

Etudiants

Rivo Andriamanalina
Lucas Huftier
Théo Couard

Encadrant & Intervenant

M. Sébastien Gauthier
M. Nicolas Viéville

Licence 3, Sciences et Technologie, Audiovisuel et Médias Numériques

2021

Projet Type 2 : Remote DSLR



Remerciements

Pour commencer, nous tenons à adresser nos remerciements à ces personnes sans qui ce projet n'aurait pas pu voir le jour :

-**M. Sébastien Gauthier** qui nous a encadrés depuis le début et jusqu'à la fin du projet et qui nous accompagnés malgré la difficulté et le temps que cela a requis. Il nous a mis à disposition les moyens de créer le prototype et tous les autres moyens pour mettre ce projet sur pieds. Il a su nous conseiller sur tous les aspects de notre projet. Pour tout cela, nous lui adressons toute notre reconnaissance

-**M. Nicolas Viéville** qui nous a énormément aidés sur les problématiques liées à l'environnement UNIX et à la programmation plus avancée en langage Python. Il a beaucoup contribué à l'amélioration de notre logiciel et il a répondu aux problèmes complexes sur lesquels on avait du mal. Un grand merci à lui.

-**M. Philippe Thomin** qui nous a aidés sur des choix techniques tels que l'intégration de nos images et de notre flux vidéo à l'interface graphique du logiciel. Il nous a aussi orientés vers le choix de concevoir le prototype pour du stopmotion, chose qui nous a aidés à établir le cahier des charges. Nous lui en remercions.

Nous tenons également à adresser toute notre reconnaissance à l'équipe pédagogique qui a su nous montrer un cadre pendant le pitch afin qu'on puisse se lancer dans un projet réalisable et ambitieux.

Glossaire

| | |
|--------------|---|
| DSLR | Digital Single-Lens Reflex |
| UNIX | Uniplexed Information and Computer Service |
| OS | Operating System (Système d'exploitation) |
| SUDO | Super User Do |
| DC | Direct Current (courant continu) |
| RPM | Rotations par minute |
| VNC | Virtual Network Computing |
| FIFO | First Input First Output |
| APN | Appareil Photo Numérique |
| IDE | Integrated Development Environment (Environnement de Développement Intégré) |
| MJPEG | Motion JPEG (Joint Photography Expert Group) |

Table des matières

| | |
|---|----|
| Introduction..... | 1 |
| I-Cahier des charges..... | 2 |
| II-Présentation du système mécanique..... | 3 |
| a) Système de translation..... | 3 |
| b) Le système de slider..... | 5 |
| c) Les mouvements panoramiques..... | 5 |
| d) Les mouvements de tilts..... | 6 |
| e) La carte Arduino-Uno : ORION (et Raspberry Pi)..... | 7 |
| f) Représentation 3D à l'échelle 1/10 du système sur Tinkercad..... | 8 |
| III-Interface graphique..... | 9 |
| a) Au démarrage..... | 9 |
| b) Présentation de l'interface utilisateur..... | 10 |
| c) Structuration du logiciel..... | 14 |
| IV-gPhoto2 sous Linux/UNIX..... | 15 |
| a) Installation de gPhoto2..... | 15 |
| b) Commandes essentielles de gPhoto2..... | 15 |
| c) Autres commandes essentielles de l'environnement Linux..... | 16 |
| c) Une question de modèle..... | 17 |
| V-Scripts Python : les fonctions de base..... | 18 |
| a) Une fenêtre et des widgets pour commencer..... | 18 |
| b) Un dictionnaire pour les valeurs à utiliser..... | 20 |
| c) le liveview..... | 21 |
| d) Créer, charger, sauvegarder..... | 23 |
| e) prendre une photo..... | 24 |
| f) paramètres..... | 25 |
| g) Contrôle de l'Arduino (côté PC)..... | 26 |
| h) interaction entre les fonctions..... | 28 |
| i) rendu..... | 29 |
| j) Threading..... | 29 |
| k) Autofocus..... | 30 |
| VI-Côté Arduino..... | 31 |
| Conclusion..... | 34 |
| Annexe..... | 35 |
| Matériel utilisé..... | 35 |
| Installation de Raspbian..... | 36 |
| Installation manuelle de Gphoto2 et de libgphoto2..... | 36 |
| Spécifications techniques ORION..... | 37 |
| Fiche de suivi..... | 40 |
| Webographie..... | 43 |

Introduction

Le groupe est constitué de trois étudiants qui ont l'habitude de collaborer ensemble et avec le même intérêt pour l'aspect technique et informatique du domaine Audiovisuel. Lorsque les enseignants nous ont décrit le but du projet Type 2, à savoir un projet qui peut-être très technique ou très artistique, on avait déjà pour idée de travailler sur un projet qui regrouperait les aspects informatique, électronique et artistique de la formation.

