MASSELOT Pierre

JOUSLIN de Pisseloup de Noray Xavier

année 2018-2019

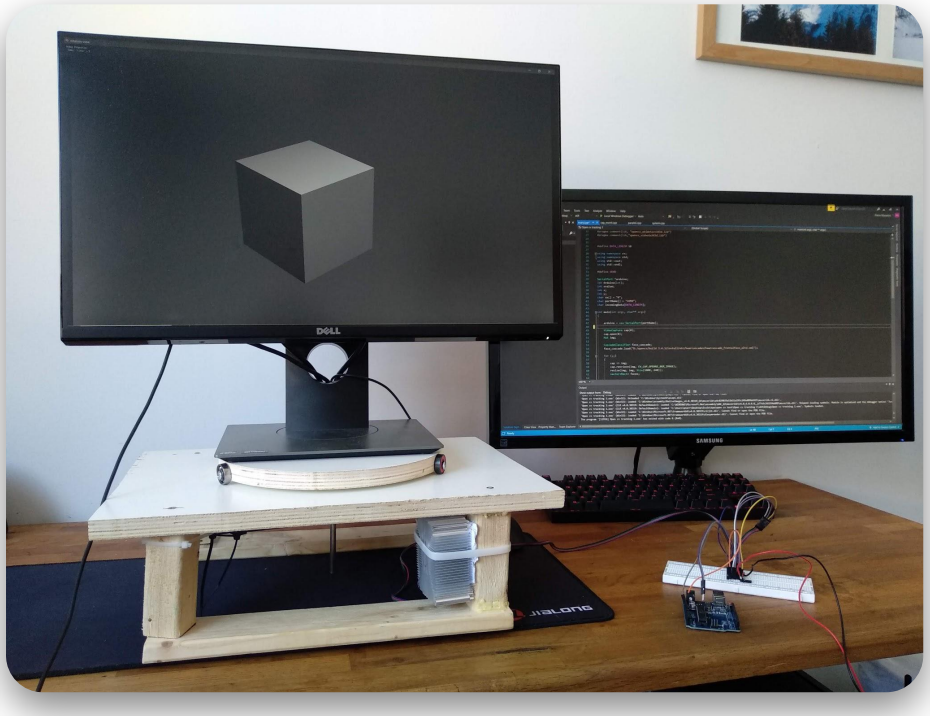
Comment faire percevoir un objet numérique en 3 dimensions dans notre réalité?

“Simulation d’une perspective cavalière en 3 dimensions sur un écran”

Thème : **l'aléatoire, l'insolite, le prévisible/3D**

Discipline dominante: Mathématiques

Autre discipline: Physiques



Sommaire

Préambule

Pendant l'été 2018, on avait envisagé de travailler sur un système permettant de visualiser des objets 3D. On a alors réfléchi à de nombreux moyens incluant l'utilisation de projecteurs et de panneaux réfléchissants pour créer une illusion en 3 dimensions. Bien entendu nous n'avons pas pu réaliser un tel système car cela aurait été bien trop coûteux. Avec l'arrivée des TPE, l'idée de la visualisation 3D s'est de nouveau imposée et on a commencé à faire de la recherche sur les technologies existantes pour finalement arriver à une idée de prototype peu coûteuse et facilement développable dans les délais impartis.

Depuis quelques années, de grandes entreprises, comme Google ou Microsoft essaient de simuler des objets 3D dans notre réalité sans nous couper de celle-ci, comme le prouvent les différents produits de ces deux compagnies : Google Glass pour la firme Google ou Hololens pour Microsoft.

La technologie qui se rapproche le plus de ce que l'on recherche est "l'hologramme". Cette technologie est cependant très coûteuse et n'est pas réellement en 3D, c'est seulement une illusion d'optique destinée à fonctionner dans un certain angle, c'est pourquoi elle est toujours en développement et reste principalement un élément de science fiction.

On cherche donc à faire apparaître un objet numérique en 3D dans notre réalité. Ce qui nous amène à la problématique suivante :

Comment faire percevoir un objet numérique en 3 dimension dans notre réalité?

On fera des recherches sur les technologies déjà existantes puis on réalisera un prototype fonctionnel pour répondre à cette problématique de telle manière que l'utilisateur n'ait pas besoin d'équipements additionnels.

Des vidéos de certaines étapes du projet sont disponibles à cette adresse ainsi que des time lapse de la réalisation de certains programmes:

https://pmasselot2003.wixsite.com/tpe20182019

I) Investigation

Réalité Virtuelle

La réalité virtuelle (en anglais, virtual reality ou VR) est une technologie qui permet de plonger une personne dans un monde artificiel créé numériquement. La réalité virtuelle permet de représenter des objets en 3 dimensions mais elle coupe l'utilisateur de la réalité car elle nécessite l'utilisation d'un casque qui empêche toute vision de l'extérieur. Notre but est de faire apparaître un objet en 3 dimensions dans la réalité, l'utilisation d'un casque est donc impossible. Le casque a également un autre désavantage, il ne permet pas à plusieurs personnes de voir l'objet représenté (à moins de posséder deux casques). On a quand même étudié le fonctionnement de ce casque dans le but d'avoir une meilleur compréhension des différents outils de représentation 3D, ce qui va se révéler utile par la suite

https://www.extremetech.com/wp-content/uploads/2015/10/OculusRift-672x371.jpg

Un casque de réalité virtuelle est composé de deux éléments principaux, le/les écrans et un ensemble de lentilles. Les écrans sont placés à quelques centimètres des yeux. Les lentilles divergentes permettent à l'utilisateur de voir les images comme si elles étaient à l'infini et donc permettent à l'utilisateur de ne pas avoir à accommoder ce qui rend l'expérience moins fatigante pour les yeux et permet au casque d'être relativement petit car la distance entre les yeux et l'écran peut être réduite à quelques centimètres . Cette technologie reste cependant très coûteuse et a des effets indésirables(nausée, perte d'équilibre, maux de têtes, etc...).

Réalité augmentée

La réalité augmentée, elle, ajoute des informations à ce que l'on voit déjà naturellement. Le principe est assez simple: analyser par un moyen (qui varie en fonction de l'utilisation) ce que l'on voit, puis superposer des informations additionnelles en fonction de la position et de la nature des objets analysés (ajouter de la couleur aux zones à risque, afficher le prix de certains produits,etc...). L'ajout d'informations se fait souvent à l'aide d'une paroi transparente et semi réfléchissante à travers laquelle l'utilisateur peut voir et sur laquelle un projecteur peut afficher des images.

https://chipsnwafers.electronicsforu.com/2018/09/14/augmented-reality-head-up-display/

Ce type de technologie est particulièrement adapté à certain milieux comme la conduite car la paroi transparente et semi réfléchissante est déjà présente (pare brise). La réalité augmentée a cependant ses limites. En effet, elle nécessite la plupart du temps de porter un équipement dédié comme des lunettes et elle ne permet pas de représenter des éléments en 3 dimensions pour plusieurs raisons. La première étant qu'on ne peut pas inclure une puissance de calcul suffisante dans ce type d'équipement et même si l'espace n'était pas un problème cela resterait extrêmement coûteux. La deuxième est le fait que les mouvements de l'utilisateur sont brusques et le moindre décalage dans l'affichage casserait l'illusion (on peut également remarquer que les mêmes désavantages apparaissent pour la réalité virtuelle).

Ce qu'on déduit de ces technologies

La réalité virtuelle montre qu'un écran est une bonne option pour représenter un objet 3D. On a alors cherché un moyen d'utiliser un moniteur d'ordinateur (que la majorité des personnes utilisent déjà avec leur ordinateur) pour afficher l'objet. **Pour cela on imagine faire tourner l'écran sur lui même tout en faisant tourner l'objet en 3 dimensions dans le sens inverse ce qui donnera l'impression que l'objet est fixe et que l'on se déplace autours de celui ci.**

A ce point dans l'avancée du TPE, il restait beaucoup de problèmes à résoudre notamment comment assurer une rotation de l'écran asservie à l'utilisateur.

Une deuxième technologie déjà existante nous a apporté la réponse: La réalité augmentée.

Bien que la réalité augmentée soit totalement incompatible avec l'affichage d'objet 3D, de nombreux procédés utilisés dans cette technologie sont utiles pour notre prototype. L'analyse de l'environnement est particulièrement intéressante. La plupart des objets qui utilisent la réalité augmentée analysent leur environnement à l'aide de différents capteurs. Ceux-ci peuvent être ultrasoniques, infrarouges mais c'est le capteur le plus simple qui nous intéresse dans ce cas : une caméra. En effet, une caméra est peu coûteuse et permet une grande flexibilité dans les objets analysés. Le développement d'un logiciel de traitement d'image est très difficile . Ainsi, on avait commencé par imaginer utiliser des logiciels d'analyse d'image existants. Par la suite, On a créé notre propre logiciel car ces logiciels n'étaient pas facilement utilisables (voir partie 3 codage du face tracking pour plus d'informations).

