AMÉLIORATION D'UNE POIGNÉE DE COMMANDE

PROYECTO DE MEJORA DE UN DISPOSITIVO DE CONTROL DE UN SIMULADOR

RESUMEN EN ESPAÑOL Y DOCUMENTO ORIGINAL EN FRANCÉS

Mario Álvarez-Castrillón González

Ingeniería de Tecnologías Industriales

Trabajo Fin de Grado

Curso 2018/2019

Universidad de Oviedo

ÍNDICE:

El proyecto que se resume a continuación representa el trabajo de fin de grado presentado en la escuela de ingeniería de Centrale Nantes durante el curso 2018-19, en el equipo bajo la dirección de la profesora Mme. Sophie Sakka. Siguiendo las pautas usuales en la citada escuela, se atendía al requerimiento de una empresa con el propósito de ofrecer soluciones a diversos problemas del ámbito de la ingeniería. El caso

que se presenta precisaba diferentes perspectivas técnicas que fueron convenientemente estudiadas haciendo necesario combinar respuestas desde el campo de la electrónica y del diseño.

El trabajo, defendido el 18 de junio de 2019, obtuvo la máxima calificación por parte del tribunal.

1.- PRESENTACIÓN

1.1 La empresa

Hypercube es una empresa fundada en 2016 que gestiona un negocio combinado de restauración y sala de juegos. Su propuesta permite a particulares y empresas ofrecer una estructura móvil e innovadora para experimentar la realidad virtual.

Ese espacio de entretenimiento de nueva generación ha sido concebido como un entorno de descanso que ofrece a los usuarios experiencias de realidad virtual a la vez que disfrutan del servicio de restauración. Después de un año de desarrollo, este proyecto -inédito en Europa- se materializa en el centro de Nantes en septiembre de 2016.

1.2 El proyecto

La empresa posee dos *Icaros*, máquinas de simulación de vuelo en realidad virtual, con un coste por unidad de 7000 euros, utilizadas hasta el momento para la rehabilitación de personas discapacitadas. Los dispositivos permiten una inmersión completa sujetando al usuario mediante el único apoyo de sus antebrazos y tibias.

El dispositivo controlador de las máquinas es de plástico, y tiene un coste de 600 euros. Ligeramente adaptado por Hypercube para compensar un defecto de fábrica, se plantean una serie de problemas a resolver:

- El cable se desconecta a menudo a causa de su emplazamiento.

- La posición de los botones de comando en el controlador no es la idónea, además de ocupar toda la superficie frontal de la empuñadura lo que provoca su pulsación accidental.
- Resulta imposible transmitir información mediante Bluetooth, en tanto que existen demasiados aparatos funcionando mediante ondas en la estancia de las máquinas.

1.3 El contexto

Se ha comprobado que el mando no resulta ergonómico, de modo que un jugador novel tiene dificultades para empuñarlo y los botones están demasiado agrupados para resultar intuitivos al uso.

De otra parte, la empuñadura sirve para sujetarse, por lo que hace falta un sistema que pueda utilizar una persona sin pulsar involuntariamente los botones mientras se sujeta.

Las propuestas al comitente del trabajo, el gerente de la empresa, Pierre-Antoine Latimier, son las siguientes:

- Una conexión por cable que resuelva los problemas.
- Una empuñadura ergonómica más fácil de empuñar, inspirada en un freno de bicicleta
- Un sistema funcional con una puesta en marcha fácil (preferiblemente *plug and play*.)

2.- DESARROLLO

2.1 Objetivos

Los resultados esperados por la empresa son:

- Un estudio preliminar de un sistema alternativo al ofrecido por *Icaros*, intentando responder a las necesidades específicas de la empresa.
- Una primera versión de un mando mejorado y funcional, que haya superado las pruebas de conexión a ordenador.

2.2 Listado de tareas

A. Estudio del mando existente.
   a) Análisis de los protocolos de comunicación con el ordenador
   b) Estudio de los componentes internos de la maneta y del código fuente de las simulaciones.

B. Diseño de un nuevo mando
   c) Dibujos de las nuevas propuestas
   d) Diseño de los modelos 3D
   e) Descripción de los procesos de fabricación y de los materiales
   f) Programación de los componentes electrónicos

C. Fabricación del nuevo mando
   g) Evaluación de las fortalezas y debilidades de los prototipos previo a la construcción del prototipo definitivo.

3.- RESULTADOS

3.1- Análisis de los protocolos de comunicación controlador-ordenador

Las primeras comprobaciones evidenciaron que el sistema consideraba a la empuñadura de control un simple periférico *serial port*. Con objeto de determinar el *Baud rate* diversos ensayos con los valores clásicos <<9600, 115200, …>> no ofrecieron respuesta del controlador. Examinadas las experiencias de simulación suministradas por M. Latimier permitieron comprobar que la gestión de la comunicación se realizaba a partir de *Unity Engine*, un motor de juego muy utilizado, codificado en C#., examinando los ficheros se localizó el fichero DLL con el código fuente completo. La ayuda de un programa de descomprensión hizo posible encontrar ese código fuente y posteriores búsquedas permitieron determinar la gestión de la conexión al puerto de serie.

Se pudo determinar que una vez conectado el mando, el envío de órdenes no se iniciaba hasta después del envío de la cadena de caracteres <<START\r\n>>. Uniendo esta consigna a nuestro primer código de test se iniciaba la recepción de órdenes.

3.2- Componentes internos

Los componentes de la carta electrónica son los que siguen:

- Microcontrolador XMEGA3204
- Chip FTDI
- Chip Bluetooth BlueMod+SR Telit
- Acelerómetro
- Giroscopio
- Procesador

Estos componentes son bastante corrientes y pueden encontrarse a la venta a un precio asequible.

3.3- Alternativas para el nuevo mando

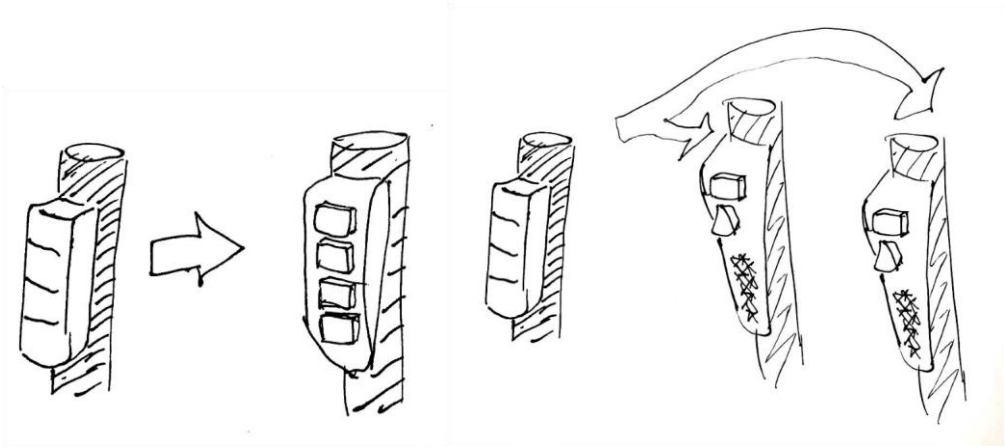
El mando existente revela diversos problemas: no es fácil de utilizar pues toda la empuñadura está cubierta de botones, además, las gafas de realidad virtual impiden al usuario comprobar las botones que está utilizando. Otro problema es el de la conexión USB al ordenador. Situada en un lateral incomoda al usuario y con frecuencia se desconecta.

