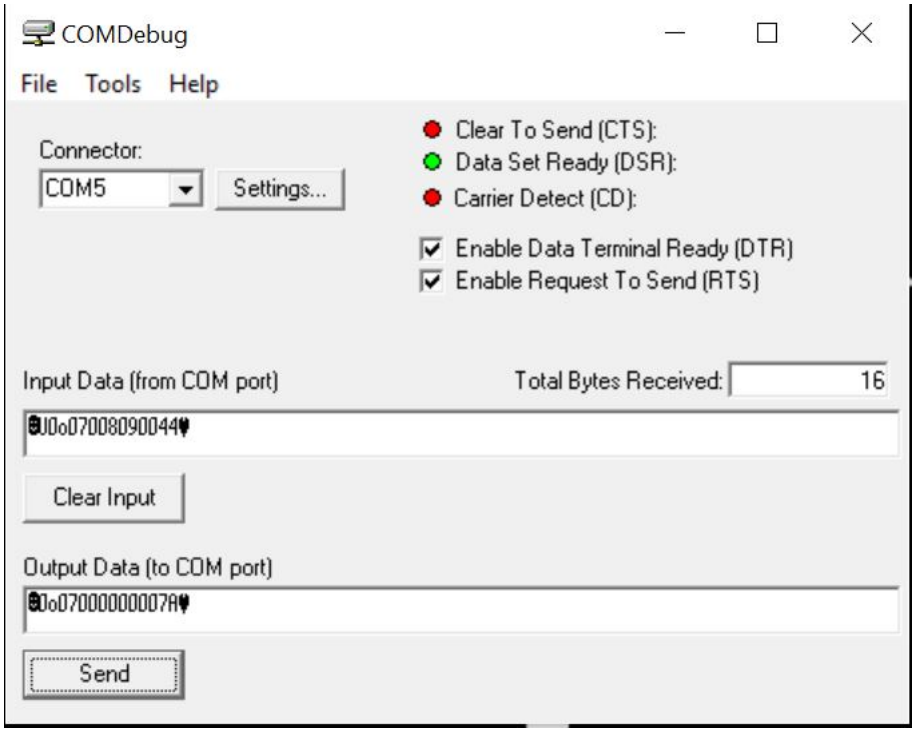
C – Encodage des trâmes

Pour comprendre l'encodage des trames à envoyer sur la « motion box », j'ai pris contact avec le support de la société « Mirai inter-tech » qui propose un kit avec le logiciel TBVST.

Lien : https://miraiintertech.com/e-store/products/PC%252dCTC-Tool-Kit.html

J'ai donc pris contact avec Sid (« sid@miraiintertech.com ») qui m'a transmit un dossier .zip contenant des datasheets qui expliquent cet encodage (page 47 du fichier « EE06426I.pdf »).

Pour envoyer des commandes, il m'a conseillé d'utiliser le logiciel « COM Debug ».



Ici un « homing » sur l'axe 0

Lien de téléchargement : http://www.taltech.com/freesoftware/COMDebugSetup.htm

Les commandes les plus utilisées sont les commandes « o » et « a » (voir STRING CALCULATOR.xls)

Les commandes en « o » dites « Homing » permettent d'initialiser les axes. En voici un exemple : 0o07000000007A

0 correspond à l'axe 0.

O correspond à la commande « homing ».

07 correspond à « Homing pattern selection code » (page 49 de EE06426I.pdf).

00000000 par défaut.

7A correspond au BCC (« checksum », il doit être correct pour que la trame soit prise en compte.

Calcul du BCC (page 3 de Mech_Ascii_Setup)

Pour ma part, j'utilisais le calculateur de ce site : http://www.scadacore.com/field-applications/programming-calculators/online-checksum-calculator/

Prendre celui-ci : 0x100 - Sum Of Bytes

Ex : Pour la trâme ci-dessus, il faut rentrer « 0o0700000000 » dans ASCII Input.

STX et ETX

Il faut également savoir que chaque trame doit être englobée entre le caractère STX et le caractère ETX. Ce sont des caractères spéciaux en ASCII, mais ils valent respectivement 02 et 03 en hexa.

Ex : « 0o0700000000 » en ascii = « 306F30373030303030303030 » en hexa

Il faut donc écrire 02306F30373030303030303030303 dans Hex Input pour récupérer les caractères spéciaux de la string.