L'analyse de l'environnement extérieur est désormais réglée, on peut voir la position de l'utilisateur et faire tourner l'écran en fonction de cette position. Le face tracking est une méthode d'analyse d'image qui analyse certaines formes du visage dans une image et une fois ces images détectés, combine leurs positions pour avoir l'emplacement exact du visage de l'utilisateur. Cette analyse inclut les paramètres suivants : orientation (yaw, pitch, roll) et position relative par rapport à la caméra axes (x,y,z). Dans la suite du document, les axes que l'on utilisera seront uniquement les axes x et y.

On a maintenant une idée globale du résultat que l'on veut atteindre mais il reste encore un important travail à réaliser : programmer et construire un prototype qui utilise toutes les méthodes mentionnées précédemment… et ce dans les délais impartis. En général,dans l'industrie, des équipes de centaines de personnes travaillent pendant plusieurs années avant d'obtenir un produit commercialisable (le développement de l'oculus rift ,qui est un casque de réalité virtuelle, a commencé en 2012 et n'a été mis sur le marché qu'en 2016).

Notre but est seulement de créer un prototype fonctionnel (proof of concept , POC) mais c'est déjà une étape ambitieuse. Ainsi, l'organisation du projet était primordiale.

II) Organisation du projet

Répartition du travail

Suite à l'investigation, on s'est réparti les différentes tâches en fonction de nos compétences. On avait prévu de travailler pendant les heures de TPE. On s'est cependant assez vite aperçu que les ordinateurs de l'école manquaient de puissance pour réaliser ce genre d'activités.On ne pouvait donc pas programmer à l'école, on a donc décidé que l'ensemble de la programmation se ferait chez nous et que les heures de TPE seraient dédiées à la recherche.

Il fallait ensuite nous répartir la charge de travail. Le projet se sépare en deux partie, chacune avec un langage différent.

- L'une en C++ (programmation orientée objet) , qui permet le contrôle du hardware et le face tracking
- l'autre en Python est responsable de l'affichage en 3 dimensions et des interaction homme machine.
- Pierre spécialisé dans le langage C++ orienté objet, s'occupe de la première partie car celle-ci est dans un langage qu'il connaît déjà bien.
- Xavier à de l'expérience en web-coding et en python c'est pourquoi il s'occupera de l'interface graphique codée en python.

Les heures de TPE n'étaient donc pas utilisables pour de la programmation. Elles ont donc servi pour nous rassembler et échanger sur l'avancée du projet et sur les problèmes rencontré.

On savait depuis le début que ce projet nous demanderait un investissement personnel important: le projet ne s'arrête pas seulement au TPE , car il a été commencé avant celui-ci et continuera probablement l'année prochaine.

Décision sur l'environnement de travail

Le projet nécessite la création d'un prototype fonctionnel. Du matériel est donc requis. Pierre possède déjà la majorité du matériel nécessaire (imprimantes 3D, fer à souder, outil multifonction de type dremel, etc...). On s'est rassemblé certains vendredi après midi pour assembler le prototype.

Du point de vue software et logiciel , on utilise visual studio 2017 pour le programme en C++ (qui se charge du face tracking et de certains éléments de liaison entre les programmes). Visual studio est un compilateur qui supporte de nombreux langages de programmation et qui possède un débogueur très performant intégré. Cela permet de corriger des erreurs de programmation plus rapidement et permet de trouver la source des problèmes rencontrés. Visual studio est un outil de programmation reconnu partout dans le monde qui a été développé par Microsoft et qui est disponible gratuitement. Visual Studio permet aussi de convertir facilement des programmes faits pour être utilisés avec un certain système d'exploitation (windows 10 par exemple) vers un autre système d'exploitation. Dans le futur, cette flexibilité va nous permettre de faire fonctionner le prototype avec un Ipad ou une tablette android par exemple.

Pour contrôler la vitesse du moteur on utilise une carte Arduino qui est un microcontroller (voir réalisation du projet/codage du microcontrôleur). On programme l'Arduino en utilisant un IDE (integrated development environment ou environnement de développement en français).Ce qui nous permet de programmer, de compiler le code. L'IDE est fourni directement par les producteurs du microcontrôleur (l'Arduino) et permet aussi l'envoi du programme à celui-ci.

Pour l'interface graphique (affichage sur l'écran de l'objet 3D), on a décidé d'utiliser Blender. Blender est un logiciel de 3D qui inclut de nombreux outils. Parmi ces outils, il y en a un qui permet de créer une scène en 3 dimensions modifiable par un simple code en python. Il existe d'autres options que Blender et certaines sont plus simples d'utilisation et mieux optimisées, mais on a choisi Blender car c'est un logiciel open source (un logiciel open source est un logiciel gratuit et libre d'accès ce qui veut dire que le code de celui ci est disponible et peut être modifié).

D'autres logiciels ont été utilisés dans la réalisation de ce projet mais sont secondaires. Parmi ceux-ci, il y a Cmake qui est un outil qui permet de compiler des librairies (fonction pré-programmée permettant de réduire grandement le temps de développement. Cela permet en une ligne de faire tourner un code de plusieurs milliers de lignes et d'utiliser des fonctions créées par d'autres programmeurs). On peut également mentionner qu'une partie du programme a été réalisée avec QT même si ce programme s'est révélé inutile par la suite.

Liste des librairies utilisées: OpenCv (librairie open source utilisée pour le programme de face tracking), SerialPort (permet la communication entre le microcontrôleur et l'ordinateur), highgui, imgproc, stdio, objdetect343, videoio343,bge(blender game engine, permet l'affichage de l'objet 3D),ctypes (liaison entre c++ et python/ liaison entre le programme de face tracking et l'interface 3D)

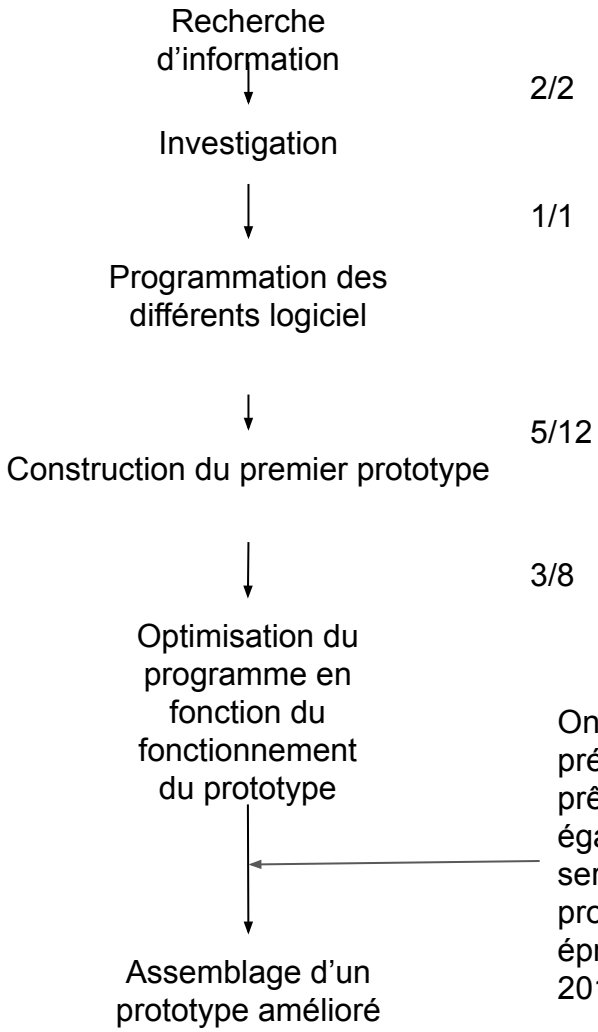
Planification

On avait planifié le projet assez rapidement, les dates d'échéance avait été fixées très rapidement et nous avions en quelques semaines une idée globale des différentes étapes à compléter avant d'arriver à un prototype fonctionnel et le temps que cela nous prendrait.

On c'est cependant rendu compte par la suite que ces dates étaient très optimistes étant donné les multiple problèmes rencontrés en cours de projet.

Schématisation du projet

Durée prévue/ durée réelle en semaines



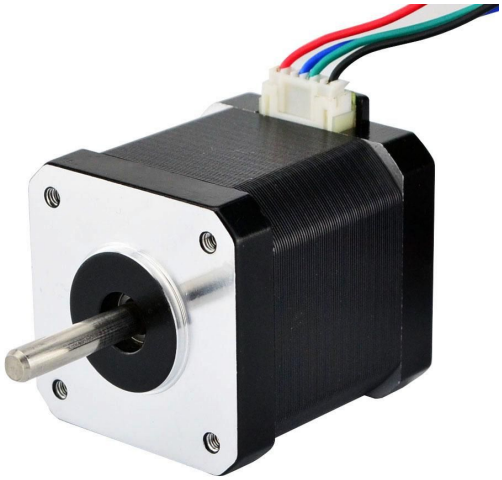
On est actuellement à cette étape et prévoyons d'avoir le second prototype prêt fin février 2019. On pense également qu'un troisième prototype sera créé avec des méthodes professionnelles mais après les épreuves de TPE, probablement en juin 2019.

