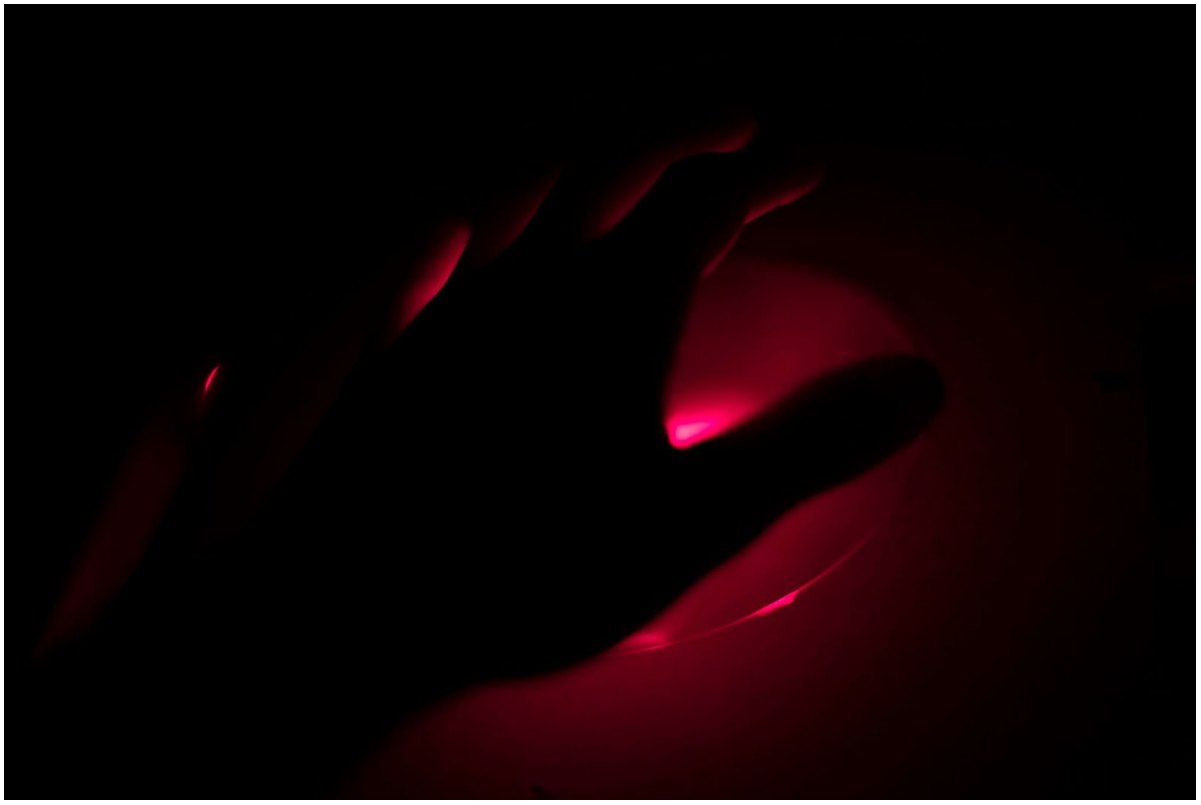
Rapport technique

Projet Pims Protis Art et Sciences
Light touch



Jad Aoun
Hippolyte Dupont
Hector Izard
Cécile Le Gall
Ewan Mer

Encadrants :
François Balembois
Eric Michel

Sommaire :

Introduction

Dans le cadre des enseignements PIMS et PROTIS, encadrés par François Balembois, Eric Michel et toutes l'équipe enseignante du Lense, nous avons réalisé le projet *Light Touch*. Notre but était de réaliser un projet alliant art et sciences pour montrer la beauté de la lumière. Nous voulions à travers ce projet permettre au spectateur de toucher et voir la lumière. Aussi, nous avons cherché à détourner la lumière d'une découpeuse laser et de l'associer à une diode rouge afin de permettre au visiteur de en même temps toucher et voir la lumière en détournant une technologie préexistante. Dans ce rapport, nous essaierons de vous décrire à la fois l'aspect artistique et technique de ce projet, de vous décrire notre avancement, malheureusement perturbé par le confinement et les solutions que nous avions mis en place ou que nous proposons de mettre en place pour terminer le projet et rendre notre oeuvre opérationnelle.

Quelques références artistique :

Dans notre projet, nous nous sommes intéressés dans un premier temps au traitement de la lumière par les artistes. Comment ont-ils fait pour révéler la beauté de la lumière ? Comment ont-intégré la lumière à leur oeuvre ? Comment pourrions nous nous en inspirer ? Quels sont les sensations et les sentiments que cela suscite en nous ?

Dans notre recherche, quelques oeuvres ont émergé et nous ont particulièrement inspirés.

Solid Light Work, Anthony McCall, 2009

Art, Mario Merz, 1969

Dan Flavin et Donald Judd, 2013

Grégory Lasserre & Anaïs, Met den Ancxt

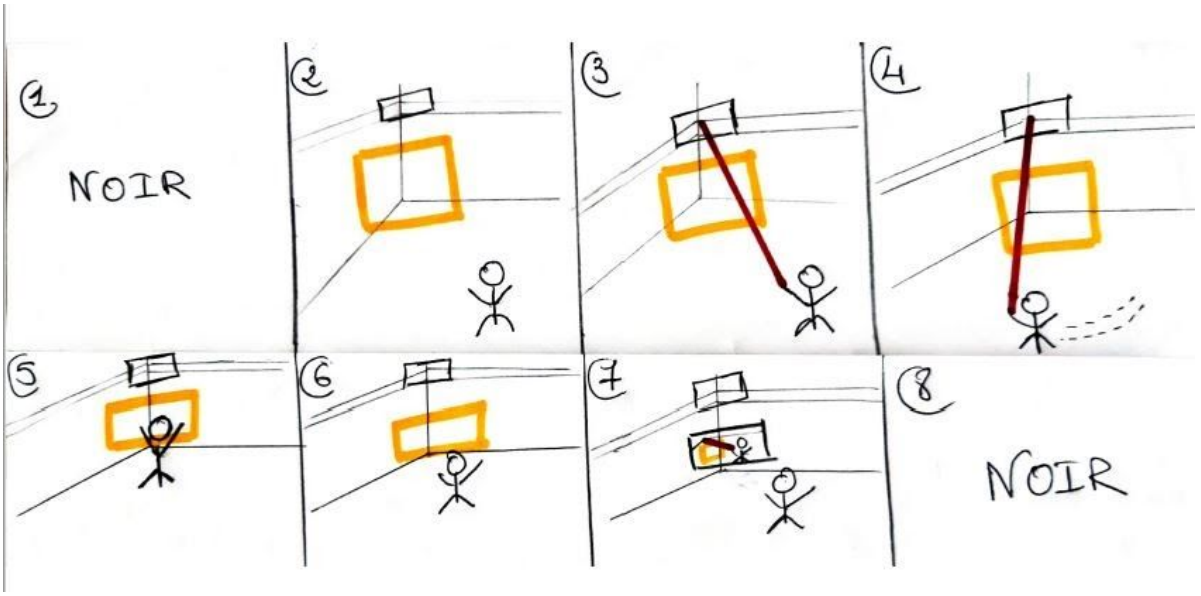
A travers ces quatre oeuvres nous nous sommes interrogés sur comment peut-on faire toucher la lumière? Comment peut-on la matérialiser? Quelle est l'importance du point de vue de la façon de voir? Comment peut-on rendre une oeuvre immersive et interactive?

Suite à cet état de l'art, nous avons conçu un projet qui avait l'ambition d'être une oeuvre interactive et immersive que nous allons vous décrire dans la partie suivante de ce rapport.

Le projet artistique :

Nous avions conçu ce projet comme un scénario dans lequel le spectateur serait le personnage principal et entrerait dans une pièce pour venir interagir avec notre oeuvre. Ainsi il ferait partie intégrante de l'oeuvre.

Aussi, pour décrire notre projet, nous nous proposons de nous appuyer sur un scénario en plusieurs étapes.



L'oeuvre est installée dans une pièce noire. Le spectateur entre. Dans un coin au fond s'allume un cadre lumineux. Puis le visiteur sent une source de chaleur sur sa main. Il regarde sa main et voit une lumière rouge qui le relie à l'oeuvre et entre ainsi dans l'oeuvre. Lorsqu'il bouge la lumière le suit. Il s'approche du cadre. Lorsqu'il est assez près le cadre s'éteint et le spectateur se voit sur un écran ou plus exactement une vidéo de lui en train de voir la lumière est projetée sur un écran. Il fait alors doublement partie de l'oeuvre.

Le projet technique :

Un fois l'oeuvre imaginée il nous faut maintenant la réaliser techniquement. Il faut mettre les mains dans le cambouis maintenant !