Se proponen dos soluciones:

Una primera, simple y evidente, implicaría mejorar el mando actual aprovechando la misma electrónica.

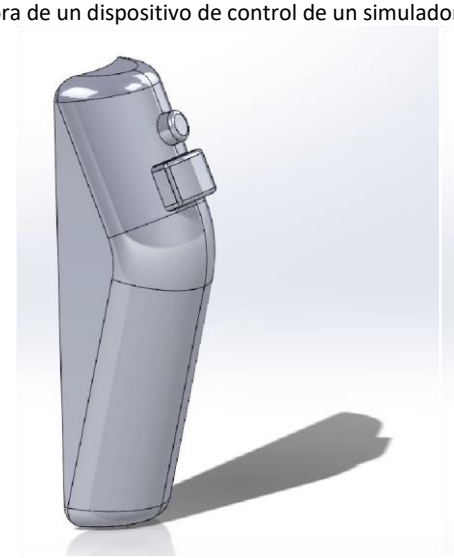
Consistiría esencialmente en cambiar el chasis y emplazar botones más pequeños pero más altos para diferenciarlos de las partes estructurales. A la vez el cable debería dirigirse al suelo evitando al usuario.

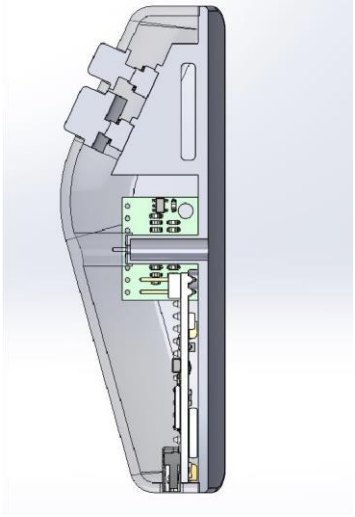
Otra propuesta, más recomendable, pasa por construir dos puños idénticos para cada mano repartiendo así las funciones de la botonera original en botones rediseñados, lo que implica, además, el diseño de un nuevo programa electrónico. (Puede verse en figura adjunta)

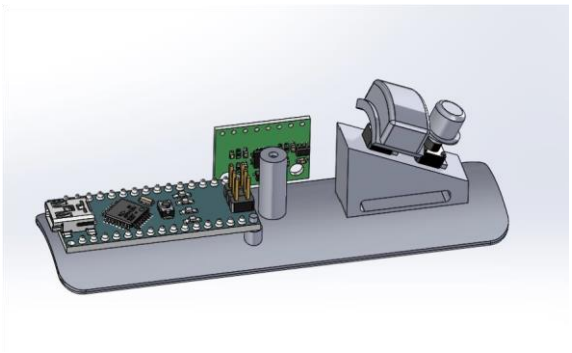


3.4- Diseño CAD

El programa de modelado elegido es SolidWorks, uno de los más utilizados del mundo y el más familiar en Centrale Nantes. La instrucción inicial daba prioridad a la concepción 3D del modelo, preservando el sistema electrónico original, lo que limitaba las posibilidades de ofrecer soluciones. Se propone finalmente rehacer la electrónica para poder distribuir las funciones en dos mandos de nueva creación.







3.5- Programación de los componentes electrónicos

Para programar la carta Arduino utilizada para la nueva configuración electrónica se utiliza un código en C, que será adaptado mediante Arduino IDE.

Los requisitos son:

- envío de información con correcta frecuencia y cadencia
- respuesta correcta a los mensajes “SERIAL” y “VERSION”
- validación de la información transmitida, sobremanera a nivel de los ángulos del giroscopio.

3.6- Determinación de los procedimientos de fabricación de los materiales

Respecto a la modificación de la ergonomía del dispositivo de control existía la posibilidad de modificar el material de su composición y dos opciones: conservar una empuñadura en plástico o fabricarlo con una aleación.

Mantener la empuñadura de plástico evitaba la -posiblemente larga- búsqueda de una aleación, resultando más ágil y al alcance de las impresoras 3D a disposición en Centrale Nantes, por lo que resultó finalmente la elegida. El resto de los componentes se compraron por internet.

El resultado puede apreciarse en la siguiente imagen:

4.- CONCLUSIÓN

Después de haber estudiado las diferentes posibilidades para realizar un nuevo mando, y en base al proporcionado por ICAROS, se realizaron las siguientes actuaciones:

- Estudio del código y del funcionamiento informático y electrónico del mando a fin de posibilitar la construcción de una réplica.
- Diseño y estudio de diferentes posibilidades de empuñadura para responder a la problemática planteada.
- Diseño por ordenador de la nueva empuñadura e impresión 3D de la misma.
- Programación según Arduino IDE de un nuevo programa electrónico.
- Evaluación y verificación del nuevo dispositivo.

El nuevo dispositivo ha sido aceptado por la empresa y funciona correctamente. El coste aproximado de su realización ha sido de unos 50€, resultando la nueva solución mucho más asequible que la precedente.

Année 2018-2019 **Sujet n° 38**

PROJET ÉTUDIANT D'ENTREPRISE
Rapport d'étude

Amélioration d'une poignée de commande

Groupe P2E
COCHARD Adrien

Correspondant Entreprise
Pierre-Antoine LATIMIER

MAZOUNI Anas
MEIRA Lucas
CASTRILLON Mario
ARNAUD Richard
BARBEREAU Élouan

Tuteur Enseignant
Sophie SAKKA

SOMMAIRE

24/05/2019

1. Présentation du projet dans son contexte

1.1. L'Entreprise

Hypercube est une entreprise qui fait à la fois café et salle d'arcade fondée en 2016. L'Hypercube – avec toutes ses activités - permet aux particuliers et entreprises de proposer une structure mobile et novatrice pour faire découvrir la réalité virtuelle lors de leurs évènements.

Cet espace de divertissement nouvelle génération a été pensé comme un lieu de vie et de rencontres. Les visiteurs peuvent se divertir grâce à des expériences en réalité virtuelle couplées à des simulateurs de mouvement ainsi que se restaurer.

Après un an de développement, ce projet novateur et inédit en Europe voit le jour et pose ses valises en plein cœur du centre-ville de Nantes, à deux pas de la place Graslin, en Septembre 2016.

1.2. Le Projet

C'est dans ce cadre que l'entreprise possède 2 Icaros, machines de simulation de vol en Réalité Virtuelle permettant d'avoir une immersion complète, coûtant 7000 euros pièce, où seuls les tibias et les avant-bras sont appuyés (figure 1). Les Icaros sont aujourd'hui surtout utilisé pour faire de la réhabilitation de personne handicapée. La poignée de commande de ces machines est en plastique, et coûte 600 euros. Elle a été légèrement bricolée par l'entreprise Hypercube pour compenser un défaut de fabrication au niveau du port manette. Comme la réalité virtuelle est utilisée comme attraction dans l'entreprise, de nouveaux joueurs passent plusieurs fois par jours sur les machines.