III) Réalisation du projet

Construction du prototype

La première étape était d'acheter le moteur. Il existe de nombreux types de moteur. Il faut déjà choisir entre un moteur dc (direct current ou courant continu) ou ac (alternative current, courant alternatif). Le microcontrôleur est uniquement capable de contrôler les moteurs dc. Parmis les moteurs dc, on peut trouver une douzaine de sous catégories et divers modes de fonctionnement. Deux types de moteur conviennent à notre application, les moteurs pas à pas (moteur avec des positions très précises et qui, comme le nom l'indique se déplacent de pas en pas ce qui le rend peu fluide) et les servo moteurs.

https://traxxas.com/news/new-product/2075-metal-gears



https://bit.ly/2SCd5mG

Les servomoteurs peuvent avoir un couple et une puissance très élevés mais sont limités à un champ de rotation de 180 degré ce qui les empêche d'être une option viable pour notre projet.

Les moteurs pas à pas ont une précision légèrement inférieure (environ 200 pas par rotation soit une précision d'environ 2 degrés) mais permettent une rotation complète et ont une puissance suffisante pour notre projet. Les moteurs pas à pas ont cependant tendance à chauffer mais cela peut être résolu avec un simple système de dissipation de chaleur (on pense récupérer des blocs de refroidissement sur des ordinateurs usagés)

On a donc décidé d'utiliser un moteur pas à pas pour le prototype. Il fallait ensuite trouver la source de puissance requise pour le moteur. En effet de nombreux modèles sont disponibles variant de quelques dizaines à plusieurs centaines de dollars. Les imprimantes 3D utilisent déjà des moteurs pas à pas. On a donc un ordre d'idée quant à la puissance requise. Il faut cependant vérifier à l'aide d'un calcul de force si le moteur sélectionné a bien la puissance requise pour faire fonctionner notre prototype.

Le moteur que l'on pense utiliser a un couple de 0,17N/m. On cherche à trouver la masse maximale que ce moteur sera capable de faire tourner. Pour qu'il y ait un mouvement de rotation,qu'il faut qu'il y ait un couple de torsion (torque) pour produire un mouvement de rotation angulaire. Le couple nécessaire à cette rotation va dépendre de la masse de l'objet et de sa répartition dans l'espace (ce facteur ne sera pas pris totalement en compte. En effet, on considère que l'objet a faire tourner est un cercle. Cette hypothèse est partiellement vraie car la plateforme de rotation a cette forme.).

Ainsi , en prenant les paramètres suivants :

- α=acceleration angulaire (radius/s^2)
- I=moment d'inertie (kg.m^2)
- T=couple en N/m
- $T=I\times\alpha$

Or le mouvement d'inertie d'un disque est I=½×masse du disque en kg×(rayon du disque en mètre)².

On obtient $T=(\frac{1}{2}mr^2)\times\alpha$ m en kilogramme et r en mètre

Donc: $m=T/(\frac{1}{2}\times r^2\times\alpha)$

On a pris par défaut l'hypothèse que la base serait un disque de 12.5 cm de rayon. Donc on a fait l'expérience de tourner autour d'une chaise de 25 cm de diamètre. Après cette expérience, nous avons remarqué qu'il fallait 5 secondes pour tourner autour de la chaise (qui à une taille similaire au prototype que l'on veut construire) à une vitesse "normale" (marche car ce sera la vitesse qu'aura quelqu'un qui observe l'objet). La vitesse de rotation en radian par secondes est donc de $0,4x\Pi$.

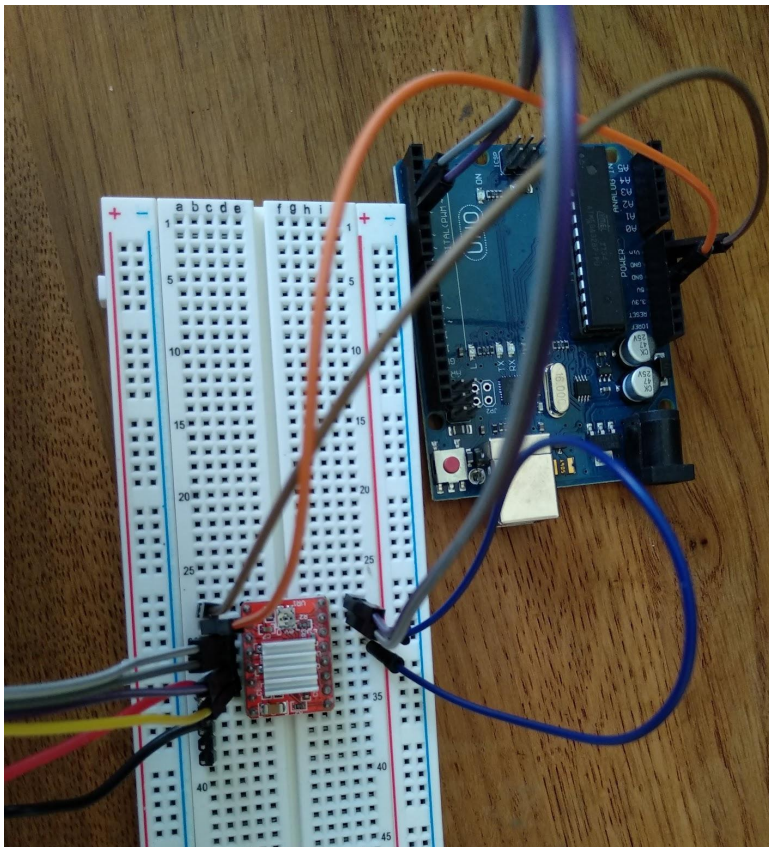
L'accélération angulaire d'un objet en rotation est le changement de vélocité angulaire en fonction du temps et s'exprime en radians/s²

- $\alpha=(w_2-w_1)/(t_2-t_1)$
- $\alpha=(0,4\Pi-0)/(0,25-0)$ on veut que la vitesse maximale soit atteinte en moins d'une seconde, un quart de seconde semble acceptable.
- $\alpha=5$
- Le couple du moteur est, d'après sa fiche technique: 26N/cm ce qui fait 0,26 N/m

- $m=0,26/(\frac{1}{2}\times0,125^2\times5)$
- m=6

On a donc une masse maximale de l'écran égale à 6 kg. On n'a pas pris en compte les frottements mais cette approximation est compensée par le fait qu'on a ajouté un engrenage 60 dents, ce qui donne un rapport ⅓ . Ainsi, sans frottement, la masse maximale de l'écran est de 18 kg. Ce résultat est très approximatif, mais nous permet de confirmer qu'avec un bon montage et quelque optimisations pour réduire le frottement, le moteur sélectionné est suffisant.
De plus, le premier élément limitant sera probablement la courroie (car elle est en charge de tout le transfert d'énergie du moteur à l'axe et sa tension est difficile à gérer avec les outils à disposition.)

Le moteur est donc sélectionné, il faut trouver un (pilote) driver. En effet, les moteurs pas à pas ne peuvent pas être contrôlés directement par la carte Arduino , il ont besoin d'un circuit qui simplifie les interactions entre ces deux composants, ce circuit est appelé driver. On utilisera un pilote de type A4988 qui est un petit driver facile a implémenter et peu coûteux.



On peut voir sur cette photo le driver en question situé au milieu de la carte de prototypage (breadboard) et de couleur rouge. Une breadboard est une plaque de prototypage qui permet de tester son circuit sans avoir besoin de recourir à de la soudure. Le driver a un potentiomètre intégré qu'il a fallu régler pour ne pas endommager le moteur (0,4 Ampères et 12 Volt).

Ce driver permet de contrôler des moteur pas à pas bipolaires qui sont les plus communs. En plus de permettre à l'Arduino de contrôler précisément le moteur, le pilote a aussi des fonctions intégrées comme une protection contre la surchauffe et contre les courts circuit. La partie logique du driver fonctionne avec des courants variant de 3,3V à 5V ce qui est exactement ce que l'Arduino fournit (5V). Cela rend ce driver parfaitement adapté à notre projet. Le driver pilote aussi la distribution du courant (12V fournis par un adaptateur) ce qui est très utile: en effet l'Arduino ne peut se charger de cette tâche car, comme dit précédemment celui ci fonctionne avec du 5V en sortie exclusivement

La carte de couleur bleue est l'Arduino. Un Arduino est un microcontrôleur open-source qui permet, à l'aide d'un algorithme, de lire des inputs (recevoir des données) et de les transformer en output. Ici l'input est la position du visage et l'output est la rotation du moteur.