Les principales matérialisations techniques de nos idées artistiques sont les suivantes :

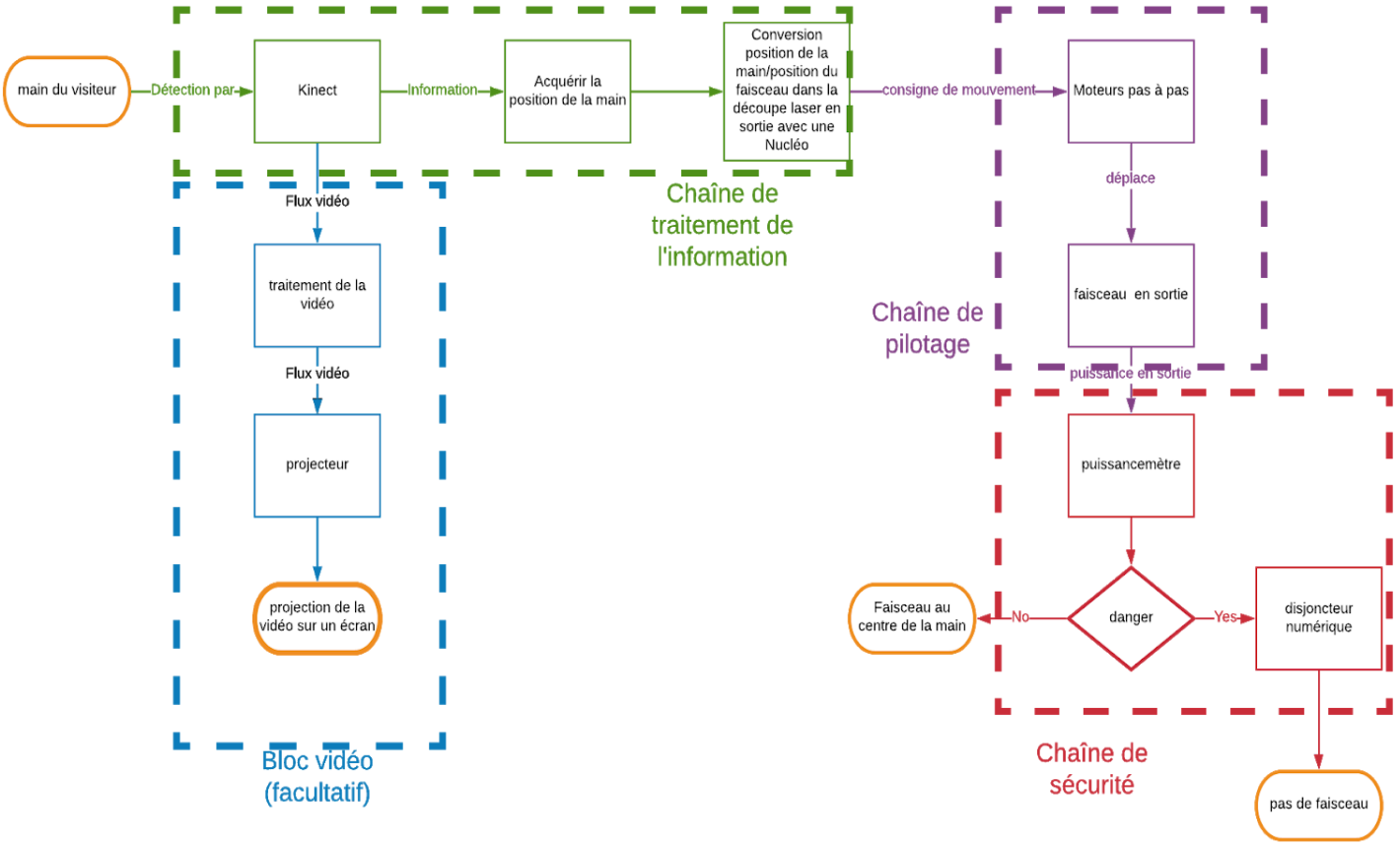
- Utiliser la lumière de longueur d'onde de $10\,\mu m$, qui est une lumière du domaine de l'IR et qui est donc invisible. Mais nous pouvons la sentir, elle chauffe dans la main. Et c'est justement la longueur d'onde d'émission des lasers CO2 qui se trouvent dans les découpeuses lasers. On va donc travailler au détournement d'une découpe laser
- Une Kinect, qui est un appareil développé pour la Xbox qui fait caméra (visible +IR) et détecte les mouvements. Elle détecte en particulier la main d'un observateur et transmet ses coordonnées à un système de pilotage des faisceaux pour les diriger sur la main.
- Le tracking sera opéré par une lentille qui doit bouger et orienter le faisceau pour suivre la main de l'individu. Et justement une découpeuse laser est dotée de bras motorisés! On ne détournera donc pas que le laser CO2 de la découpe mais aussi ses moteurs! C'est tout l'art de la frugalité!

Aussi bien dans la conceptualisation que dans notre façon de travailler, cette partie technique peut se découper en plusieurs blocs résumés sur la figure ci-dessous.

L'idée de départ est donc de partir d'une découpeuse laser préexistante et de la transformer, de la détourner pour l'utiliser à nos fins. Nous voulions utiliser ses moteurs, une partie de ses miroirs, son contenant et bien sur son laser CO2 afin de réaliser dans notre oeuvre la partie la plus importante la partie permettant de voir et de toucher en même temps la lumière.

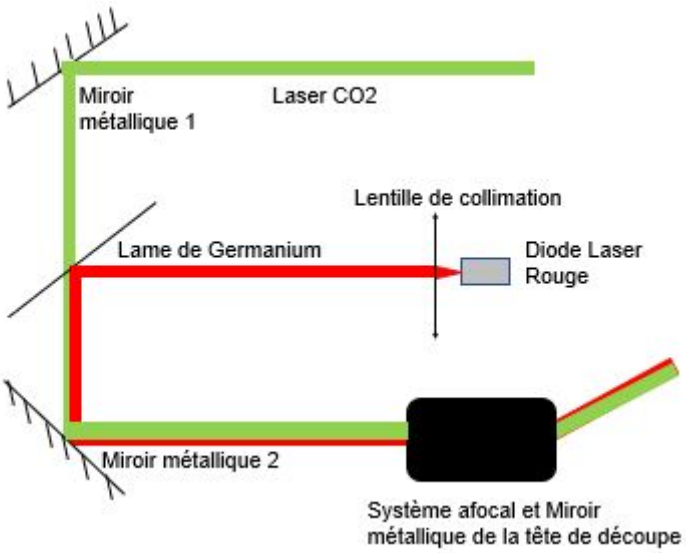
Dans cette partie nous évoquerons les différentes parties techniques de notre projet à savoir :

- La conception optique
- La détection avec la kinect
- L'interface graphique pour commander les laser
- La supervision des différentes tâches
- La partie motorisée
- Le cadre
- La partie vidéo



1.La conception optique

Pour la conception du système optique, on est parti de ce qu'on avait, c'est à dire un laser CO_2 réfléchie par un premier miroir métallique puis par un deuxième qui le dirige vers la tête de la découpe laser puis un dernier qui le focalise directement dans le plan de découpe. Le laser étant invisible, nous devons le coupler avec un faisceau visible. En plus nous devons modifier la fin du trajet , car nous voulons collimater le faisceau en sorti de la découpe et non pas le focaliser.
Voilà la solution adoptée :

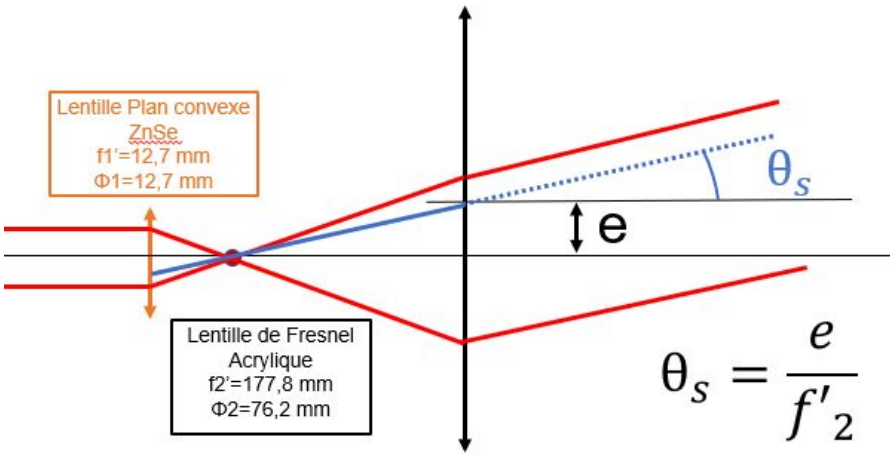


La lame de germanium est réfléchissante dans le visible et transparente dans l'IR !