L'objectif du projet est résoudre les problèmes suivants : x À cause de son positionnement, le câble se débranche souvent. x La poignée et plus précisément la position des boutons n'est pas agréable, c'est en lien avec le problème d'ergonomie évoqué dans notre mission. x La transmission des signaux par port serial pose des problèmes de driver au niveau de Windows.

- Il est impossible de transmettre les informations par Bluetooth, dans la mesure où il y a déjà trop d'appareils fonctionnant avec des ondes dans la pièce où la machine fonctionne.

On donne en Annexe 1 quelques images supplémentaires de l'Icaros dans l'environnement de travail d'Hypercube.

Figure 1 : Une vue de l'Icaros provenant du fabricant

1.3. Le contexte et les contraintes

Pierre-Antoine Latimier, notre responsable au niveau de l'entreprise a pu voir par expérience que la poignée de commande n'est pas ergonomique: un nouveau joueur a du mal à la prendre en main, et les boutons sont trop resserrés pour que la prise en main soit intuitive. D'autre part, la poignée de commande sert aussi pour se tenir en jouant, il faut donc un système utilisable par une personne cramponnée à la poignée.

Les attentes de M. Latimier sont donc les suivantes :

- une communication filaire pour éviter les soucis, les HTC Vive utilisés communiquent déjà par infrarouges.
- une poignée qui soit ergonomique et facile à prendre en main, on pense par exemple pour cela à une poignée type frein de vélo.
- un système fonctionnel avec une mise en place facile (on préférera du plug and play).

2. Démarche

2.1. Les livrables attendus

Les livrables attendus par l'entreprise sont les suivants :

- Ce rapport qui permet d'apporter à l'entreprise une première étude d'un système différent de celui mis en place par Icaros, et essayant de répondre à des problématiques propres à l'entreprise.
- Une première version d'une poignée améliorée et fonctionnelle, avec des tests de connexion à un ordinateur réalisés.

2.2. Liste des tâches à effectuer

1) Étude de la poignée existante
    a) Analyse des protocoles de communication entre la poignée et l'ordinateur
    b) Étude des composants internes de la manette et du code source des simulations.
2) Conception d'une nouvelle poignée
    a) Maquettage papier des nouvelles propositions de la poignée.
    b) Conception des modèles 3D de la nouvelle poignée
    c) Détermination des procédés de fabrication et des matériaux
    d) Programmation des composants électroniques
3) Fabrication de la nouvelle poignée
    a) Analyse des points forts et des points à améliorer des premiers prototypes afin de construire le prototype final

3. Résultats obtenus

3.1. Analyse des protocoles de communication poignée ordinateur

Lors de notre première visite, nous avons remarqué que le système d'exploitation considérait la poignée un simple périphérique *serial port.* Dès lors, notre premier objectif était de déterminer les paramètres de cette connexion serial.

Le principal paramètre qu'il fallait déterminer est le *Baud rate*. Après plusieurs essais avec des valeurs classique « 9600, 115200… », nous n'avons reçu aucune donnée de la poignée.

Nous avons ensuite décidé d'examiner les expériences de simulation que M. Latimier nous a fourni lors de notre première visite et d'essayer de comprendre comment celles-ci gèrent la communication série.

Celles-ci semblaient être construites avec *Unity Engine*, un moteur de jeu largement utilisé, codé en C #. En examinant les fichiers, nous avons réussi à repérer le fichier DLL compilé contenant le code source complet.

À l'aide d'un décompilateur dnSpy, nous avons pu trouver ce code source et, après d'autres recherches, nous avons trouvé la classe qui gère la connexion au port série.

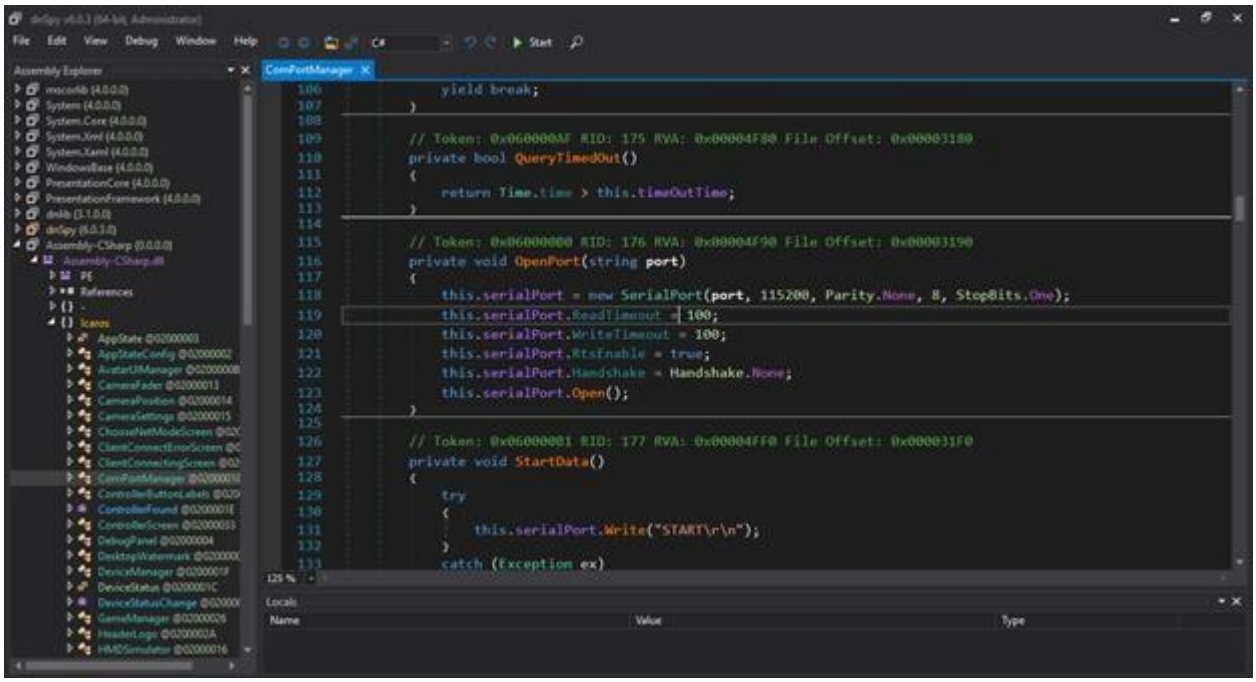


Figure 2 : Baud rate dans le code source du jeu

Nous avons découvert qu'une fois la poignée connectée par serial, l'envoi des données ne commence qu'après l'envoi de la chaîne de caractères « START\r\n ». En rajoutant cette consigne à notre premier code de test, nous commençons à recevoir des données, cependant, ces données ne sont pas codées en ASCII.