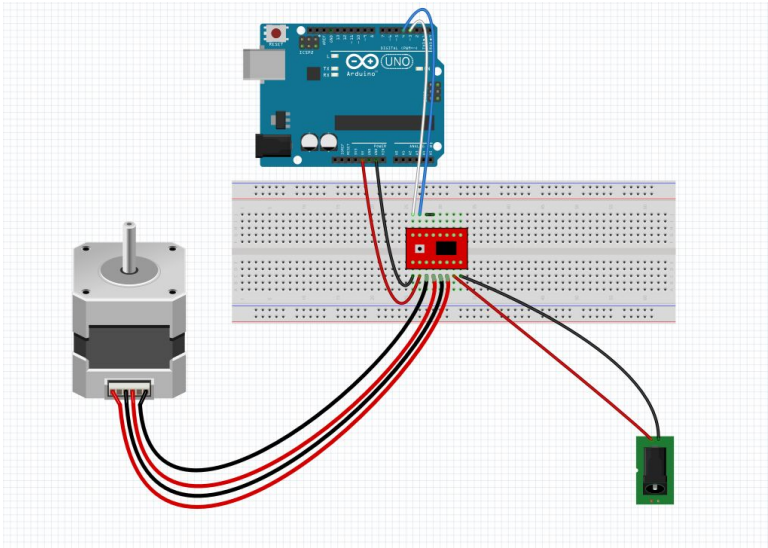


Schéma du circuit réalisé avec fritzing (logiciel de prototypage).La prise jack en bas à droite alimente le moteur et l'arduino est alimentée par un port USB connecté directement à l'ordinateur. On peut voir que deux ports du driver sont connectés entre eux. Ces deux ports sont responsables de la mise en sommeil du stepper (pour qu'il ne consomme pas d' énergie). Quand ils sont reliés entre eux, le moteur fonctionne continuellement. On pourra améliorer ce point dans le futur pour réduire la consommation, mais notre but étant d'avoir un prototype fonctionnel cela n'était donc pas nécessaire et cela aurait rendu la programmation de l'Arduino plus compliquée.

Assemblage du prototype

On découpe d'abord la base en forme circulaire à l'aide d'une scie sauteuse dans une planche en contreplaqué de 1,5 cm d' épaisseur.

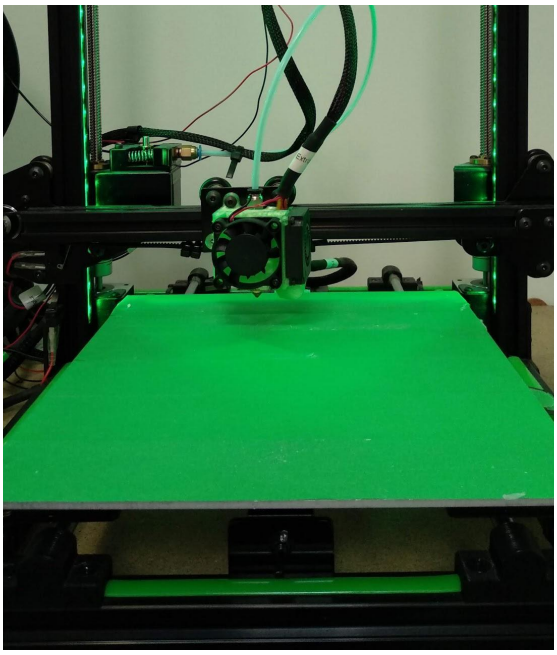
On a ensuite découpé quatre pieds à la bonne longueur.

Note : utilisation systématique des équipements de protection individuels adaptés (gants, lunettes...)

Assemblage du prototype

On a ensuite coupé l'axe de rotation à la bonne longueur avec un dremel (petite scie circulaire).

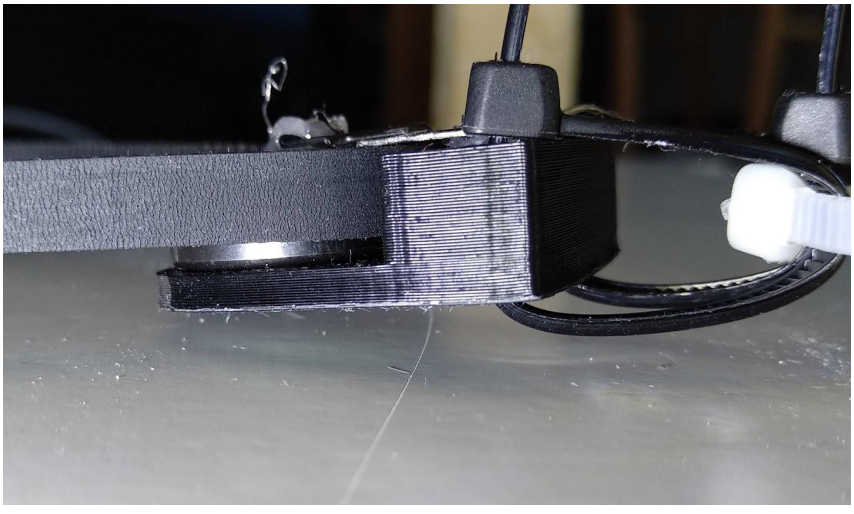
L'axe est issu d'une vieille imprimante jet d'encre qu'on a démontée.

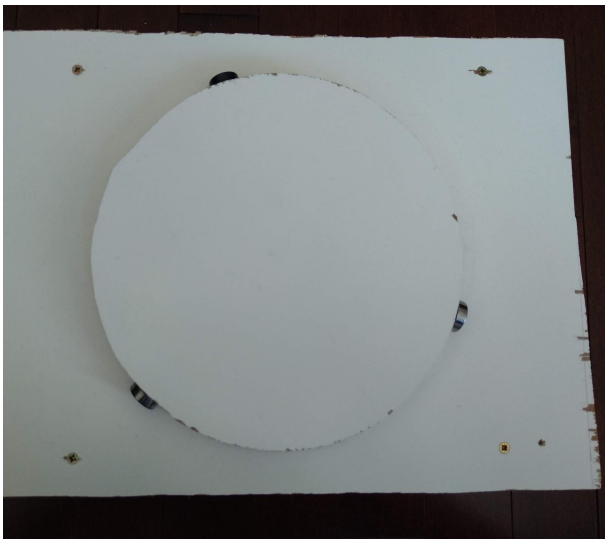


Finalement, on a désigné et imprimé en 3D un système de tension de la courroie visible ci-dessous.(imprimante 3D: Anet E10, épaisseur de couche: 0,2 mm , remplissage: 100%).

Le timelapse du design de cette pièce est situé sur le site

(https://pmasselot2003.wixsite.com/tpe20182019)



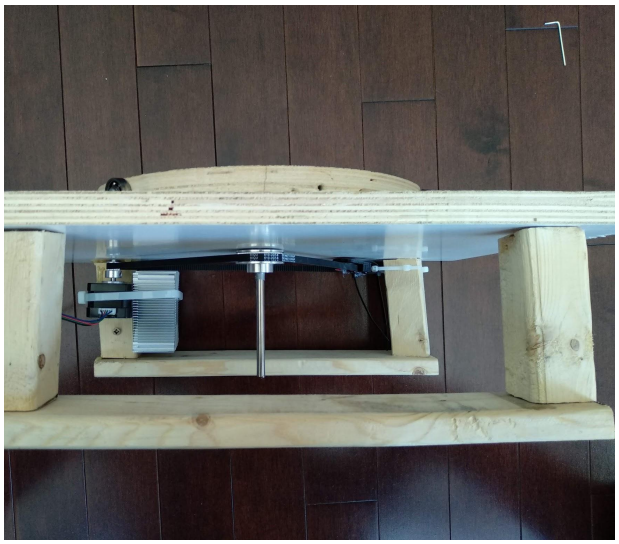


On a placé l’axe sur la base puis on y a ajouté des roulements à bille pour réduire la friction.

Cette base va être glissée dans la plateforme assemblée précédemment qui est constituée de quatre pieds et d’un plateau supérieur (comme une table). La planche supérieure est perforée et c’est à travers ce trou que l’on place l’axe.

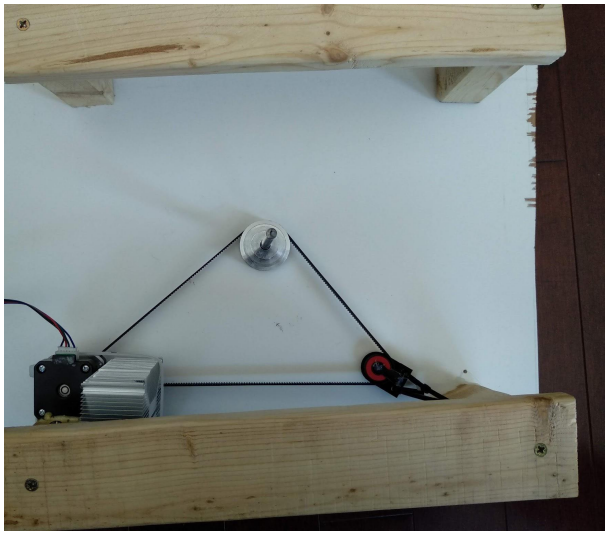
Les engrenages pour la courroie de distribution sont ensuite placés sur l’axe central.

Le moteur pas à pas est fixé sur un des pieds et le système de refroidissement y est attaché. La transmission thermique est assurée par de la patte thermique.



On place la courroie et on la tend à l’aide de la pièce imprimée en 3D qui est constituée d’un roulement à bille sur lequel la courroie va circuler et que l’on va éloigner au maximum avec des serres câbles (aussi appelé colliers). Une pièce métallique est fixée par dessus par la suite pour empêcher un décrochement de la courroie

L’écran sera placé sur la base circulaire et pourra tourner de gauche à droite librement. On s'est alors rendu compte que pour pouvoir faire plusieurs tours complets il fallait résoudre le problème des câbles. Deux câbles sont impliqués: le cable hdmi qui connecte l’écran à l'ordinateur et le câble d’alimentation de l’écran.