Pour réussir le détournement de la découpe Laser, on devait concevoir une système optique permettant de convertir les coordonnées (X,Y) de la tête de découpe en une direction (θ,φ) dans l'espace de la pièce tout en mettant en forme le faisceau pour qu'il fasse la taille de paume de la main.

Comme solution technique, on a eu l'idée d'utiliser un système afocal avec la première lentille fixée sur la tête de la découpe laser et la deuxième immobile dans le référentiel de la découpe. Le principe est de désaxer l'axe optique des deux lentilles, plus les axes optiques des deux lentilles sont déaxés, plus le faisceau collimaté en sortie sortira avec un angle important. Ainsi On a ainsi une relation directement proportionnel entre (X,Y) et (θ,φ) illustrée sur la figure ci-dessous.

Le choix des focales étant choisies de telle sorte à obtenir un grossissement permettant d'obtenir un faisceau tenant dans la paume.

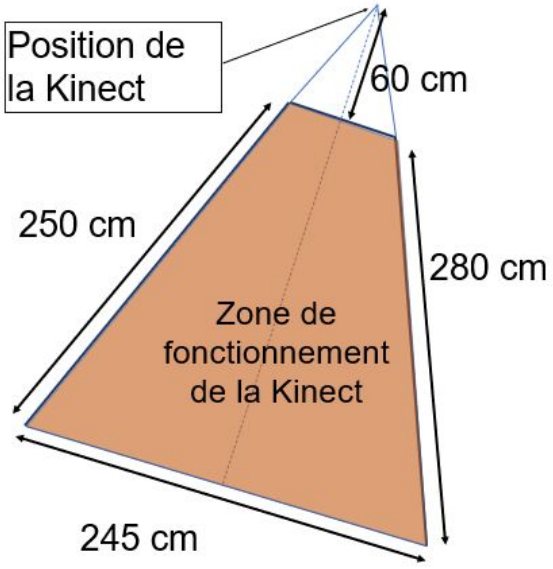


Il reste à déterminer un dernier duo d'optiques qui permet de collimater la combinaison des faisceaux rouge et CO2 et qui permet également de balayer une bonne région de l'espace car, le but est que le faisceau de sortie suive la main de l'utilisateur. On sait que les lentilles de ZnSe sont transparentes à 10 μm et on a évalué leur transmittance dans le rouge à 30% .

On peut voir que la plage totale (θ,φ) sera limitée par le diamètre de la deuxième lentille. Pour avoir un champ raisonnable, il faudra une lentille relativement grande tout en restant suffisamment fine et légère pour pouvoir être implémenté facilement sur la découpe. Pour toutes ces raisons, nous nous sommes tournées vers une association lentille de ZnSe + lentille de Fresnel beaucoup moins chère que des lentilles ZnSe de grandes dimensions pour de l'optique utilisée dans l'infrarouge.

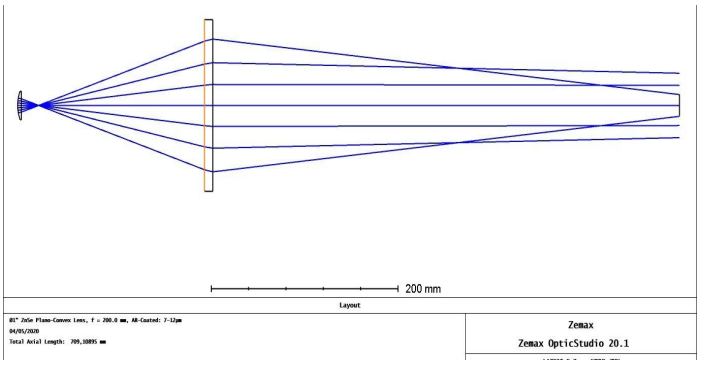
Lentille	Focale (mm)	Diamètre (mm)	Matériaux	Prix
Lentille Plan-Convexe	12.7	12.7	ZnSe	168 euros
Lentille de Fresnel	177.8	76.2	Acrylique	29 euros

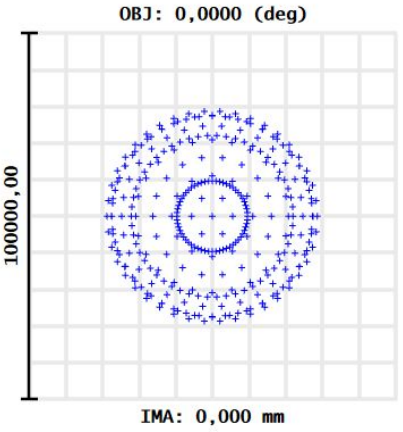
On déduit que le grandissement global sera de $\frac{f'_2}{f'_1}$ vaudra 14. Le champ total théorique dans l'approximation de l'optique géométrique (distorsion de champ négligée) de $\frac{\Phi_2}{f'_2}$ soit 24°. Si on reprend le champ de vue total de la Kinect présentée ci-dessous qui est de l'ordre de 41°. En conclusion, on peut s'attendre au mieux à un champ de fonctionnement du dispositif de 24° dans les deux directions.



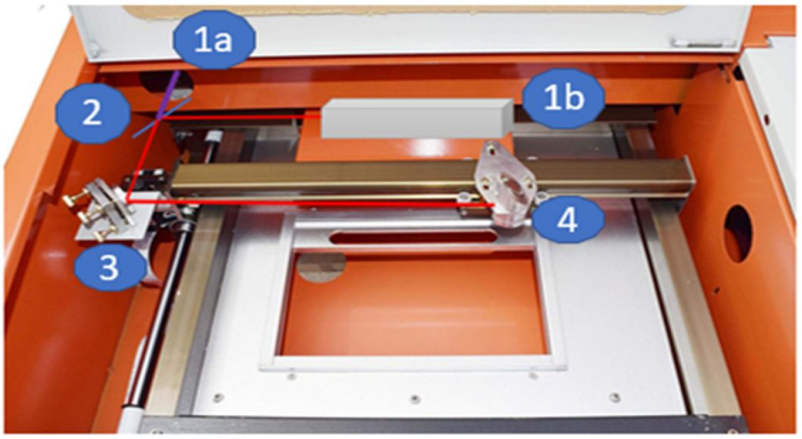
La modélisation optique

En Protis, nous avons commencé à simuler uniquement la partie orientation et et agrandissement du faisceau à 633 nm sur le logiciel de conception optique Zemax. Nous avons obtenu un faisceau en sortie de l'ordre de 43 mm de diamètre avec assez peu d'aberrations sphérique sur l'axe (ou du moins pas très visible en supposant un faisceau d'entrée de 3 mm de diamètre. Néanmoins, il aurait fallu faire l'étude dans l'infrarouge (ce qui n'était pas possible car les informations sur l'acrylique à 10 microns n'étaient pas disponible sur Zemax)





2. Détournement de la découpe et Sécurité Laser



1a Faisceau IR
1b Diode laser rouge
2 Lame de germanium
3 Miroir
4 Miroir + Lame de ZnSe

\+

Lentille de Fresnel
qui collimate le tout

Le schéma ci-dessus est vraiment représentatif d'un travail réalisé. Nous n'avons malheureusement pas de photo mais nous avons vraiment implémenté en 1b une diode laser rouge et donc son système d'alimentation aussi. Pour une meilleur tenue , on a consolidé le support de la découpe avec une cale métallique. En 1a sort le faisceau CO2. Avec un travail minutieux et un peu de patience on a positionné un miroir de germanium et sa monture de telle sorte à l'aligner avec le miroir 3 sur la course du laser CO2 et de manière à combiner les deux faisceaux le plus parfaitement possible. Le laser CO2 étant invisible on laissait sur un support non inflammable et on détectait la trace de combustion. On ne le voit pas aussi sur la photo mais on a perforé le capot supérieur de la découpe laser pour venir y fixer notre lentille de Fresnel dans son support. Au final après des heures de bricolage et d'alignement on avait en sortie de la découpe laser, un faisceau collimaté rouge et qui chauffait dans la main .

Un petit point s'impose niveau sécurité laser .

Le laser CO2, fonctionnera en régime impulsionnel (moins de 10s d'exposition sur la main). En prenant 1,5 cm de diamètre la tache image sur la main, on peut monter à une puissance de 600mW. Or notre laser CO2 peut aller jusqu'à 50W. Il

est donc nécessaire de l'étalonner. La découpe est munie d'un potentiomètre et d'une fenêtre de contrôle sommaires: le potentiomètre n'est pas gradué, et l'aiguille nous donne une valeur d'intensité de courant.

Nous allons donc nous-même caractériser le dispositif. Nous récupérons la tension d'alimentation du laser CO2 en fonction de la position du potentiomètre. Et en même temps grâce à un puissancemètre placé en sortie on mesure la puissance du faisceau CO2 après passage dans toutes les optiques .

Les résultats sont présentés ci-dessus.