Les commandes en « a » elles permettent d'atteindre une position (un angle), en voici un exemple : 0aFFFFD8F00005

0 correspond à l'axe 0.

a correspond à la commande de changement de position.

FFFFD8F0 les 8 digits de la « target position ».

00 par défaut (place holder).

05 correspond au BCC (« checksum », il doit être correct pour que la trame soit prise en compte.

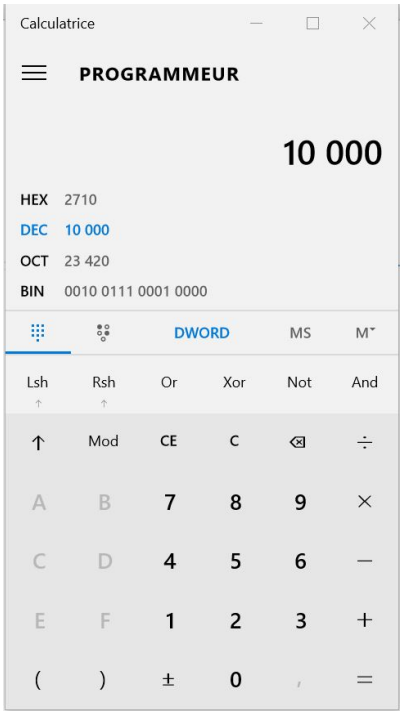
Calcul des 8 digits « target position »

Le simulateur peut bouger de 0° à 35° sur chaque axe. Cette plage d'inclinaison est constituée d'environ 20 000 pulses.

Ex : 17,5° correspond à 10 000 pulses.

C'est ce nombre de pulses qu'il faut traduire en hexa et qui constitue la « target position » des 8 digits vue précédemment.

10 000 pulses = 2710



Note : Quelle que soit la commande, ne pas oublier de calculer le BCC et ensuite d'ajouter les STX et ETX en début et fin de trame.

Ici : .0aFFFF2710002D.

Autres perspectives :

Il existe des dizaines d'autres commande possible, il serait intéressant de regarder du côté de la commande « v » qui permet de changer l'accélération et la vitesse du simulateur.

III – Unity, logiciel de simulation 3D

A – Lien avec Unity

Après avoir compris le fonctionnement des trames, l'étape suivante a été d'automatiser les calculs du BCC et des « target position » pour former nos propres trames en ayant seulement un angle comme entrée. Cette étape a été réalisée en insérant des scripts dans le logiciel Unity.

Le calcul a été automatisé avec des scripts sur Unity :

```
using UnityEngine;
using System.Collections;
using System.Text;

public class ActuatorData : MonoBehaviour {

    public bool usePulses = false;
    public bool initAxes = true;

    public enum Axe{X = 0, Y = 1, Z = 2};
    public Axe[] axe_raw = new Axe[] {Axe.X, Axe.Y, Axe.Z};

    public string[] initialPositionsAxes = new string[] {""+STX+"0o07000000007A"+ETX,""+STX+"1o070000000079"+ETX,""+STX+"2o070000000078"+ETX};
    public Axe _Axe = Axe.X;
    public byte _axe = 0;
    public enum Command{a,o};
    public Command _Command = Command.a;

    const char STX = '\u0002';
    const char ETX = '\u0003';

    public string _command = "";
    public float DegreePosition = 15;
    public float PulsesRev = 800;
    public float DegreeByPulse = 0.45f;
    public string TargetPosition = "FFFFFFFF";
    public byte PlaceHolder = 0;
    public byte BCC = 0;
```