On a alors pensé à intégrer des cercles en acier dans la base et un point de contact entre ces cercles et une alimentation. Les cercles en acier seront alors constamment en contact avec l’alimentation pendant la rotation. On avait eu cette idée en observant la manière dont les voitures électriques sur les maquettes de circuit fonctionnaient. Après un peu plus de recherche, on s'est rendu compte que la méthode trouvée était déjà constamment utilisée en ingénierie et que le terme exact est collecteur à bague rotatif.

Pour la transmission de l’image on utilisera un appareil comme un chromecast qui permet de ne plus avoir besoin du cable HDMI. Cet appareil utilise le wifi pour transmettre le signal vidéo de l’ordinateur à l’écran.

Codage du face tracking… de l'intérêt d'une puissance de calcul suffisante

https://pmasselot2003.wixsite.com/tpe20182019

(timelapse des dernières heures avant l'aboutissement du programme)

Le face tracking est une technologie qui permet de traquer la position du visage. On cherche à avoir un objet qui ne nécessite pas de porter d'équipements. On a donc décidé d'utiliser une caméra et la reconnaissance faciale. C'est moins précis et plus difficile à programmer que d'autres méthodes mais les avantages sont majeurs et la précision est de toute façon faible étant donné que l'on utilise un moteur pas à pas.

Pour nous simplifier la tâche on avait initialement pensé à utiliser un logiciel déjà existant appelé facetracknoir(qui signifie face tracking sans led infrarouge et donc avec uniquement la caméra). Ce programme est utilisé dans le monde du jeu vidéo et d'après la description le fichier INI(fichier de stockage des données) était l'endroit où la position du visage est stocké.

On pensait donc accéder à ces informations en accédant au fichier INI.Le code du programme est accessible en ligne à cette adresse

(https://sourceforge.net/p/facetracknoir/codegit/ci/master/tree/).

On a réussi à ouvrir le fichier INI du logiciel de face tracking mais au lieu des coordonnées auxquelles on s'attendait il y avait seulement les réglages du tracker et pas la position. On devait alors décider si on construisait notre propre logiciel de toutes pièces ou si on continuait à chercher un moyen de se connecter avec un programme extérieur au logiciel déjà existant.

Le logiciel initial que l'on pensait utiliser(Facetracknoir) pour le face tracking est composé de centaines de programmes liés entre eux, chaque programme est composé de plusieurs milliers de lignes de code et chaque programme appelle lui-même des librairies (qui sont elles aussi composées de centaines de plus petits programmes).

L'analyse du code principal n'ayant rien donné,il aurait fallu analyser les codes secondaires ce qui était une tâche titanesque sans aboutissement assuré. **On a donc décidé de créer notre propre logiciel de face tracking**, ce qui n'est pas plus simple mais qui nous assurera un résultat et qui nous permettra d'apprendre beaucoup plus que si on avait seulement utilisé le programme déjà fonctionnel.

Codage du face tracking… de l'intérêt d'une puissance de calcul suffisante

https://pmasselot2003.wixsite.com/tpe20182019
(timelapse des dernières heures avant l'aboutissement du programme)

Le logiciel OpenCV est probablement la meilleur option pour créer notre logiciel de face tracking. OpenCV est une librairie open source (que l'on peut utiliser gratuitement et qui est accessible à tout le monde.).

Pour pouvoir utiliser OpenCV dans notre programme il a fallu d'abord l'intégrer à notre environnement de développement ce qui s'est révélé beaucoup plus difficile que prévu. Il a tout d'abord fallu télécharger les fichiers de la librairie puis, avec Cmake (logiciel spécialisé dans la compilation, dans notre cas de la compilation de librairie) créer une solution (ensemble de programmes) visual studio 2017.

Cmake crée cette solution et attribue automatiquement tout les bons réglages ce qui simplifie grandement la tâche. Il faut ensuite "build" (compiler et créer un fichier exécutable) de cette solution en debug et release (différents mode de compilation et de débogage, le premier permet de vérifier l'intégration et le bon fonctionnement de la librairie et le deuxième construit la librairie mais l'optimisme. En effet visual studio n'inclut alors aucune option de débogage).

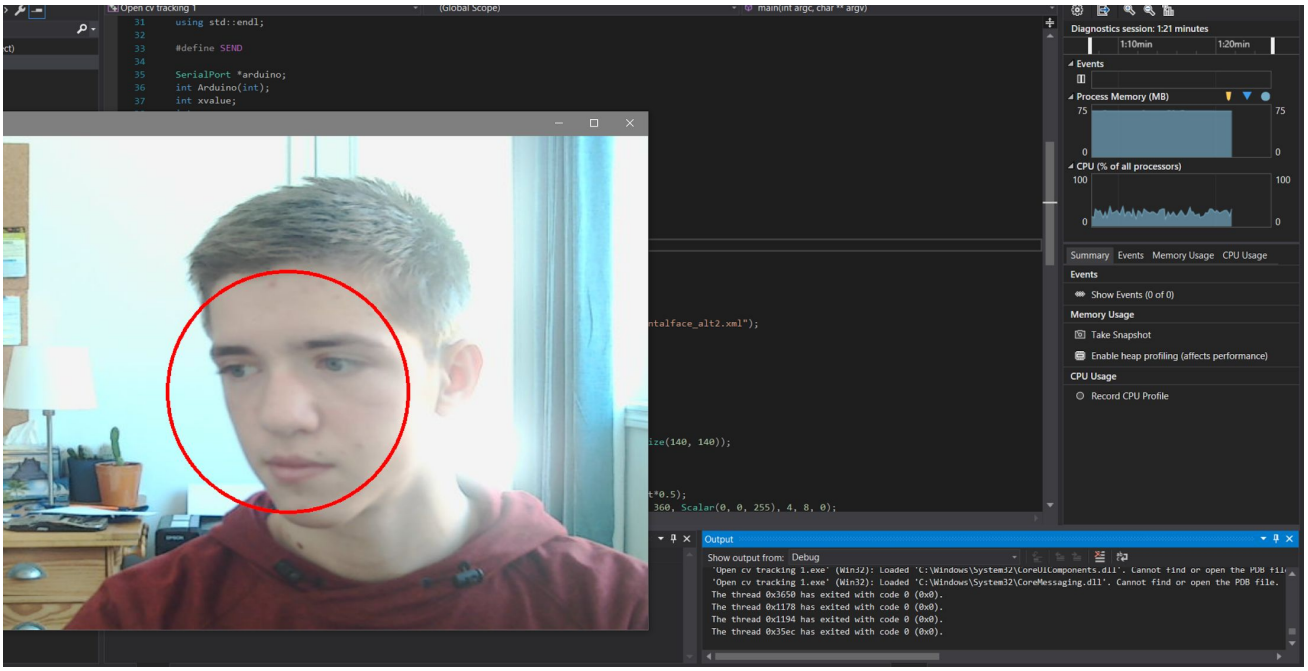
On a du refaire cette étape à de multiples reprises suite à des erreurs et c'est là que la puissance des ordinateurs est entrée en jeu. On a au total compilé 12 fois la librairie et chaque compilation avec cmake a pris une heure. La compilation du code visual studio généré à elle même pris une heure. Cela fait un total de 24h de calcul intensif !

En comparaison, si on avait travaillé avec des ordinateurs similaires à ceux disponibles à l' école cela aurait pris 6 fois plus de temps rendant la tâche quasi irréalisable..

On a alors commencé à coder le programme (celui-ci est présent à la fin du document). La structure est assez simple, on inclut les fichiers nécessaires et les librairies requises puis on déclare les variables notamment celle de la position du visage mais aussi certaines nécessaires au bon fonctionnement du programme.

On capture ensuite une image à l'aide de la caméra. On analyse cette image et on trouve la position du visage qu'on enregistre. Ce cycle est répété tellement rapidement que si on affiche toutes les images à la suite les unes des autres cela donne une vidéo, la fréquence de rafraîchissement des images n'étant pas visible à l'oeil nu ....

On affiche enfin cette vidéo pour nous permettre de vérifier si on est bien visible sur la caméra. Puis on affiche un rond rouge autour du point enregistré comme étant la position du visage. Le diamètre du cercle va varier en fonction de la taille du visage sur la caméra. Cela va nous permettre de bien vérifier le fonctionnement du face tracking.



Le logiciel de face tracking est assez gourmand en puissance de calcul du processeur.

Le programme analyse chaque image pixel par pixel et cherche des formes prédéfinies. Ces formes sont définies par un fichier que l'on a téléchargé . (le fichier s'appelle haarcascade_frontalface_alt2.xml) . Si un nombre suffisant de formes sont retrouvées le programme considère statistiquement un visage comme présent et reconnu par la position moyenne de ces formes.

On a rencontré plusieurs problèmes lors de la création de ce programme. Le premier était la déclaration des librairies, le deuxième était le fait que le programme considérait certaines formes comme étant des visages. Cela a été résolu avec l'addition d'un meilleur filtre au niveau de l'analyse des formes mais cet ajout indispensable a rendu l'application encore plus exigeante en ressource processeur.

La mauvaise intégration de la librairie était liée à de nombreux facteurs. Parmis ces facteurs les principaux étaient des réglages liés au compilateur lui-même et des déclaration de librairies qui devaient être spécifiées.