Les projets étaient une opportunité de mettre en œuvre tout ce qu'on a appris tout au long de la formation. Sachant que le Type 1 nous a permis de travailler sur les prises de vue et de son, le Type 2 nous a offert la liberté de montrer ce qu'on nous a enseigné dans les autres disciplines.

On a réfléchi à plusieurs sujets pour le Type 2 : conception d'un synthétiseur, un traducteur midi de n'importe quel son, un instrument de musique virtuel. Au final, notre choix s'est porté sur le contrôle à distance d'un appareil photo. On devrait avoir le contrôle total sur l'appareil via un ordinateur. C'est ainsi qu'est né le projet **Remote DSLR** que nous vous présentons sur ce document.

Dans ce document, vous trouverez une description du système électro-mécanique qui permet de faire tous les mouvement de caméra (translation, tilt et panorama), ensuite l'interface graphique du logiciel qui permet de contrôler l'ensemble (paramétrage de l'appareil et mouvement), nous aborderons l'adaptation à l'environnement Linux/UNIX (installation des dépendances) ainsi que l'exécutable gphoto2, et nous terminerons avec l'explication des codes du Logiciel.

Les parties en commentaires dans les explications des lignes de codes sont importantes pour bien comprendre la démarche adoptée au cours du projet.

I-Cahier des charges

Pour la promotion de leur nouveau set de jouets de la licence Dragon Ball, l'entreprise **Lego™ France** nous a contactés pour concevoir un système servant à faire stop motion en touchant le minimum l'appareil photo. Le système doit être accompagné d'un logiciel pour les aider dans le processus de création du média.

Voici donc les fonctionnalités que doit avoir notre système :

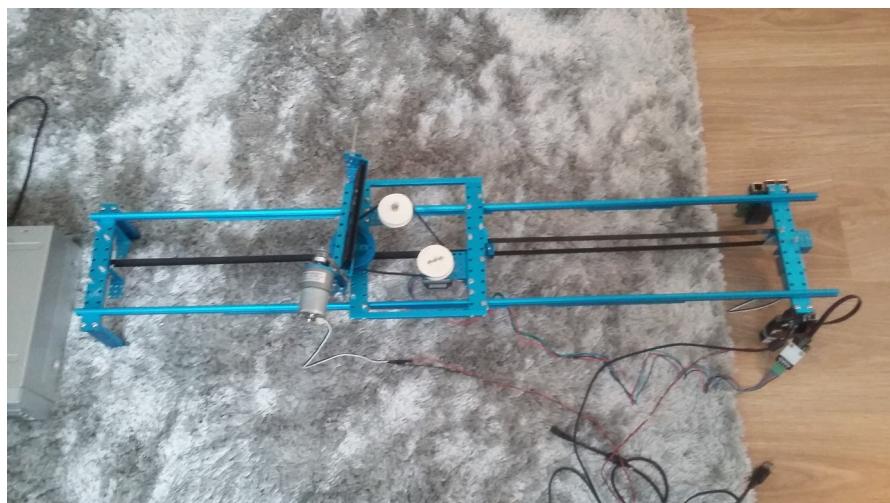
| Partie mécanique | Partie software |
|---|--|
| <ul style="list-style-type: none">-slider motorisé pour permettre la translation sans avoir à nous déplacer au risque de perturber la scène et limiter la manipulation par l'humain-Une plateforme munie d'un socle pour faire glisser l'appareil sur slider-Motorisation permettant de faire un mouvement panoramique-Motorisation permettant de faire un mouvement de tilt | <ul style="list-style-type: none">-contrôle du système mécanique-modification des ISO-modification de l'ouverture-modification du temps d'exposition-modification de la balance des Blancs-autofocus (si possible)-prise et stockage chronologique des photos-un liveview permettant de vérifier la scène en cours-durée de l'animation-superposition de l'image précédente sur le liveview-faire un rendu |