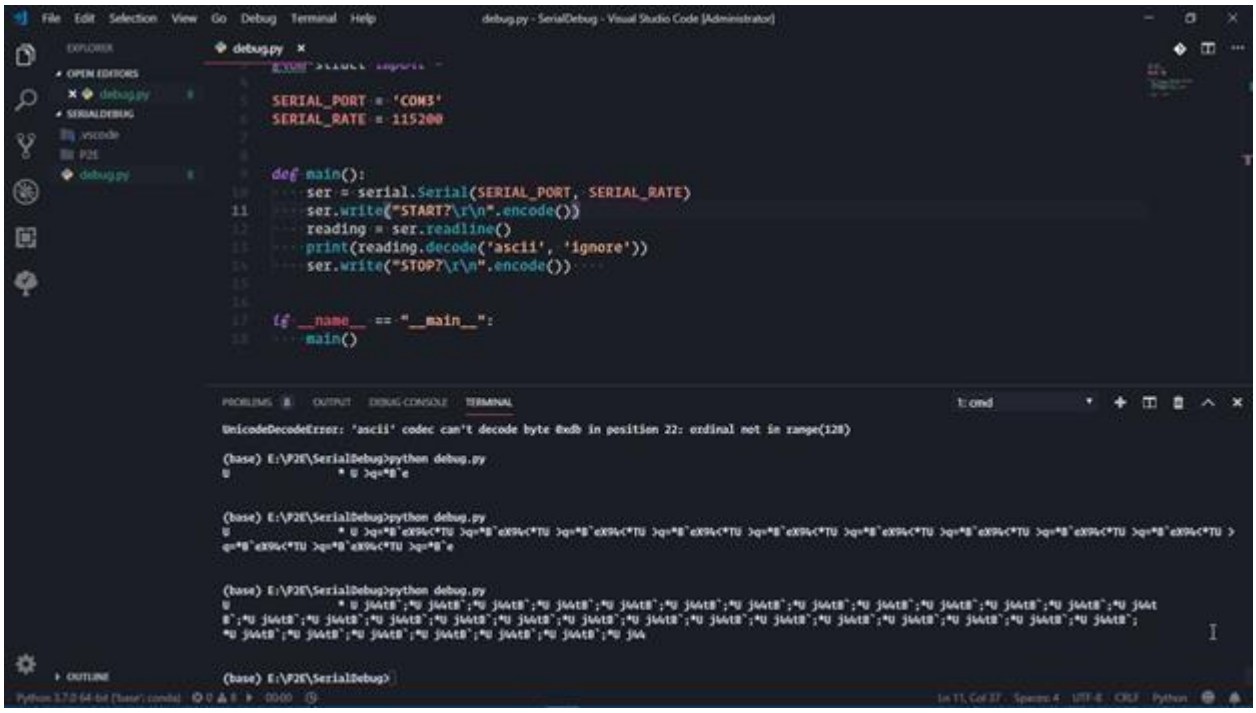
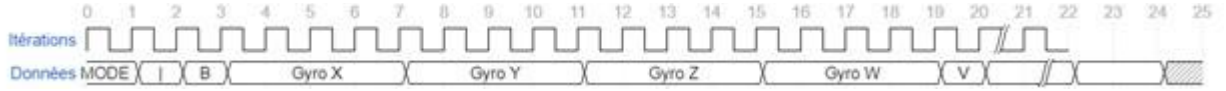


Figure 3 : lecture des informations renvoyées par la poignée

Malgré le fait que lorsqu'on envoie les commandes suivantes, nous recevons des caractères compréhensibles:

« **SERIAL ? \r\n** » - retourne : « Icaros1b15 »
« **VERSION ?\r\n** » - retourne : « Icaros HW3.0 SW2.6.1 »

Il fallut alors par suite analyser plus le code source des expériences pour comprendre comment les données reçues sont traitées. Nous avons découvert que les données sont divisées en paquets de 20 octets et chaque paquet est de la forme suivante :



- **MODE** : un octet qui permet de déterminer le mode de lecture : 0x85 = "Controller DATA".
- **|** : Un séparateur non utilisé.
- **B** : permet de déterminer l'état des 5 boutons à l'aide de l'opérateur bit à bit « & : et logique ».
- **Gyro** : chaque 4 octets (32 bits) forment un nombre en virgule flottante représentant l'angle mesurée.
- **V** : Tension de la batterie interne.

Remarque: les 4 nombres retournés ne sont pas des angles mais forment un quaternion à partir du quelle on pourra déduire les 3 angles de rotation dits d'Euler.

Nous avons rédigé un programme qui permet de décoder ces paquets de données, ce qui nous permettra dans le futur de tester et concevoir un nouveau système électronique.

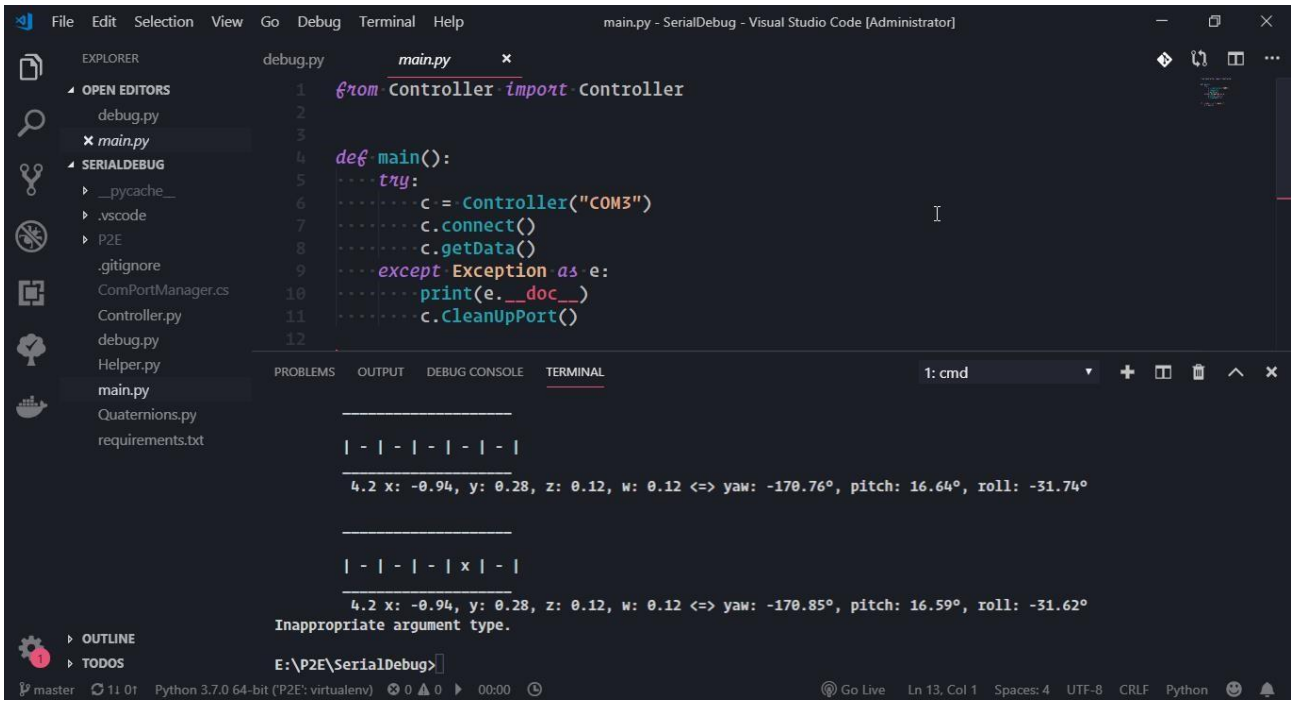


Figure 4 : conversion des informations renvoyées par la poignée en données compréhensibles

Le code source utilisé pour les simulations peut être trouvé en Annexe 2.