Un timelapse de l'ensemble de la programmation du face tracking est disponible sur (https://pmasselot2003.wixsite.com/tpe20182019)

Il correspond à près d'un mois de travail et plus de 30 heures de programmation.

Le logiciel doit aussi communiquer la position du visage avec les autre programmes (interface graphique et arduino). La communication entre le programme de face tracking et l'interface graphique sera traitée par la suite car particulièrement complexe.

La communication entre l'arduino est assuré par un port USB à l'aide d'une communication Serial. La transmission en Série est la transmission de données qui se succèdent bit par bit sur une voie de communication entre deux points. Dans notre cas les deux points sont l'arduino et l'ordinateur , la voie de transmission est le câble USB. Lors de la programmation de la communication Serial on a également rencontré des problèmes. Au début, on voyait que l'Arduino recevait des informations (les LED tx et rx (lumière qui indique la réception ou l'envoi de données) clignotent, mais la carte n'était cependant pas capable d'analyser les informations transmises.

On soupçonnait que le délimiteur("/n") qui arrête la lecture du string (chaîne de caractères reçus) ne fonctionnait pas bien et que l'Arduino n'analysait en réalité jamais les données reçues.L'hypothèse mentionnée précédemment s'est révélée fausse car l'arduino analyse les termes mais seulement quand ils sont envoyés avec un délai de 1 seconde. Cela veut dire que le processeur (équivalent du cerveau) de l'Arduino est trop lent pour analyser le message en moins de 50 millisecondes.

Or ce délai est trop long pour fonctionner avec le logiciel de face tracking qui nécessite au minimum une analyse toute les 50 millisecondes pour être suffisamment fluide (20 images par secondes, c'est encore un peu faible par comparaison avec le monde du cinéma oú la norme est de 24-25 images par seconde pour avoir suffisamment de fluidité).La limite d'envoi est donc soit du côté de l'Arduino soit du côté de Visual Studio (l'ordinateur).

Il est nettement plus probable que ce soit du côté de l'Arduino étant donné que sa puissance de calcul est bien plus faible que celle de l'ordinateur. On a par la suite fait de nombreuse optimisations du programme de l'Arduino.

La première de ces optimisations a été de définir une limite de caractère à la variable envoyée à l'Arduino (à la place de 255 caractères maximum seulement 10 ce qui est largement suffisant car on enverra au maximum deux variables (variant entre -999 et +999 ce qui fait 8 caractères, on garde deux caractères pour avoir plus de flexibilité dans le futur)).

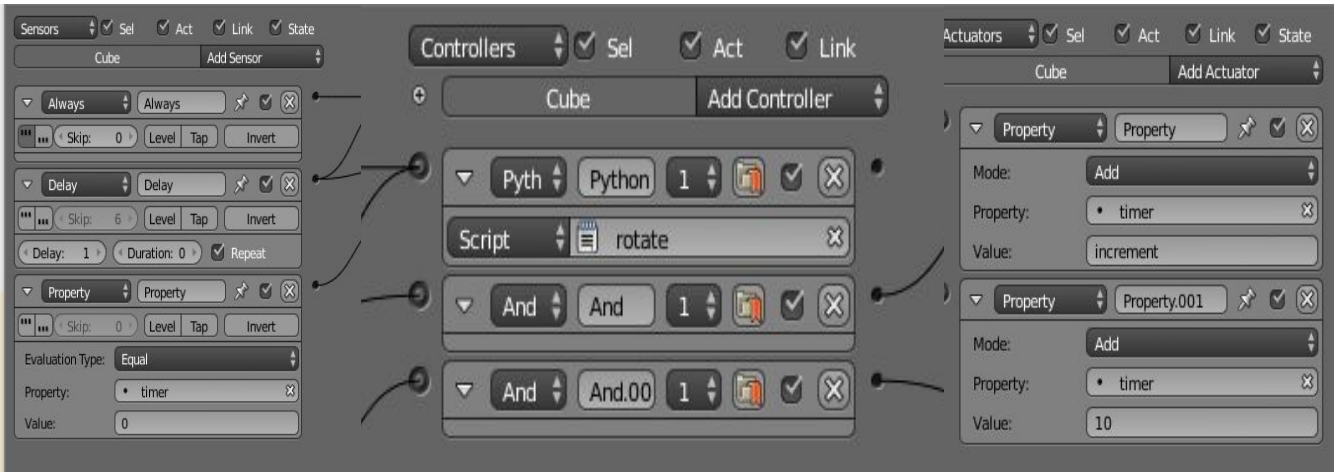
On a également changé une des fonctions de lecture des données reçues car cette fonction était connue pour être longue.

Codage de l’interface 3D

La 3D est une technique qui a été très vite appliquée dans le domaine du numérique. On peut donc retrouver de nombreux logiciels. Nous allons privilégier des logiciels libres, cela veut dire qu’il sont libres de droit et d’utilisation, que le code permettant de créer ces programmes est ouvert au public , nous donnant le droit de modifier et améliorer ces logiciels pour notre utilisation personnelle. Ce genre de logiciels est assez commun mais la plupart sont peu développés et leur documentation est restreinte. Blender est un de ces logiciels open source mais celui ci est bien documenté et facile à utiliser, de plus il est reconnu comme étant un des meilleurs outils de création 3D gratuit. Blender peut se programmer en python. On a utilisé Blender game engine qui permet de créer des scènes en 3 dimensions visibles par l’utilisateur.Il est utilisé dans la création de jeux vidéo et nous permet de contrôler un objet 3D dans l’espace à l’aide d’un programme assurant une rotation fluide et synchronisée.

Dans la programmation du logiciel, il a fallu indiquer les librairies pour que le logiciel reconnaisse et interprète les différentes formules utilisées dans le programme. Dans ce programme, on a utilisé la librairie bge pour blender game engine qui permet d’introduire des formules permettant de créer des actions dans blender et la librairie ctypes qui permet de faire un lien entre le langage python et le langage c++.

L’interface blender game engine est composée de deux partie: une partie sert à la programmation en python et une autre, intitulée “game logic”, partitionnée en 3 sous-catégories: les capteurs (tout ce qui est en rapport aux interactions entre la machine et l’utilisateur), les controllers(lien entre les deux parties de blender game engine qui permet de rassembler les différents éléments créés) et une dernière instance qui permet d'actualiser les variables que l’on a créées précédemment.



La partie game logic est un peu plus intuitive car elle consiste à assembler des blocs qui vont interagir avec le programme linéaire par la suite. On l'utilise par exemple pour créer un minuteur. Le minuteur est composé d’un délai qui va actualiser la valeur de la position du visage toute les 20 millisecondes. Une fois la partie game logic terminée, il fallait créer un lien entre celle-ci et le code ce qui est fait avec des déclarations dans le code lui même.

Après avoir maitrisé l’ensemble des interfaces, il a fallu programmer le code python. Python est un langage très commun utilisé par des millions de personnes, il y a donc de très nombreuse ressource sur internet qui permettent de résoudre les problèmes rencontré lors du codage de l’interface graphique.

Codage du microcontroller

La programmation de l'Arduino était la partie la plus simple car on avait déjà beaucoup d'expérience avec ce type de microcontroller.

On a cependant rencontré certains obstacles. On déjà mentionné précédemment les problèmes rencontrés lors de la connection série Serial et leurs solutions. Un obstacle majeur de l'Arduino était de réaliser plusieurs tâches en même temps.

L'Arduino n'est pas équipée pour faire plusieurs tâches complexe en parallèle, ce qui était un problème majeur étant donné que l'on devait analyser les données tout en faisant tourner le moteur et que l'interruption d'une de ces deux actions implique l'échec immédiat et inévitable de l'ensemble de la représentation 3D car la rotation n'est pas synchronisée.

On a donc implémenté une librairie qui permet à l'Arduino d'avoir la capacité de faire du multitâche. Cela fonctionnait correctement, mais on avait toujours un problème au niveau de la fluidité du processus. Le moteur n'atteignait pas la bonne vitesse et était très saccadé.

Cela a été résolu en remplaçant les fonctions delay par un argument (**if(currentMicro-previousMicro >= xrefreshrate)**). Ce changement permet de résoudre le problème car la fonction delay, assez simple à implémenter, arrête totalement l'arduino pendant la durée du delay : en d'autres termes, la rotation du moteur était interrompue à chaque fois que cette fonction était utilisée. La nouvelle fonction résout ce problème car elle permet à d'autres programmes de fonctionner jusqu'à ce que le temps (delay) imparti soit atteint et que le code suivant la fonction soit exécuté.

Quand on a voulu modifier la vitesse de rotation en fonction de la variable x , le moteur ne fonctionnait plus. La solution au problème était de changer le type de variable utilisé pour faire les calculs. Tout était défini en integer(valeur entière). Or le résultat de la division n'était pas entier ce qui causait des erreurs. La solution était donc de remplacer l'entier par une variable de type float (les variables de type float ne sont pas couramment utilisées dans la programmation car elles ralentissent les programmes c'est pourquoi on les utilise seulement quand c'est nécessaire).