Tension de contrôle (V)	Puissance de sortie (W)
0	0,12
0,64	0,58
0 ,72	1,95
0,76	2,8
1,53	3,37

Pour ne pas dépasser le seuil de sécurité laser on doit se cantonner à 600mW en sortie et donc à 0.651V d'alimentation. Il faudra développer un système de boucle retour d'interruption immédiate de l'alimentation en cas de dépassement de cette valeur.

En ce qui concerne la diode rouge, c'est une diode à 150mW. En prenant compte que la lentille de ZnSe et la lentille de Fresnel ne laissent passer que 30% du faisceau rouge , et pour concilier sécurité laser et bonne visibilité , on peut aller jusqu'à une alimention de 1,5 V pour la diode rouge ce qui correspond à moins de 3mW en sortie.

3. Logiciel d'exploitation de LightTouch

Dans cette partie, nous allons nous intéresser au logiciel qui permet de faire fonctionner le projet LightTouch. Il centralise à la fois la partie **"chaîne de traitement de l'information"** et la partie **" Chaîne de sécurité"** en un seul et unique logiciel codé en C#. Ce langage de programmation a été choisi dans la mesure où il est particulièrement bien adapté pour communiquer avec la Kinect et qu'il permet également de créer des interfaces graphiques pour communiquer entre l'Homme et la machine.

Généralités sur le langage C# sur Visual studio

Le langage C# est un **langage orienté** objet qui permet de définir une classe "form" qui contient de nombreux outils pour **la conception d'interface graphique** (boutons, textes, fenêtre, messages pop up…). Visual Studio est le logiciel qui permet d'écrire en C#, de débugger le code. Il permet aussi **la création de *set up*** pour installer le logiciel fini. Il est ainsi facile de passer le logiciel l'interface graphique d'un ordinateur à un autre, une fois le projet fini (sans même avoir à télécharger Visual Studio). Le code de la page principale du logiciel du projet est mis en annexe.

Visuel de la page principale notre interface graphique.

Interface graphique : chaîne de sécurité

L'interface graphique sert principalement à autoriser ou non les lasers à s'allumer. Lorsque l'on clique sur le bouton "Go!", le logiciel envoie un caractère "1" sur le port

série. La Nucléo récupère ces deux bits d'information et en déduit qu'il faut envoyer une tension au relais SPDT (Cf deuxième annexe). Le relais relie alors les deux lasers rouges et CO2 à leur tension de commande. (cf schéma électrique.)
Lorsque l'on clique sur le bouton "STOP", le logiciel envoie un caractère "0" sur le port série. La nucléo récupère ces deux bits d'information et en déduit qu'il ne faut plus envoyer une tension au relais SPDT (Cf deuxième annexe). Le relais relie alors les tensions de commande des deux lasers rouges et CO2 à la masse. Les lasers sont alors éteints.

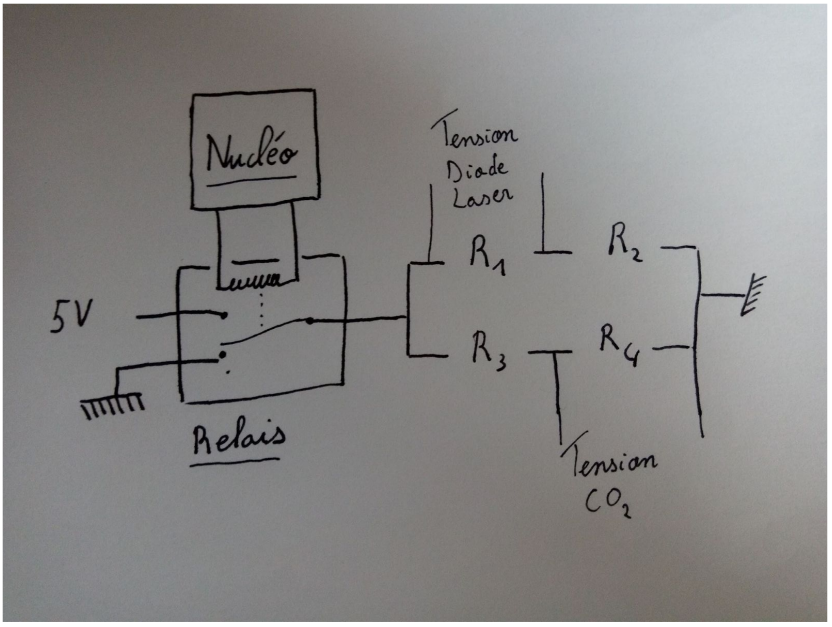


Schéma du circuit de sécurité

Tache de fond : Chaîne de traitement de l'information

Le logiciel permet également de faire le lien entre la kinect et les moteurs pas à pas. La bibliothèque microsoft.kinect permet en effet de récupérer l'image de la caméra de la kinect et de récupérer l'information sur la position de la main. Cette information sur la position de la main est alors transformée en consigne envoyée au moteur pas à pas pour envoyer les faisceaux lasers sur la main. Le code est en annexe 1.

Schéma de principe du logiciel

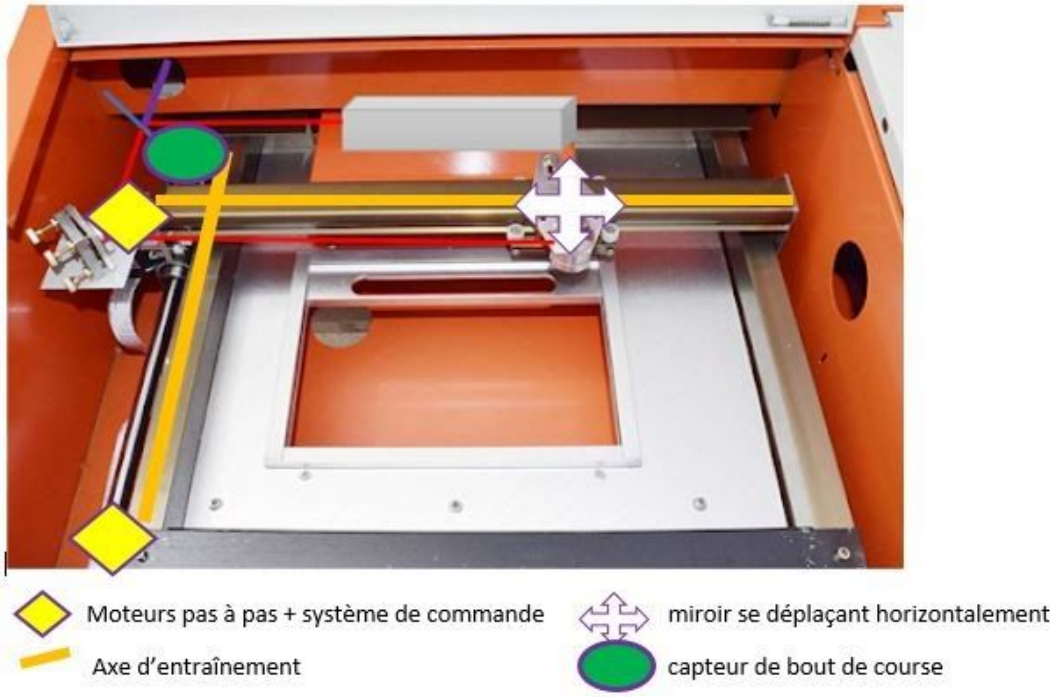
4. La partie motorisée

Dans cette partie, nous voudrions vous présenter la partie motorisée de notre projet. De quoi s'agit-il ?

Pour pouvoir suivre avec notre double faisceau la main du visiteur, nous avons besoin de déplacer nos miroirs. Comme nous l'avons expliqué dans la partie *Conception optique*, le faisceau traverse finalement une lentille de Fresnel. La position du faisceau sur la lentille de Fresnel permet d'envoyer le faisceau dans différentes directions. Il s'agit donc ici d'être capable de déplacer le faisceau en amont de cette lentille. En pratique, le dernier miroir du montage est placé sur un socle, que nous avons retourné pour que le faisceau sorte (au contraire de l'utilisation de la découpe laser). Ce socle se déplace horizontalement sur un cadre en métal grâce à deux moteurs pas à pas.

Dans le système initial, la commande des moteurs se fait par l'intermédiaire du logiciel graphique. Mais nous voulions pouvoir commander par nous même les moteurs. Pour cela, nous avons eu besoin d'une carte Nucléo branché à des amplificateurs de puissance et à des drivers de moteurs pas à pas. Nous avons fait une synthèse de nos recherche sur le pilotage du moteur pas à pas que l'on pourra trouver en annexe.