3.2. Les composants internes et le code source des simulations

Les composants – après une analyse fine et avec la fiche produit – de la carte électronique existante sont les suivants : x Microcontrôleur XMEGA3204 x Chip FTDI

x Chip Bluetooth BlueMod+SR Telit x Accéléromètre x Gyroscope x Processeur

Ces composants sont assez standards et peuvent être trouvés à faible prix dans le commerce. On ne détaille ici pas leur fonctionnement, car il ne nous est pas nécessaire d'avoir de telles connaissances dans notre étude : pour l'étape de réalisation d'une nouvelle carte électronique, on branchera juste des composants similaires à une carte Arduino.

3.3. Maquette papier des différentes possibilités envisagées pour la nouvelle poignée

Figure 5 : détails de la poignée

La poignée actuelle pose plusieurs problèmes : d'abord elle n'est pas facile à utiliser car toute la façade, où l'utilisateur doit se tenir, est entièrement couverte de boutons. De plus, comme l'utilisateur porte des lunettes de réalité virtuelle, il ne peut même pas regarder pour voir exactement les boutons sur lesquels il appuie.

Un autre problème est le placement de la connexion USB vers l'ordinateur (figure 5). Elle se situe sur un côté, ce qui rend la manette moins confortable et déconnecte le câble de temps en temps (D'où la solution temporaire trouvé par l'entreprise consistant à coller le câble avec de la pâte à modeler).

Après une étude complète de la poignée de commande, où nous regardons les aspects ergonomique et électronique, nous proposons deux solutions :

Une première, plus simple et évidente, consiste à améliorer la manette actuelle sans changer l'intérieur (utilisant donc la même carte électronique).

Cette solution consiste essentiellement à refaire le châssis (partie extérieure) en faisant des boutons plus petits mais plus hauts pour bien les différencier des parties structurales (figure 6). Le câble doit être aussi dirigé vers le sol afin de ne pas gêner l'utilisateur.

Cependant cette solution ne résout pas vraiment le problème de praticité, puisque la plupart des nouveaux utilisateurs se crispent sur la poignée, et appuieront donc toujours sur les boutons.

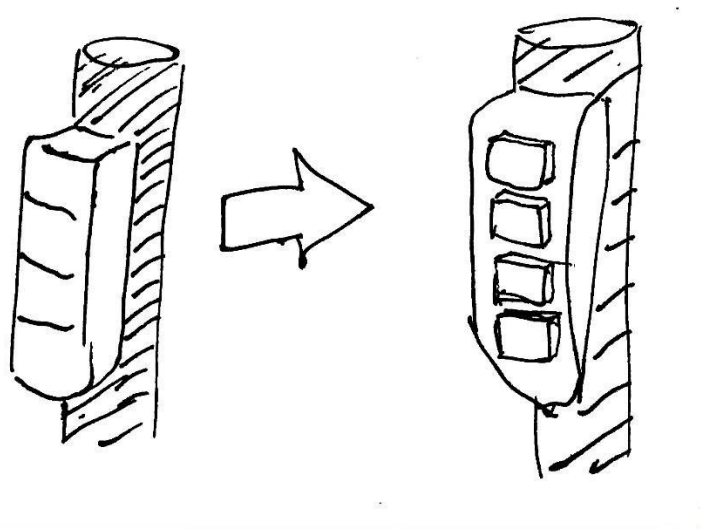


Figure 6 : Dessin de poignée unique

Ainsi, la solution que nous retenons est la suivante : réaliser deux poignées identiques (une pour chaque main) en répartissant les boutons (figure 7). Même si cette solution est plus compliquée du point de vue technique (elle implique la création d'une carte électronique complètement nouvelle) elle présente de grands avantages.

Premièrement, elle présente une surface suffisamment grande pour ne toucher les boutons que quand l'utilisateur a vraiment l'intention de le faire, et le réflexe de serrer la poignée ne pose donc plus de problème.

Deuxièmement, cette distribution de boutons rend l'utilisation beaucoup plus simple et intuitive : On a Accélérateur / Accepter sur la commande droite et Frein / Refuser sur la gauche. Notons que les 2 boutons présents sur une poignée sont distinguables au toucher, puisqu'ils n'ont pas la même forme. Cela permet également de s'habituer plus rapidement à leur disposition. De plus, les boutons du haut sont situés suffisamment loin de la main pour ne pas être pressés involontairement, et ne seront donc utiles que dans le menu du jeu, où l'utilisateur n'a pas besoin d'être concentré sur la direction de son véhicule, et n'est donc généralement pas crispé sur les manettes.

L'une des manettes aura donc la plupart des composants électroniques, et une sortie USB ; l'autre n'aura que ses boutons et une sortie USB. Les deux manettes seront donc liées pour permettre la transmission des données des boutons vers la carte électronique.

Figure 7 : Dessin avec dédoublement de la poignée

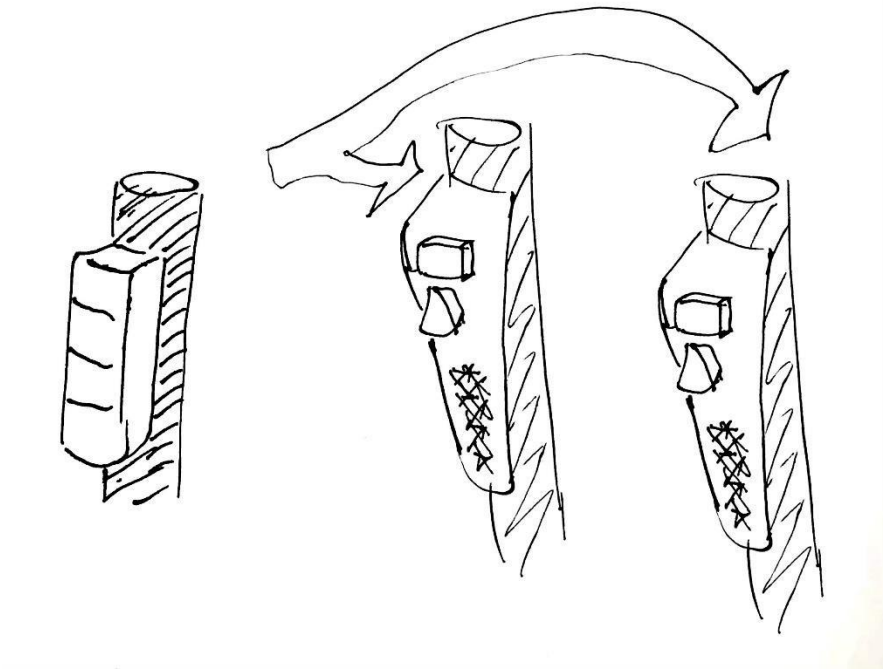


Figure 7 : Dessin avec dédoublement de la poignée

Après plusieurs essais et dimensionnements on arrive à la proposition donnée dans la partie suivante, représentée sous SolidWorks.