Codage du dll

Il fallait trouver un moyen de transférer la position du visage entre le programme python dans blender et le programme de face tracking. On avait recherché pendant plusieurs semaines les différente solutions possibles mais aucune n'était convenable, soit car elles auraient mis trop de temps à être intégrées au programme soit car elles nécessitent une connection internet et un réglage d'IP très précis ce qui n'était pas une bonne option car cela aurait ajouté un délai de transmission et le prototype n'aurait été fonctionnel que chez nous , aurait mis longtemps à installer si on avait voulu l'utiliser dans une autre location.

L'idée du dll nous est alors venue car c'était un des éléments du code faisant partie de facetracknoir dont l'emplacement devait être spécifié pour que le programme en question reçoivent la position du visage. Un dll est un fichier qui peut interagir avec différents programmes. Ils sont la plupart du temps utilisés pour pouvoir appeler des fonctions complexes à partir de plusieurs programmes différents. Dans notre cas on pensait l'utiliser comme messager entre les deux programmes. Le code de face tracking envoie la valeur de x au dll et le code python demande au dll la valeur de x.

Cependant , cette idée n'avait pas fonctionné, le dll fonctionnait mais les données ne semblaient pas être modifiées pour l'autre programme comme si ce dernier utilisait une copie de ce dll. On a donc finalement abandonné l'idée d'utiliser le dll et repris nos recherches.

L'option initialement choisie était d'utiliser un wrapper, un programme qui permet de passer des fonctions et variables d'un langage de programmation à un autre. Nous l'avons rejetée rapidement car très difficile d'utilisation. Il semblait cependant que cela soit la seule option, la documentation est longue (600 pages) mais bien expliquée, il devrait donc être possible de créer le wrapper en utilisant l'outil SWIG dans les délais impartis.

En recherchant dans la documentation de SWIG, on a trouvé la mention de "SHARED DATA SEGMENT" dans un dll qui fait partie de SWIG. Affinant nos recherches, on a vu que cette fonction partage le dll entre les différents programme dans la mémoire vive.

La mémoire vive, aussi appelée RAM, est un moyen de stockage à court terme qui permet aux programmes de stocker et d'accéder très rapidement à différentes données.Une fois l'ordinateur éteint toutes les données stockées dans la mémoire vive sont détruites : c'est pourquoi il existe des moyen de stockage à long terme comme les disques durs, les clés usb,... Le dll est par défaut unique à chaque programme qui l'utilise : c'est pourquoi la variable ne fonctionnait pas entre le code c++ et python. Avec l'ajout de cette fonction on a finalement réussi à faire fonctionner le dll entre programmes

Conclusion

On a créé suite au projet 10171 lignes de codes, 10GB de données soit l'équivalent de 1073741824 bit (1 et 0) en langage machine. Plus de 300 heures ont été passées au développement du projet et le résultat répond à nos attentes et à été particulièrement intéressant à réaliser. On a vu à travers la réalisation d'un prototype fonctionnel que l'on pouvait représenter un objet en 3 dimensions dans notre réalité avec la combinaison de technologies déjà existante.

Notre prototype utilisait le face tracking pour repérer la position de l'utilisateur et orientait l' écran vers celui-ci. Un objet en 3 dimension était affiché sur l'écran et tournait sur l'écran de façon à rester fixe dans l'espace, comme dans la réalitée ce qui permet à l'utilisateur de se déplacer tout autours de cet objet.

https://pmasselot2003.wixsite.com/tpe20182019 (vidéo prototype fonctionnel)

Notre méthode d'affichage en 3 dimension a cependant ses limites, la première est le fait qu'elle nécessite l'utilisation d'un moteur et de partie amovible ce qui ajoute une légère imprécision et un potentiel point de défaillance.
On peut également remarquer que notre prototype ne possède qu'un axe de rotation, l'utilisateur peut tourner autour de l'objet mais ne pourrait pas regarder par dessus par exemple.
De multiple problèmes ont été rencontrés lors de la réalisation de ce projet, la plupart étaient reliés au code. L'un des éléments les plus importants qui était nécessaire à leur résolution était la patience et ténacité. La plupart de ces erreurs étaient résolues en quelques jours à l'aide de recherche internet et de l'observation détaillée de la documentation. Cependant certains problèmes ont été résolus après des mois d'efforts.... L'interaction entre le programme Python et le programme C++ était un de ces problèmes, on était tout d'abord peu familier avec les interactions interlanguages et les méthodes que l'on essayait était soit trop difficiles à mettre en place soit tout simplement non fonctionnelles et il nous restait très peu de temps pour conclure notre POC (proof of concept). Il a fallu faire preuve de patience ce qui n'est pas toujours simple quand on n'obtient aucun résultat après des dizaines d'heures de travail. On a réussi à continuer nos recherche et après des dizaines d'échecs la solution a été trouvée et s'est révélée être une légère modification apportée à notre premier essai d'intercommunication (Dll).

Par ailleurs, notre projet ne s'arrêtera pas à ce stade. En effet, on pourrait résoudre certaines des limites mentionnées précédemment en ajoutant un axe de rotation haut/bas par exemple. Un certain travail d'optimisation sera nécessaire pour réduire la puissance de calcul requise pour faire fonctionner le logiciel de face tracking. Ce point est très important car actuellement, notre prototype fonctionne avec un ordinateur puissant construit sur mesure avec un processeur 6 coeurs ce qui est bien supérieur aux ordinateurs que possède la majorité du public.

On a également d'autre idées d'amélioration de notre prototype, notamment un moyen de contrôle plus ergonomique qu'un clavier et une souris : on imagine pouvoir contrôler l'ordinateur à l'aide des mouvements de notre main en utilisant les mêmes méthodes utilisé pour la reconnaissance faciale déjà utilisée dans ce projet.

Resources utilisée pour le calcul de force:

Torque Formula (Moment of Inertia and Angular Acceleration).2005-2019,Softschools.
http://www.softschools.com/formulas/physics/torque_formula/59/
Angular Acceleration Formula.2005-2019,Softschools.
http://www.softschools.com/formulas/physics/angular_acceleration_formula/153/

Datasheet (fiche technique de composants qui nous permettent d'avoir les informations de base qui lui sont relié, les datasheet sont fournis par les manufactures et inclut au minimum les dimensions du produit et l'alimentation électrique requise):

Stepper motor.OSM techonology.
https://www.osmtec.com/downloads/stepper_motor/stepper_motor.pdf (moteur)

Images:

Dejan.How To Control a Stepper Motor with A4988 Driver and Arduino.MECHATRONICS .https://howtomechatronics.com/tutorials/arduino/how-to-control-stepper-motor-with-a4988-driver-and-arduino/

3d Nielsen.https://3dnielsen.dk/1109-large_default/pololu-a4988-1-stk.jpg

Ressources liées à la programmation et au problèmes rencontrés:

geeks' academy.Writing and loading a DLL with C++ and Python.4 janv. 2017 [7 décembre 2019].
https://www.youtube.com/watch?v=P-_kKCzqSn0

How can I use a DLL file from Python?, STACK OVERFLOW
https://stackoverflow.com/questions/252417/how-can-i-use-a-dll-file-from-python

SWIG-3.0.http://www.swig.org/Doc3.0/SWIGDocumentation.pdf
Code:

Global variables and linkage.LearnCpp.https://www.learncpp.com/cpp-tutorial/42-global-variables/

Transmission série.Wikipedia, 5 octobre 2018.https://fr.wikipedia.org/wiki/Transmission_s%C3%A9rie

Blender: How to rotate an object in Blender 2.74 using python script .STACK OVERFLOW.
https://blender.stackexchange.com/questions/43086/how-to-rotate-an-object-in-blender-2-74-using-python-script

Kris Occhipinti.BGE10 - Python in Blender Game Engine - Move then Rotate Object - Linux.28 mai 2012.https://www.youtube.com/watch?v=TdN67EPsgBc

Embedding Python in Another Application.Python Software Foundation.https://docs.python.org/2/extending/embedding.html
https://stackoverflow.com/questions/30435295/how-to-exchange-data-between-python-and-c

Kris Occhipinti.7 févr. 2012 [5 octobre 2018].BGE2 - Python in Blender Game Engine - Rotate Object - Linux.avi.https://www.youtube.com/watch?v=OdvpSllPHnQ
https://docs.blender.org/manual/en/dev/game_engine/python_api/index.html?highlight=python

L'ensemble des codes qui suivent représentent simplement une petite partie du programme, il permettent une compréhension assez basique du programme mais plusieurs centaines d’autre codes secondaires ont été nécessaires pour faire fonctionner notre prototype comme désiré.