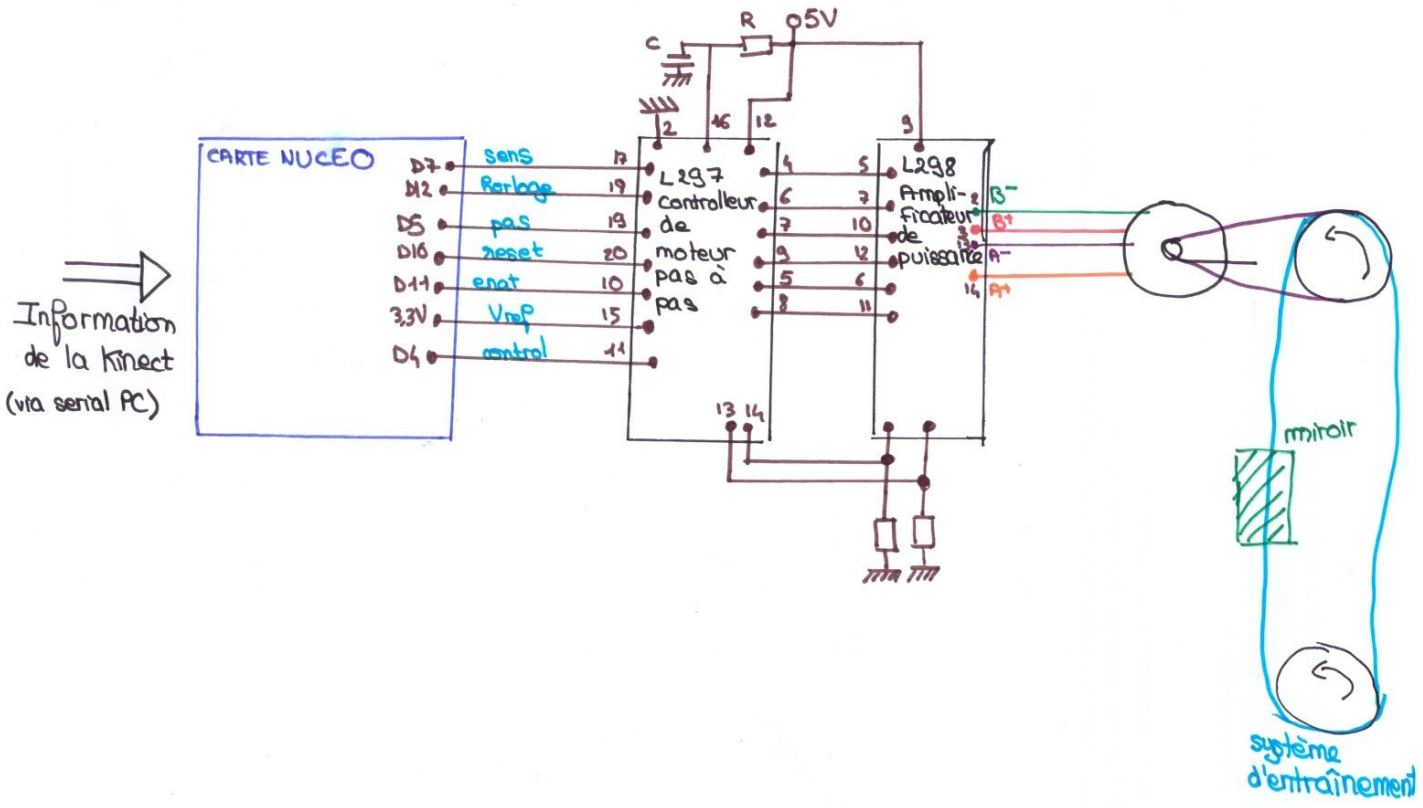
On est donc face à la situation décrite par la photo ci-dessous.



Eléments de motorisation de la découpe laser

Pour commander les moteurs, nous avons choisi d'utiliser une carte Nucléo reliée à un contrôleur de moteur pas à pas L297 et à un amplificateur de puissance L298. Grâce à l'assemblage de ces deux composants, on peut commander les moteurs directement en vitesse (pas ou demi-pas) et en position.

On réalise alors le circuit électrique décrit pour un moteur sur le schéma ci-dessous.



Circuit électrique pour un moteur

La carte Nucléo reçoit les informations de la Kinect comme nous l'avons décrit dans la partie *interface graphique*. Elle se sert de la position de la main pour contrôler le sens des moteurs, leur vitesse et le nombre.

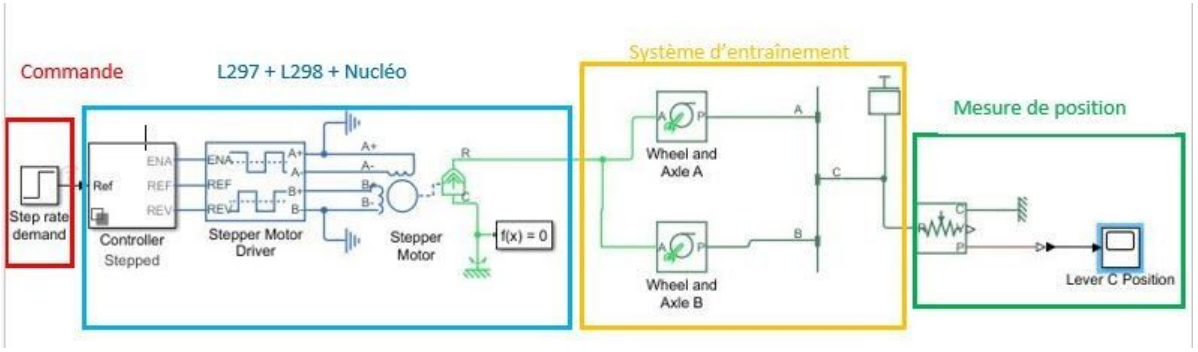
Les capteurs de bout de course sont déjà présents dans le système. Le LEnsE nous a fourni des adaptateurs pour pouvoir récupérer l'information sur la carte

Nucléo et donc initialiser notre système pour qu'au début de l'utilisation le moteur se trouve dans le coin en haut à gauche.

Simulation sous Simscape

Nous avons voulu modéliser la partie motorisation de notre système sous Simscape qui est un module de Matlab qui permet de réaliser des modèles multi-physiques.

On a donc réalisé le modèle ci-dessous :



Modèle sous Simscape du système de motorisation

La commande permet de choisir la position et la vitesse.
On utilise les outils de Simscape pour modéliser la carte L297 par le *Controller* en version dite *Stepped* et la carte L298 par le *Stepper motor Driver.* Simscape permet également d'insérer un moteur pas à pas dans le circuit.

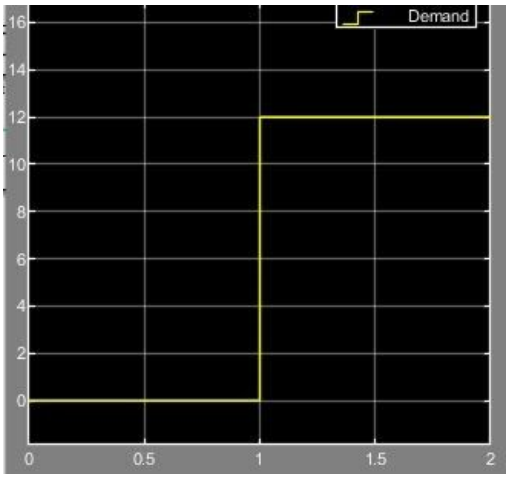
On choisit de modéliser les deux systèmes poulie-courroie par une système de deux roues entraînant un axe rigide. Le support du miroir est lui remplacé par une masse.

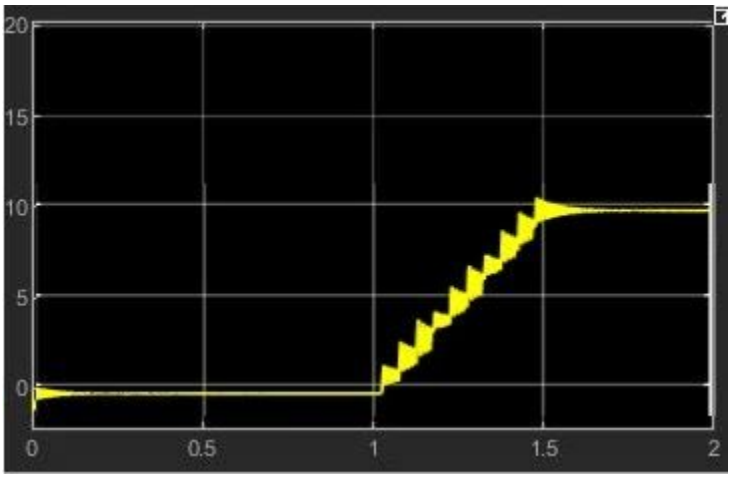
On mesure la position de la masse avec un capteur de position. En l'absence de possibilité de mesurer le poids réel de cet ensemble, on l'estime à 400g.