3.4. CAO

Le logiciel de modélisation choisi est SolidWorks. Il est l'un des plus utilisés au monde. Utilisé aussi dans les cours à Centrale, c'est le logiciel avec lequel on est plus à l'aise.

Le projet d'une nouvelle carte électronique peut être trop compliqué et, comme conseillé par notre accompagnatrice, ne devrait pas être la priorité. Afin de commencer à proposer des solutions plus ergonomiques, on choisit de garder la carte électronique originale pour l'instant et proposer des modifications de sa structure.

Vu que la priorité est la conception 3D du modèle, en gardant la carte électronique originale, on a moins de liberté pour donner des solutions. On propose alors une solution avec les quatre boutons dans la même main, en faisant spécialement attention à l'ergonomie des boutons. On étudiera des possibilités et faisabilités d'implémentation des gâchettes ou de différents boutons, avec la contrainte de la carte originale. Si on arrive à avoir une nouvelle carte électronique, on pourra proposer solutions différentes pour les joueurs.

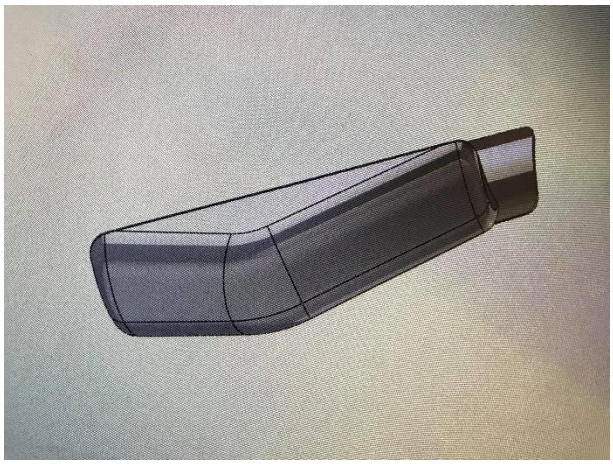


Figure 8 : première modélisation des poignées

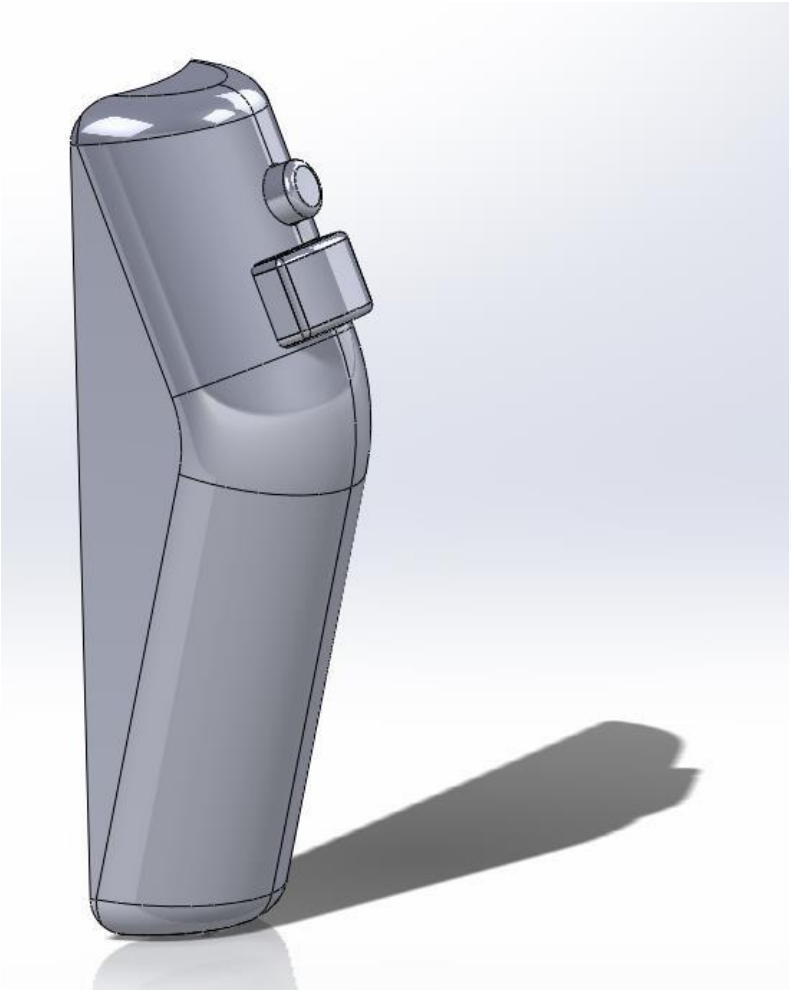


Figure 9 : Ajout des boutons sur une poignée

Figure 10 : Ajout des différents composants électroniques

3.5. Programmation des composants électroniques

Pour programmer la carte Arduino utilisée pour la nouvelle carte électronique, on utilise un code en C, qui sera ensuite passé sous Arduino IDE. Il a y plusieurs contraintes à respecter pour la

programmation de cette nouvelle carte électronique, qui sont des contraintes de mimétisme par rapport à la carte électronique du système existant :

- Envoi d'information avec la bonne cadence/fréquence.
- Réponse correcte au messages 'SERIAL' et 'VERSION'
- Validité de l'information transmise, surtout au niveau des angles du gyroscope (transformation d'angle d'Euler en Quaternions).

On trouve le code source écrit en C en Annexe 3. Pour pouvoir programmer ce code sans problème techniques, on utilise un simulateur de carte Arduino :