Code de l'Arduino

```
#include <Arduino_FreeRTOS.h>
int ledPin =  13;
int ledState = LOW;
int stepperState =LOW;
unsigned long previousMicro = 0;
unsigned long previousMicro1 = 0;
long OnTime = 750;
long OffTime = 750;
long xrefreshrate=10000;
long delaystepper=10000;
int x=0;
float xpos=0;
int xt =0;
int xabs=0;
char data[10];
const int stepPin = 3;
const int dirPin = 4;
void TaskBlink( void *pvParameters );
void TaskAnalogRead( void *pvParameters
);
void setup() {
  Serial.begin(9600);
  Serial.begin(9600);
  pinMode(ledPin, OUTPUT);
  pinMode(stepPin,OUTPUT);
  pinMode(dirPin,OUTPUT);
  digitalWrite(dirPin,HIGH);
  while (!Serial) {
    ;
  }
  xTaskCreate(
    TaskBlink
    ,  (const portCHAR *)"Blink"
    ,  128
    ,  NULL
    ,  2
    ,  NULL );
  xTaskCreate(
    TaskAnalogRead
    ,  (const portCHAR *) "AnalogRead"
    ,  128
    ,  NULL
    ,  2
    ,  NULL );
}
void loop()
{

}
```

```
void TaskBlink(void *pvParameters)
{
  (void) pvParameters;
  for (;;)
  {
     unsigned long currentMicro = micros();
if(currentMicro-previousMicro1 >=
xrefreshrate){
previousMicro1 = currentMicro;
String input;
int i = 0;
while (i < 10)
{ while (!Serial.available()> 0) {}
 data[i++] = Serial.read();
 x = atol(data);
}

if (x<-10){
  digitalWrite(dirPin,LOW);}

if (x>10){
  digitalWrite(dirPin,HIGH);}
  }

  }
}

void TaskAnalogRead(void *pvParameters)
{
  (void) pvParameters;


  for (;;)
  { unsigned long currentMicro = micros();
  if(x>50||x<-50){
//xabs=abs(x);
//xpos=50000/xabs;
//delaystepper=long(xpos);
//xt=xpos*20;
//delaystepper=long(xt);
if((stepperState == HIGH) && (currentMicro -
previousMicro >= delaystepper))
  {
    stepperState = LOW;
    previousMicro = currentMicro;
    digitalWrite(stepPin, stepperState);
  }
  else if ((stepperState == LOW) &&
(currentMicro - previousMicro >=
delaystepper))
  {
    stepperState = HIGH;
    previousMicro = currentMicro;
    digitalWrite(stepPin, stepperState);
  }}
  }}
```

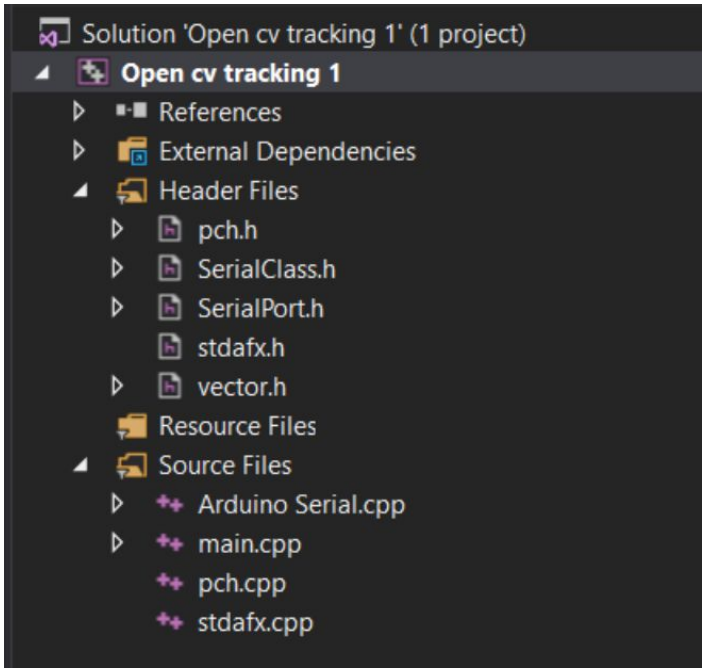
Code Face tracking

C’est seulement le code principal, il rassemble des centaines d’autres codes que l’on ne peut pas mettre au format papier

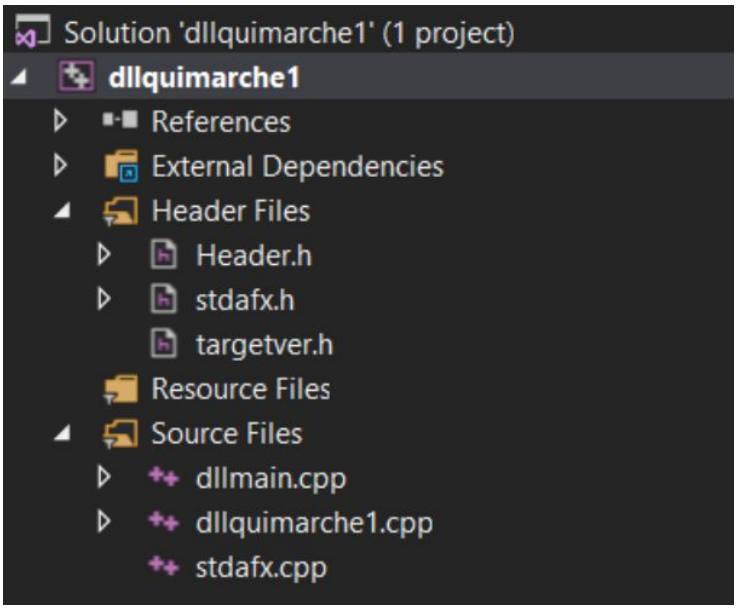
```
#include <stdio.h>
#include <stdlib.h>
#include <opencv2/core/core.hpp>
#include <opencv2/highgui/highgui.hpp>
#include "opencv2/imgproc.hpp"
#include "opencv2/objdetect.hpp"
#include <iostream>
#include <string.h>
#include "SerialPort.h"
#include <string>
#include <windows.h>
#include "Header.h"



#pragma comment (lib, "opencv_core343d.lib")
#pragma comment (lib, "opencv_highgui343d.lib")
#pragma comment (lib, "opencv_imgproc343d.lib")
#pragma comment (lib, "opencv_video343d.lib")
#pragma comment (lib, "opencv_imgproc343d.lib")
#pragma comment (lib, "opencv_imgcodecs343d.lib")
#pragma comment(lib, "opencv_objdetect343d.lib")
#pragma comment(lib,"opencv_videoio343d.lib")


#define DATA_LENGTH 10

using namespace cv;
using namespace std;
using std::cout;
using std::endl;

#define SEND

SerialPort *arduino;
int Arduino(int);
int xvalue;
int x;
int y;
char xx[] = "0";
char portName[] = "COM4";
char incomingData[DATA_LENGTH];
```

```
int main(int argc, char** argv)
{

	arduino = new SerialPort(portName);

	VideoCapture cap(0);
	cap.open(0);
	Mat img;

	CascadeClassifier face_cascade;
	face_cascade.load("D:/opencv/build
3.4.3/install/etc/haarcascades/haarcascade_frontalfa
ce_alt2.xml");

	for (;;)
	{
	cap >> img;
	cap.retrieve(img,
CV_CAP_OPENNI_BGR_IMAGE);
	resize(img, img, Size(1000, 640));
	vector<Rect> faces;

	face_cascade.detectMultiScale(img, faces,
1.1, 4, 64 | CV_HAAR_SCALE_IMAGE, Size(140,
140));

	for (int i = 0; i < faces.size(); i++)
	{
		Point center(faces[i].x +
faces[i].width*0.5, faces[i].y + faces[i].height*0.5);
		ellipse(img, center,
Size(faces[i].width*0.5, faces[i].height*0.5), 0, 0, 360,
Scalar(0, 0, 255), 4, 8, 0);

		x = faces[i].x - 320;
		y = faces[i].y - 160;

	}

	if (faces.size() == 0) {
		x = 0;
	}
	getx(x);
	Arduino(x);
	std::cout << xvalue<<"\n";

	int key2 = waitKey(20);
	imshow("facetracking", img);
	}
	return 0;
}
int Arduino(int x)
{
	snprintf(xx,10, "%d", x);
	if (arduino->isConnected()) {
	bool hasWritten =
arduino->writeSerialPort(xx, DATA_LENGTH);
	}
	return 0;
```

Solution du code principal qui contient le face tracking



Solution du Dll



Code du Dll

```
#include "stdafx.h"
#include <utility>
#include <limits.h>
#include "Header.h"
#pragma data_seg(".shared")

#pragma comment (linker,"/section:.shared,RWS")
__declspec(allocate(".shared")) int x = 0;

int getx(int a)
{
	x = a;
	return 0;
}
int returnx()
{
	return x;
}
```

Code Blender

```
import bge
from ctypes import*
cont=bge.logic.getCurrentController()
owner=bge.logic.getCurrentController().owner
own=cont.owner
mydll= cdll.LoadLibrary("C:\\Users\\pierr\\Desktop\Ecole\\tpe\\open cv test\\Open cv tracking
1\\x64\\Debug\\dllquimarche1.dll")
x=0
rotationspeed=0
timer=own['timer']
def main():
        x=mydll.returnx()
        print(x)
        if x>50:
      rotationspeed=0.005
        if x<-50:
      rotationspeed=-0.005
        if x<50 and x>-50:
      rotationspeed=0
        i=0
        while (i==0):
        if timer==0:
        if own["delay"] >= (own["start"] + own["break"]):
            own['p'] ==0.002

            own['start']=own['delay']
                print (own['p'] )
            owner.applyRotation([0,0,rotationspeed])
        else:
         owner.applyRotation([0,0,rotationspeed])
        i+=1
main()
```