La principale difficulté de cette modélisation est d'estimer les différents paramètres du système. En effet, notre découpe laser ne possède pas de plan. Il faudrait donc idéalement caractériser chaque partie et regarder les références directement marquées sur les composants. Privée de cette possibilité, la modélisation doit venir nous alerter sur les différents paramètres à prendre en compte dans notre commande du système.

Pour cette simulation, on stimule notre moteur par un moteur pas à pas tel que un pas correspond à 2 degrés.

Par exemple, pour une commande échelon, on a la réponse suivante :





Commande échelon Réponse du système

On observe alors que la réponse du système n'est pas parfaite. Sous l'effet de différents paramètres et notamment du poids, le miroir présente une erreur statique. Il faudra tenir compte de ce facteur dans notre commande.

Commander le moteur :

La commande des moteurs se fait par l'intermédiaire de deux codes MBED: l'un associé au contrôleur L297 et l'autre commandant la séquence des moteurs pas à pas.

Le premier permet entre autres de régler les horloges des moteurs pour faire avancer la séquence vers le pas suivant et de déterminer le sens de fonctionnement des moteurs. Un exemple de code est fourni à la fin de l'annexe 3.

Le deuxième commande directement la séquence des moteurs pas à pas, c'est-à-dire l'activation successive des bobines des moteurs pour faire un nombre de pas donné. Ce code est présenté en annexe 4.

Où en sommes nous ?

Pour résumer l'aboutissement de ce projet, nous pouvons dire que la partie optique a été installé sur notre découpe laser et nous pouvions voir et sentir le double faisceau dans la main.
Nous avions commencé à réfléchir à l'installation dans l'espace de notre projet en investissant la salle Art et Science.
Nous avions également testé les moteurs mais sans le code final permettant de faire le lien avec la Kinect. Et nous avions vérifié que la Kinect était capable de détecter et suivre notre main dans un espace suffisant. Nous avons maintenant le code permettant de faire le lien entre les deux et donc de transmettre les coordonnées de la main dans l'espace.
Le confinement nous a amené à basculer d'un projet très pratique avec beaucoup de tests immédiats à un projet avec beaucoup plus de simulations. Nous avons donc modélisé notre système optique et notre système mécanique et réaliser une interface graphique.
Nous avons particulièrement insisté dans toutes les phases de notre projet sur la sécurité laser et nous avons notamment réalisé un circuit de sécurité permettant d'arrêter.

Comment poursuivre le projet ?

Nous avions identifié dans notre projet des phases prioritaires. Les circonstances ne nous ont pas permis de les réaliser.

Si nous devions poursuivre le projet, nous objectifs immédiats seraient :

- de placer les différents éléments de motorisation dans le support et les relier physiquement;
- de réaliser le cadre lumineux
- de réaliser la partie traitement et affichage vidéo prise par la webcam
- de faire fonctionner l'ensemble du système.

Conclusion

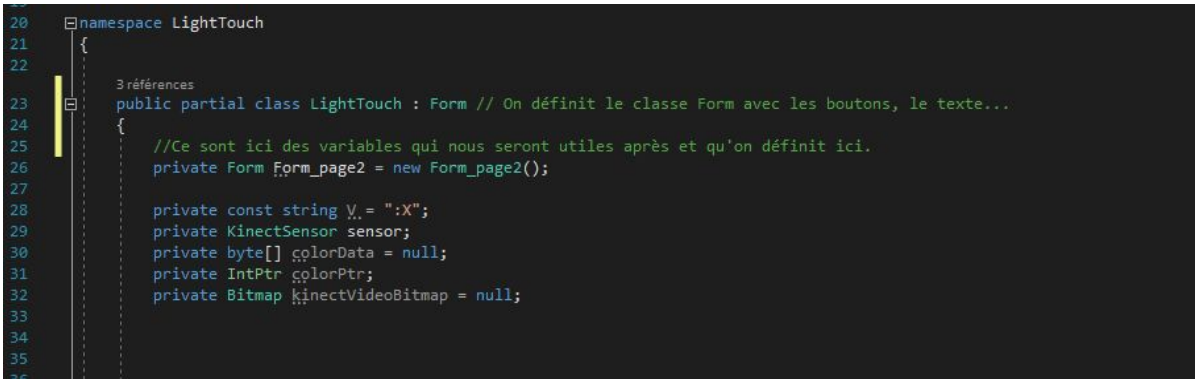
Avec ce projet, nous avons voulu montrer la beauté de la lumière. D'un projet artistique, d'un projet d'oeuvre, nous sommes passés à un projet technique, alliant optique, mécanique, informatique, bricolage, communication et gestion de groupe. Durant ce projet, nous nous sommes émerveillés, nous avons rêvé, nous avons peiné mais finalement nous avons appris.
Nous n'avons pas pu arriver au bout du chemin et vous présenter notre oeuvre terminé mais nous avons essayé autant se faire d'en construire chaque briques avec l'espoir de pouvoir un jour les assembler et enfin *Toucher la lumière*.

L'équipe Light Touch
Jad Aoun
Hippolyte Dupont
Hector Izard
Cécile Le Gall
Ewan Mer

Annexes

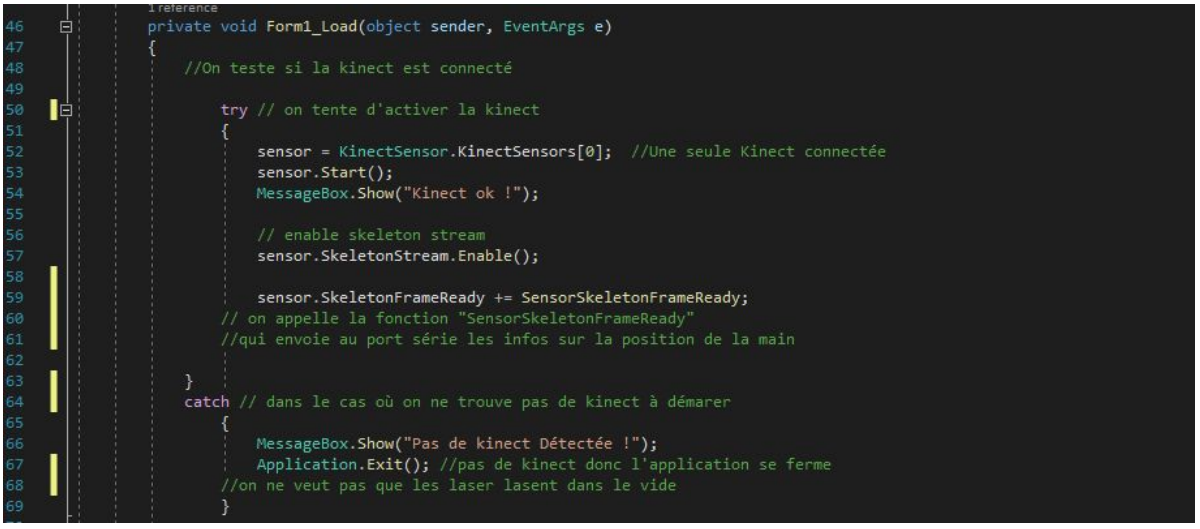
Annexe 1 : Le code de la page principale de l'interface graphique.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Threading.Tasks;
using System.Windows.Forms;

using System.IO.Compression;
using System.IO;

using System.Drawing.Imaging;
using System.Runtime.InteropServices;

using Microsoft.Kinect; // La bibliothèque qui nous permet de récupérer les infos de la kinect.
using System.IO.Ports;
using System.Threading;
```



```
namespace LightTouch
{
    3 références
    public partial class LightTouch : Form // On définit le classe Form avec les boutons, le texte...
    {
        //Ce sont ici des variables qui nous seront utiles après et qu'on définit ici.
        private Form Form_page2 = new Form_page2();

        private const string V = ":X";
        private KinectSensor sensor;
        private byte[] colorData = null;
        private IntPtr colorPtr;
        private Bitmap kinectVideoBitmap = null;
```

```
        1 référence
        public LightTouch()
        {
            InitializeComponent();
        }
```



```
        1 référence
        private void Form1_Load(object sender, EventArgs e)
        {
            //On teste si la kinect est connecté

                try // on tente d'activer la kinect
                {
                    sensor = KinectSensor.KinectSensors[0];  //Une seule Kinect connectée
                    sensor.Start();
                    MessageBox.Show("Kinect ok !");

                    // enable skeleton stream
                    sensor.SkeletonStream.Enable();

                    sensor.SkeletonFrameReady += SensorSkeletonFrameReady;
                // on appelle la fonction "SensorSkeletonFrameReady"
                //qui envoie au port série les infos sur la position de la main

            }
            catch // dans le cas où on ne trouve pas de kinect à démarer
                {
                    MessageBox.Show("Pas de kinect Détectée !");
                    Application.Exit(); //pas de kinect donc l'application se ferme
                //on ne veut pas que les laser lasent dans le vide
                }
```