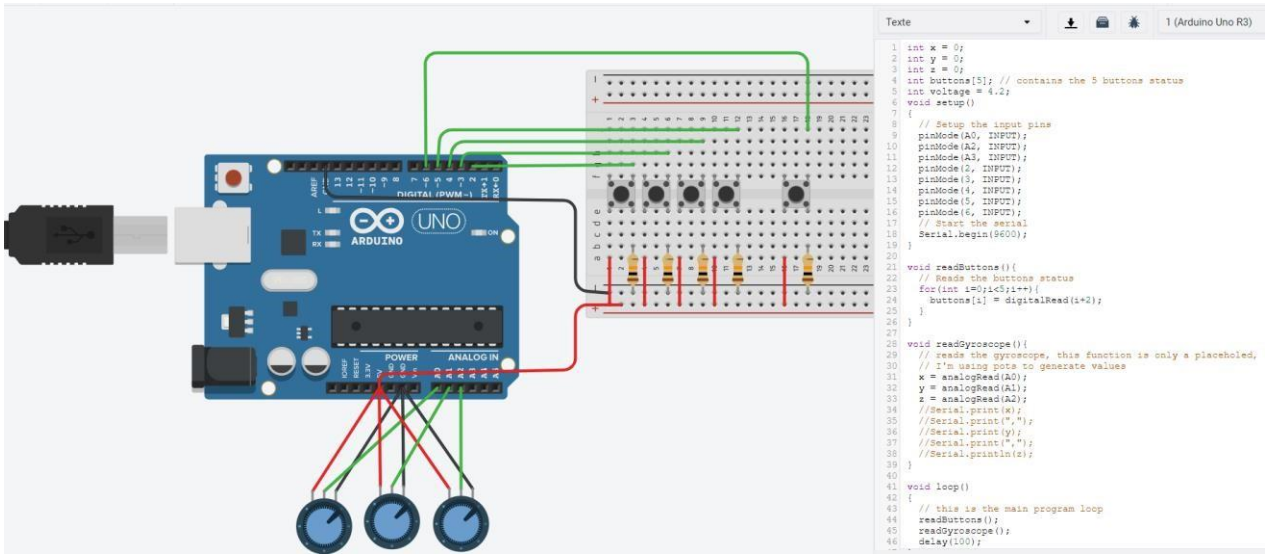


Figure 11 : Simulateur Arduino pour commencer le code

Figure 11 : Simulateur Arduino pour commencer le code

3.6. Détermination des procédés de fabrication des matériaux

Suite à la modification de l'ergonomie de la poignée de commande nous avions le choix de modifier le matériau de base de celle-ci.
Nous avions ainsi le choix entre plusieurs solutions :

- La première était de conserver une poignée en plastique -
  La seconde était de la fabriquer avec un alliage

Le choix de conserver le plastique comme matériau premier paraissait évident afin de conserver les mêmes propriétés que l'ancienne poignée. De plus, les recherches pour trouver un alliage adapté aurait supplanté le travail de modélisation et mis le groupe en retard. Le principal avantage de fabriquer la poignée en plastique est aussi la simplicité de fabrication à l'aide des imprimantes 3D, et nous avions la possibilité de la concevoir à Centrale ou bien via l'intermédiaire de Mr. Latimier qui avait la possibilité d'en utiliser une.
Finalement, la poignée a été conçue en plastique à l'école Centrale par imprimante 3D. Le reste des composants a été acheté sur internet.

4. Conclusion

Après avoir étudié les différentes possibilités pour la réalisation d'une poignée de commande, en nous basant sur celle existante produite par la société Icaros, nous avons donc réalisé les étapes suivantes :

- Étude du code et fonctionnement informatique et électronique de la poignée dans le but de pouvoir en construire une par mimétisme
- Dessin et études des différentes possibilités de poignée pouvant répondre à la problématique
- Dessin sur ordinateur puis conception de la coque de la poignée
- Programmation sur Arduino IDE d'une nouvelle carte électronique

Cette étude est donc un bon départ dans la réalisation d'une nouvelle poignée : tout est calculé, ne reste plus qu'à mettre en œuvre. Il faudra aussi adapter et remodifier que ce soit la coque ou la carte électronique selon les défauts et biais constatés à l'usage.

5. Table des illustrations

6. Annexes

6.1. Images de l'Icaros

Photos réalisées par nos soins dans l'entreprise Hypercube

6.2. Code source utilisé pour le reverse engineering de la carte électronique de l'ancienne poignée

On présente ici le code utilisé pour comprendre comment la carte électronique de la poignée fonctionne et les différents messages transmis. Cette étape est nécessaire dans le sens où il est impossible de modifier le code source des jeux donc on doit faire croire à l'ordinateur qui fait tourner le jeu que la manette utilisée est celle originale de l'Icaros.

```
using System; using
System.IO.Ports; using
System.Text;

namespace P2E
{
    class program
    {         private static SerialPort
serialPort;
```

```
    private  const float RECONNECT_WAIT_TIME = 3f;
```

```
        private   const float RECONNECT_GIVEUP_TIME = 5f;
private  const float RESCAN_TIMEOUT = 2f;          private
static string lastPeripheralName;           private static
string disconnectError;              private static bool
reconnecting;        private static byte[] dataBuffer = new
byte[20];         private static byte[] messageBuffer = new
byte[40];                private static int bufferIndex;
private static float timeOutTime;           private static
float portTimeout = 2f;         private static bool error;
public static bool FirstButtonIsDown;         public static
bool SecondButtonIsDown;              public static bool
ThirdButtonIsDown;                   public static bool
FourthButtonIsDown;                  public static bool
FifthButtonIsDown;                  public static float
IcarosPitchAngle;                   public static float
IcarosRollAngle;             public static float Voltage;
private  const float halfPI = 1.57079637f;         private
const float quarterPI = 0.7853982f;         private static
float GamePadControlSpeed = 1f;        private static float
MaxPitchAngle = 0.436332315f;          private static float
MaxRollAngle = 0.610865235f;          private static string
currentReadMode;         private  enum ReadMode
        {
            None,
            Data,
            Message
        }
        static void Main(string[] args)
        {
            // Instatiate this
            SerialPortProgram();
        }
        private static void OpenPort(string port)
        {
            serialPort = new SerialPort(port, 115200, Parity.None, 8,
StopBits.One)
            {
                ReadTimeout = 100,



                WriteTimeout = 100,
```

```
            RtsEnable = true,
            Handshake = Handshake.None
        };
        serialPort.Open();
    }
```

```
         private static void
SerialPortProgram()
        {
            OpenPort("COM3");
            serialPort.DataReceived += new
SerialDataReceivedEventHandler(port_DataReceived);
            // Begin communications
serialPort.Write("START\r\n");
            Console.ReadLine();
        }
        private static void port_DataReceived(object sender,
SerialDataReceivedEventArgs e)
        {
         if (serialPort != null &&
serialPort.IsOpen)
        {             bool
flag = false;
            // Show all the incoming data in the port's
buffer             try             {               while
(serialPort.BytesToRead > 0)
             {                 int num =
serialPort.ReadByte();
             if (bufferIndex ==
0)
            {
if (num == -1)
                {
                    currentReadMode = "none";
                }
                else if ((byte)num == 85)
                {
                    currentReadMode = "data";
                }
else                 {
                    currentReadMode = "msg";
                }
}
            if (currentReadMode == "data")
            {
                if (num > -1)
                {
dataBuffer[bufferIndex++] = (byte)num;
```

```
                }                if
(bufferIndex == 20)
```

```
                {
                    bufferIndex = 0;
                    HandleNewBytes(dataBuffer);
                }             }
else if (currentReadMode == "msg")
            {
if (num > -1)
                {
messageBuffer[bufferIndex++] = (byte)num;
                }                 if (bufferIndex > 2 &&
messageBuffer[bufferIndex - 1] == 10 && messageBuffer[bufferIndex - 2] == 13)
                {                                                  string  @string  =
Encoding.ASCII.GetString(messageBuffer, 0, bufferIndex - 2);
                    bufferIndex = 0;
                    Console.WriteLine("msg: {0}",new object[] { @string });
                }
            }
                    }
            }
catch             {
                flag = true;
                    Console.Write("error");
            }
            }
        }
        public static void HandleNewBytes(byte[] bytes)
        {
            int num = (int)bytes[1];
bool flag = (num & 1) != 0;             if
(!FirstButtonIsDown && flag)
            {
                Console.WriteLine("first button clicked");
            }
            if (FirstButtonIsDown && !flag)
            {
                Console.WriteLine("first button released");
```

```
            FirstButtonIsDown = flag;
bool flag2 = (num & 2) != 0; if
(!SecondButtonIsDown && flag2)
```

```
        {
            Console.WriteLine("second button clicked");
```

```
            }             if
(SecondButtonIsDown && !flag2)
            {
                Console.WriteLine("second button released");
}
            SecondButtonIsDown = flag2;
bool flag3 = (num & 4) != 0;             if
(!ThirdButtonIsDown && flag3)
            {
                Console.WriteLine("third button clicked");
            }             if
(ThirdButtonIsDown && !flag3)
            {
                Console.WriteLine("third button released");
            }
            ThirdButtonIsDown = flag3;
bool flag4 = (num & 8) != 0;             if
(!FourthButtonIsDown && flag4)
            {
                Console.WriteLine("fourth button clicked");
            }             if
(FourthButtonIsDown && !flag4)
            {
                Console.WriteLine("fourth button released");
}
            FourthButtonIsDown = flag4;
bool flag5 = (num & 16) != 0;             if
(!FifthButtonIsDown && flag5)
            {
                Console.WriteLine("fifth button clicked");
            }
            if (FifthButtonIsDown && !flag5)
            {
                Console.WriteLine("fifth button released");
}
            FifthButtonIsDown = flag5;
            float new_x = BitConverter.ToSingle(bytes, 2);
float new_y = BitConverter.ToSingle(bytes, 6);             float
num2 = BitConverter.ToSingle(bytes, 10);             float num3
= BitConverter.ToSingle(bytes, 14);
            Voltage = (float)bytes[18] / 10f;
            Console.WriteLine("Voltage: {0}, x: {1}, y: {2}, 2: {3}, 3: {4}",
new object[] { Voltage, new_x, new_y, num2, num3 });
        }
```