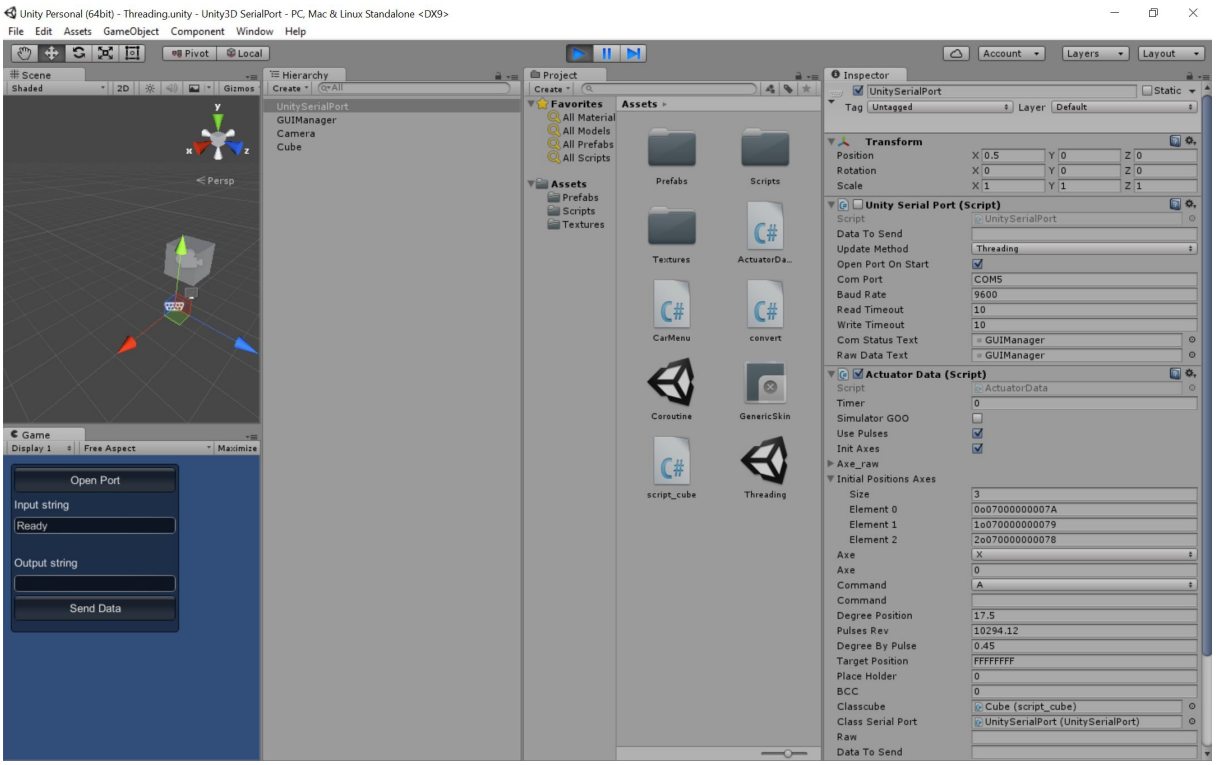
Partie du script d'encapsulation des trames

Ensuite, nous envoyons les trames directement sur le port série de la « Motion box » à l'aide du script « UnitySerialPort.cs ».

De cette façon nous avons réussi à faire bouger le simulateur par le biais de Unity sur l'axe 0, en revanche, pour des raisons que l'on ignore encore, le déplacement sur les axes 1 et 2 reste impossible. En effet, après avoir parlé avec le support de chez Mirai qui confirme que les commandes envoyées sont les bonnes, la solution reste introuvable.

Amélioration du code :

Le code à ensuite été amélioré pour le rendre plus facile d'utilisation et également pour l'automatiser au démarrage du logiciel Unity.
J'ai créé, avec l'aide du chef de projet, un nouveau script (script_cube) attaché à un nouveau « game object », un cube, qui représente la future voiture. L'idée ensuite a été de récupérer l'angle de ce cube sur son axe horizontal par rapport à la normale. Cet angle est ensuite traduit en trame et envoyé sur le simulateur par le biais du script vu précédemment « ActuatorData.cs ».
Pour les tests de ce script, l'inclinaison du cube est simulée en appuyant sur les flèches du clavier.



Interface Unity du plugin

Sur cette version, lorsque l'on fait un « play », un appui sur « espace » lance une initialisation sur l'axe 0, le simulateur reçoit donc un « homing » et s'arrête sur la position 0°.
Il faut ensuite cocher la case « Simulateur GOO » pour pouvoir faire bouger le simulateur avec les flèches du clavier.

B – Travail restant à réaliser

- Réussir à activer la motion sur les axes 1 et 2.
- Optimiser l'automatisation du code.
- Optimiser les calculs de pulses (précision du simulateur).
- Fusionner les scripts avec la dernière version du projet de Las Vegas.
- ++ S'interesser à faire varier la vitesse et l'accelération du simulateur.