```
73          //On code ici une fonction qui quand elle est appelée envoye la position de la main sur le port série
74
            1 référence
75          private void SensorSkeletonFrameReady(object sender, SkeletonFrameReadyEventArgs e)
76          {
77              Skeleton[] skeletons = new Skeleton[0];
78
79              using (SkeletonFrame skeletonFrame = e.OpenSkeletonFrame())
80              {
81                  if (skeletonFrame != null)
82                  {
83                      skeletons = new Skeleton[skeletonFrame.SkeletonArrayLength];
84                      skeletonFrame.CopySkeletonDataTo(skeletons);
85                  }
86
87                  foreach (Skeleton skeleton in skeletons)
88                  {
89                      if (skeleton.TrackingState == SkeletonTrackingState.Tracked)
90                      {
91                          Joint rightHand = skeleton.Joints[JointType.HandRight];
92                          float rightX = rightHand.Position.X;
93                          float rightY = rightHand.Position.Y;
94                          float rightZ = rightHand.Position.Z;
95
96                          if (serialPort2.IsOpen)
97                          {
98                              serialPort2.Close();
```

```
97                          {
98                              serialPort2.Close();
99                          }
100                         serialPort2.Open();
101
102                         serialPort2.WriteLine(String.Format("<{0}>", rightX));
103                         Thread.Sleep(2000);
104
105                         serialPort2.WriteLine(String.Format("<{0}>", rightY));
106                         Thread.Sleep(2000);
107
108
109                     }
110                 }
111             }
112         }
113
114
115
116
117
118
119
120
121
```

```
120
121
122
123         // SECURITE SECURITE SECURITE SECURITE SECURITE SECURITE
124
125
126             // c'est une fonction texte qui renvoie vers un autre code
            1 référence
127         private void texte_securité_Click(object sender, EventArgs e)
128         {
129
130         }
131
132
133         //il s'agit du bouton "Go!"
            1 référence
134         private void m_go_Click(object sender, EventArgs e)
135         {
136             // on dit à la nucléo de laser en envoyant "1" au port série
137             if(serialPort1.IsOpen)
138             { serialPort1.Close();
139             }
140             serialPort1.Open();
141
142
143             serialPort1.WriteLine("1");
144         }
```

```
145
146         // il s'agit du bouton "STOP"
            1 référence
147         private void m_stop_Click(object sender, EventArgs e)
148         {
149             // on dit à la nucléo de ne plus laser en envoyant "0" au port série
150
151             if (serialPort1.IsOpen)
152             {
153                 serialPort1.Close();
154             }
155             serialPort1.Open();
156
157             serialPort1.WriteLine("0");
158             Form_page2.ShowDialog();
159         }
160     }
161 }
162
```

Annexe 2 : Le code de la nucléo utile pour arrêter les lasers

```
1  #include "mbed.h"
2  AnalogIn analog_value(A0);
3  Serial pc(USBTX, USBRX);
4  char arret="0" ;
5
6  void rx_getData() {
7      arret= pc.getc()); // on récupère l'info sur le port série
8  }                      // avec cette fonction
9  int main() {
10     pc.baud(115200);
11     pc.attach(&rx_getData);// dès qu'une info est dispo sur le port série
12  // cette fonction permet de la récupérer
13     while(1) {
14         if(arret=="1")
15         {analog_out.write(1);
16         }
17
18         if(arret=="0")
19         {analog_out.write(0);
20         }
21     }
22 }
```

Annexe 3 : Comment piloter un moteur pas à pas

Objectif : Piloter un moteur pas à pas avec une carte Nucléo, un contrôleur de moteur pas à pas L297 et un driver L298N.

Qu'est-ce qu'un moteur pas à pas ?

Le moteur pas à pas est composé de bobines et d'un aimant (ou d'un morceau de fer blanc). Le principe est de faire tourner l'aimant en alimentant en courant (et donc en créant un champ magnétique) successivement les bobines.

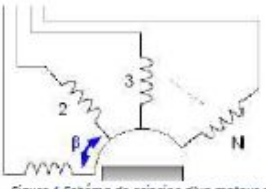


Figure 1 Schéma de principe d'un moteur pas à pas de type MRV source Wikipédia

Le sens de rotation dépend donc du sens d'alimentation des bobines. Si on considère quatre bobines, on a donc pour faire un tour la table de vérité suivante :

Dans le sens 1			
Bobine 1	Bobine 2	Bobine 3	Bobine 4
1	0	1	0
0	1	0	1
1	0	1	0
0	1	0	1

Dans le sens 2			
Bobine 1	Bobine 2	Bobine 3	Bobine 4
0	1	0	1
1	0	1	0
0	1	0	1
1	0	1	0

En multipliant le nombre de bobines, on multiplie le nombre de pas possibles.
On peut également faire varier le champ magnétique des bobines afin que le rotor se trouve à des positions intermédiaires et ainsi également multiplié le nombre de pas.

On trouvera une animation sur le fonctionnement du moteur pas à pas sur le site de l'Université du Mans : http://ressources.univ-lemans.fr/AccesLibre/UM/Pedago/physique/02/electri/pasapas.html

Il existe différentes technologies de moteur pas à pas :

- Les moteurs pas à pas à reluctance variable MRV
- Les moteurs pas à pas à aimants permanents MP
- Les moteurs pas à pas hybride MH

Les moteurs pas à pas à reluctances variables MRV :

Les moteurs pas à pas MRV possèdent n bobines et un rotor ferromagnétique sans aimant. Le rotor possède plus ou moins de dents et les plots statoriques (les plots portant les bobines) peuvent également en posséder. La modification du nombre de dents modifient les caractéristiques du moteur pas à pas et notamment son nombre de pas.

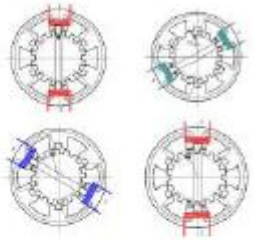


Figure 2 Schéma de principe d'un moteur pas à pas MRV source IAI

Par exemple, pour un moteur pas à pas MRV à plots statoriques non dentés, l'angle d'un pas vaudra: $\alpha = 2\pi\frac{N_{ps}-N_{dr}}{N_{ps}N_{dr}}$ avec N_{ps} le nombre de bobines et N_{dr} le nombre de dents du rotor.

Les moteurs pas à pas à aimants permanents MP :

Le rotor est constitué de plusieurs aimants permanents. Cette fois, un pas correspond à un angle : $\alpha = \frac{\pi}{pm}$ avec p le nombre d'aimants dans le rotor et m le nombre de phases c'est-à-dire le nombre de bobines pouvant être alimentées séparément.

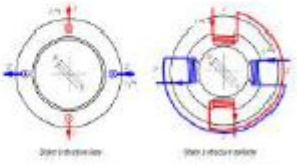


Figure 3 Moteur MP structure de base source IAI

Moteur pas à pas hybride MH :

Ce moteur combine les technologies des deux précédents en utilisant des aimants permanents dans un circuit ferromagnétique.

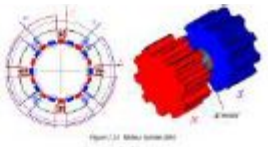


Figure 4 Moteur hybride source IAI

Comparaison des trois sortes de moteurs :

Le tableau suivant (issu d'un cours de l'IAI) compare les performances des trois types de moteur pas à pas.

Type de moteur pas à pas	*Moteur à réluctance variable*	*Moteur à aimants permanents*	*Moteur hybride*
Résolution (nb de pas par tour)	Bonne	Moyenne	Elevée
Couple moteur	Faible	Elevée	Elevée
Sens de rotation	Dépend : - *de l'ordre d'alimentation des phases*	Dépend : - *de l'ordre d'alimentation des phases* - *du sens du courant dans les bobines*	Dépend : - *de l'ordre d'alimentation des phases* - *du sens du courant dans les bobines*
Fréquence de travail	Grande	Faible	Grande
Puissance	Quelques Watts	Quelques dizaines de Watts	Quelques KWatts
Inconvénients	Pas de mémoire de position		

Le contrôleur de moteur pas à pas :

On utilisera ici un contrôleur L297 en sortie de la carte Nucléo. Il permet de commander un moteur pas à pas unipolaire ou bipolaire. Il permet de commander le moteur en direction (via la commande $CW/\overline{CCW}$) et en vitesse en choisissant de faire fonctionner le moteur en mode pas ou demi-pas (via la commande $HALF/\overline{FULL}$). Il faut également lui fournir y=une horloge (via l'entré CLOCK) pour faire avancer la séquence vers le prochain pas.

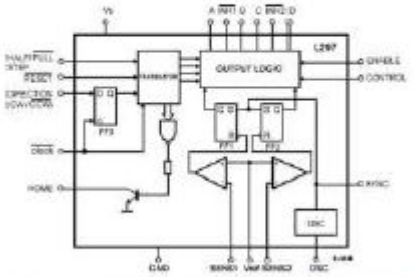


Figure 6 Décomposition en blocs fonctionnels du circuit L297 source Positron-libre

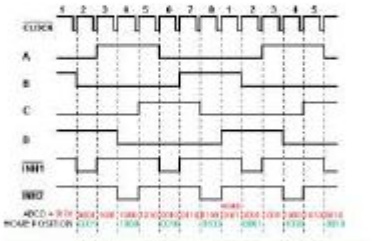


Figure 5 Chronogramme de commande de moteur pas à pas en mode demi_pas avec le L297 source Positron-libre

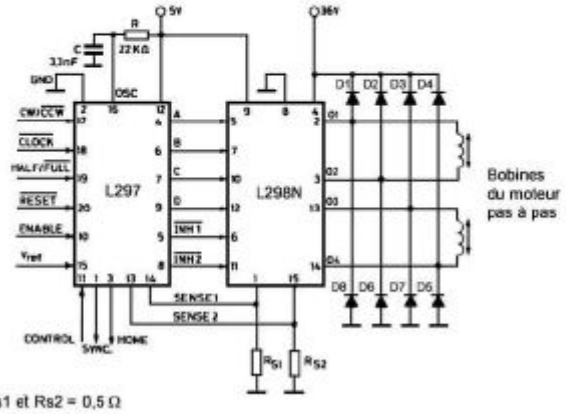
On utilise le L297 avec un étage de puissance (ou driver), soit le L298 (pour des courants jusqu'à 2A) soit le L293E (pour des courants jusqu'à 1A).

Driver L298 :

Il permettra si on le souhaite de faire tourner deux moteurs en même temps.

C'est un pont en H dont on trouvera l'explication du fonctionnement dans le tuto du LENSE *Comment faire varier la vitesse d'un moteur à courant continu ?.*

Pour relier le L297 et le L298, on utilise le schéma suivant :



Bobines du moteur pas à pas

Figure 7 Schéma d'une interface entre le L297 et le L298

Rs1 et Rs2 = 0,5 Ω
D1 à D8 = Diodes rapides 2 A

nt piloter un moteur pas à pas ?

Code MBED :

Pour piloter le moteur pas à pas via la Nucléo on utilisera par exemple, le code suiv
MBed :