}

Centrale Nantes Sujet n° 38 Hypercube

```
}
```

À la fin de l'utilisation de ce code, on récupère les différentes données de fonctionnement de la poignée sous une forme assez brute. On utilise alors du code Python pour analyser les données fournies par le code C# :

```
import serial import
time from time import
sleep  import struct

SERIAL_PORT = 'COM3'
SERIAL_RATE = 115200
 dataBuffer = [0x00]*20 # Contains bytes read from the serial
buffer messageBuffer = [0x00]*40 # TODO: remove messageBuffer, it's
useless bufferIndex = 0 # used to go through the data buttons =
[False] * 5 #Buttons default status
 def
main():
    global dataBuffer,messageBuffer,bufferIndex

    ser = serial.Serial(SERIAL_PORT,
SERIAL_RATE,timeout=100,write_timeout=100,rtscts=True)
    # To check the versions
    # ser.write("VERSION?\r\n".encode())
    # reading = ser.readline()
    # print(reading)     #
sleep(2)
ser.write("START\r\n".encode())
sleep(2)
    while ser.in_waiting:  # Or: while
ser.in_waiting         num = ord(ser.read())
if bufferIndex == 0:
            # TODO: figurout the other read modes & their use
cases                 if num != -1:                 if num !=  85:
                    currentReadMode = "message"
else:
                    currentReadMode = "DATA"
else:
                currentReadMode   =   "None"
if currentReadMode == "DATA":



            if num > -1:
```

```
                dataBuffer[bufferIndex] = num
bufferIndex += 1 if bufferIndex == 20:
                bufferIndex = 0
                HandleNewBytes(dataBuffer)
 def
HandleNewBytes(dataBuffer):
    global buttons     num =
dataBuffer[1]     buttons[0] = (num & 1)
!= 0     buttons[1] = (num & 2) != 0
buttons[2] = (num & 4) != 0
buttons[3] = (num & 8) != 0
buttons[4] = (num & 16) != 0     x =
bit32ToFloat(dataBuffer[2:6])     y =
bit32ToFloat(dataBuffer[6:10])     z =
bit32ToFloat(dataBuffer[10:14])     w =
bit32ToFloat(dataBuffer[14:18])
Voltage = (dataBuffer[18] & 127) / 10
    # (x,y,z,w) formes un quaternions
    # mais le gyroscope ne nous retourne que 3 angles, les angles d'euler
(precession, nutation, rotation propre)
    #
https://en.wikipedia.org/wiki/Conversion_between_quaternions_and_Euler_angles
# faut créer une fonction qui prend le role/pitch/yaw et retourne 4 angles
print(buttons)
    print(x,y,z,w,Voltage)
 def bit32ToFloat(c):     byts =
bytearray()     byts.extend(c)
return struct.unpack('<f', byts)[0]
 if __name__ ==
"__main__":
    main()
```

6.3. Code source de programmation de la carte Arduino

Le code suivant a pour but d'imiter le code de la poignée originale de l'Icaros. On essaye donc dans la mesure du possible de faire en sorte que la carte électronique ici programmée renvoie les bons messages.

Ce n'est pas une version définitive du code, il est encore en développement à ce jour, car le prototype est encore en phase de test.

```
```

```
#include <Cmd.h>
 float x =
0; float y
= 0; float
z = 0;
float w =
0;
```

```
bool buttons[5] = {false, false, false, false, false}; // contains the 5
buttons status float voltage = 4.2; bool start = false; void setup()
{
    // init the command line and set it for a speed of
115200     Serial.begin(115200);     cmdInit(&Serial);

    //Note: Commands are case sensitive
cmdAdd("START", cmd_start);     cmdAdd("STOP",
cmd_stop);     cmdAdd("VERSION?",
cmd_version);     cmdAdd("SERIAL?",
cmd_serial);
}  void
loop()
{     cmdPoll();
if(start){
printData();
    }
}  void hello(int arg_cnt, char
**args)
{
    Serial.println("Hello world.");
}
//called for START command void
cmd_start(int arg_cnt, char **args)
{
    start = true;
}

//called for STOP command
void cmd_stop(int arg_cnt, char **args)
{
    start = false;
}

//called for VERSION? command
void cmd_version(int arg_cnt, char **args)
{
    cmdGetStream()->println("V0.1"); }
```

```
//called for SERIAL? command void
cmd_serial(int arg_cnt, char **args)
{     cmdGetStream()-
>println("P2E_38");
}


void printData(){
x = random(0,5);
y = random(0,5);
z = random(0,5);
w = random(0,5);
byte b = 22;
    byte buffer[] = {85,b,x,y,z,w,voltage};
    Serial.write(buffer,7);
} `
```