```
// Programme pour tester le montage L293+L297
//avec un Nucléo et un seul moteur

#include "mbed.h"

DigitalOut Sens(D7);
PwmOut clock(D12);
DigitalOut pas(D5);
DigitalOut Reset(D10);
DigitalOut Enable(D11);
DigitalOut Control(D4);
DigitalOut led(D13); //LED de controle

int main() {

  clock.period_ms(10);
  clock.write(0.1);

  Control = 0;
  Reset = 1;
  Enable = 1;
  wait_ms(100);

  while(1){

    //Test : on commence par tourner dans le sens 1 avec un pas entier
    led = 1;
    Sens1 = 1;
    pas = 0;
    wait(1);

    //on tourne dans l'autre sens avec un pas demi entier
    led = 0;
    Sens1 = 0;
    pas = 1;
    wait(1);
    }
}
```

Référence :

Pour les moteurs pas à pas :

- Université du Mans :

http://ressources.univ-lemans.fr/AccesLibre/UM/Pedago/physique/02/electri/pasapas.html

- Institut d'Automatisation Industrielle et le cours de M.Correvon

https://www.chireux.fr/mp/cours/electromecanique/Chap07%20-%20Les%20moteurs%20pas%20a%20pas.pdf

- Wikipédia

Pour le contrôleur L297 :

- Datasheet :

https://www.alldatasheet.com/view.jsp?Searchword=L297

- Positron-libre :

https://www.positron-libre.com/electronique/apprendre/moteur-pas-a-pas/circuit-commande-l297.php

Pour le driver L298 :

- Datasheet :

https://www.alldatasheet.com/view.jsp?Searchword=L298

- Tuto du LENSE : Comment faire varier la vitesse d'un moteur à courant continu :

http://lense.institutoptique.fr/nucleo-faire-varier-la-vitesse-dun-moteur-a-courant-continu-3/

Annexe 4 : Code MBED pour la commande des moteurs

```
#include "mbed.h"

// initialisation des pins

DigitalOut pin_bobine_1_1X(D2); // chaque pin correspond à une des bobines du moteur pas à pas
DigitalOut pin_bobine_1_2X(D3);
DigitalOut pin_bobine_2_1X(D4);
DigitalOut pin_bobine_2_2X(D5);

DigitalOut pin_bobine_1_1Y(D6); // second moteur pas à pas
DigitalOut pin_bobine_1_2Y(D7);
DigitalOut pin_bobine_2_1Y(D8);
DigitalOut pin_bobine_2_2Y(D9);

float position_X = 0; // initialisation de la position des moteurs à 0
float position_Y = 0;

void reglage_pins_X(int pin1X, int pin2X, int pin3X, int pin4X)
{
    pin_bobine_1_1X = pin1X;
    pin_bobine_1_2X = pin2X;
    pin_bobine_2_1X = pin3X;
    pin_bobine_2_2X = pin4X;
}

void reglage_pins_Y(int pin1Y, int pin2Y, int pin3Y, int pin4Y)
{
    pin_bobine_1_1Y = pin1Y;
    pin_bobine_1_2Y = pin2Y;
    pin_bobine_2_1Y = pin3Y;
    pin_bobine_2_2Y = pin4Y;
}

void nombre_pas(float course_X, float course_Y float pas_X, float pas_Y) // détermination du nombre de pas
{
    nombre_de_pas_X = (int)(course_X-position_X)/pas_X;
    nombre_de_pas_Y = (int)(course_Y-position_Y)/pas_Y;
    position_X = course_X;                                          // on réinitialise la position
    position_Y = course_Y;
}
```

```
void prochainStep_X(int numero) // activation successive des bobines pour faire un pas (moteur "X")
{
    if (numero == 0) {
        reglage_pins_X(1, 0, 1, 0);
    }
    if (numero == 1) {
        reglage_pins_X(0, 1, 1, 0);
    }
    if (numero == 2) {
        reglage_pins_X(0, 1, 0, 1);
    }
    if (numero == 3) {
        reglage_pins_X(1, 0, 0, 1);
    }
}

void prochainStep_Y(int numero) // idem pour le moteur "Y"
{
    if (numero == 0) {
        reglage_pins_Y(1, 0, 1, 0);
    }
    if (numero == 1) {
        reglage_pins_Y(0, 1, 1, 0);
    }
    if (numero == 2) {
        reglage_pins_Y(0, 1, 0, 1);
    }
    if (numero == 3) {
        reglage_pins_Y(1, 0, 0, 1);
    }
}

void marche_avant_X(float attente, int nombre_de_pas_X) // commande de marche avant selon l'axe X
{
    for (int i=0; i <= nombre_de_pas_X; i++) {
        prochainStep_X(i % 4);
        wait(attente);
    }
}

void marche_arriere_X(float attente, int nombre_de_pas_X) // commande de marche arrière selon l'axe X
{
    for (int i=0; i <= nombre_de_pas_X; i++) {
        prochainStep_X(3 - (i % 4));
        wait(attente);
    }
}

void marche_avant_Y(float attente, int nombre_de_pas_Y)  // idem selon l'axe Y
{
    for (int i=0; i <= nombre_de_pas_Y; i++) {
        prochainStep_Y(i % 4);
        wait(attente);
    }
}

void marche_arriere_Y(float attente, int nombre_de_pas_Y)
{
    for (int i=0; i <= nombre_de_pas_Y; i++) {
        prochainStep_Y(3 - (i % 4));
        wait(attente);
    }
}

int main()
{
    while(1) {
// 200 pas en marche avant pour le moteur "X", rotation rapide
        marche_avant_X(0.01, 200);
        wait(0.1);
    }
}
```

Remerciements

Nous aimerions remercier toutes les personnes qui nous ont aidé à concevoir et réaliser une partie de ce projet et plus particulièrement :

Monsieur François Balembois qui nous a aidé à aller chercher l'inspiration au plus profond de nous même et à conjuguer art et sciences sans complexes et qui nous a donné sa confiance dans un projet qui partait d'une simple idée.

Monsieur Eric Michel qui nous a aidé à scénariser notre oeuvre d'art interactive et à mélanger les influences artistiques.

Monsieur Thierry Avignon qui a fait preuve de grande patience et qui nous a aidé à monter et détourner la machine laser tout en nous conseillant sur la dimension optomécanique.

Monsieur Cédric Lejeune qui nous a aidé à nous mettre en place et qui nous a donné la liberté de travailler au Lense.

Monsieur Julien Villemejane qui fut d'une aide précieuse pour la compréhension des moteurs de la découpe et de l'identification fastidieuse des pièces.