II-Présentation du système mécanique

Lors de la construction de notre système, nous avons utilisé différentes pièces métalliques, vis et écrous mais également des petits moteurs provenant du constructeur Makeblock®. Nous avions aussi à disposition diverses longueurs de courroies. Ces différentes pièces nous ont gracieusement été prêtées par M.Gauthier. L'avantage était que nous avions des pièces compatibles entre elles, il n'y avait pas à s'inquiéter pour les pas de vis mais également pour la fixation des différents moteurs. Un autre point important était le contrôle des différents moteurs : nous pouvions communiquer avec ces derniers via l'environnement Arduino. Les références des moteurs ainsi que différents croquis sont en effet disponibles sur le site du fabricant (<https://www.makeblock.com/>).

a) Système de translation

Le système de translation est composé majoritairement de deux grandes bandes métalliques, fixées bout à bout. Nous obtenons une longueur totale d'environ 1040mm, mais la longueur de déplacement utile est de 742 mm. Ces deux bandes sont fixées de part et d'autre du système sur deux pieds.

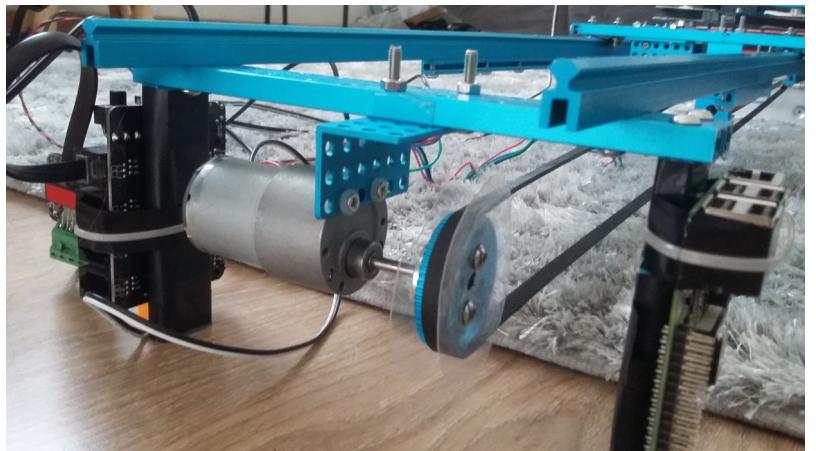


Système de translation

Nous retrouverons par la suite, le système du slider et la tête (socle pour l'appareil photo) fixée dessus. Afin que la translation puisse s'effectuer, nous avons fixé une grande courroie, juste en dessous du système de translation. Elle recouvre l'intégralité de la distance entre les deux pieds et est couplée à un moteur à courant continu (*DC Gear motor rpm 50 37mm*), qui permettra de la faire tourner et donc ainsi, de déplacer l'ensemble. Il fallait s'arranger pour bien tendre la courroie. Nous l'avons donc légèrement pincée afin de réduire sa longueur. Nous nous sommes débrouillés pour la fixer sur l'élément mobile du slider. Nous nous ne voulions bien sûr pas couper la courroie.



La courroie prise entre deux pièces métalliques



Le moteur continu permettant de faire tourner la courroie du système de translation

b) Le système de slider

Le système de slider se présente sous la forme d'une plateforme rectangulaire qui se déplace sur les deux grandes bandes métalliques par l'intermédiaire de la courroie inférieure. Il glisse sur ces dernières grâce à huit roulements à billes, réduisant toutes frictions, fixés de chaque côté des bandes pour les quatre coins de l'ensemble.



Un roulement à billes

c) Les mouvements panoramiques

Pour obtenir des mouvements panoramiques, nous avons intégré à la plateforme mobile slider un système de trois roues fixées à une courroie. Un moteur pas à pas est fixé sur une des roues et permettra de faire tourner le socle où viendra se positionner l'appareil photo numérique. Au commencement, nous avions utilisé seulement deux roues mais la courroie était trop longue et donc nous avions un problème concernant sa tension. Nous l'avions scotchée sur elle-même pour réduire sa longueur, mais cette technique était bien trop rudimentaire et n'était pas durable. Nous avons donc pallié ce problème en rajoutant une troisième roue à l'ensemble. Pour éviter que la courroie ne sorte de son axe, nous avons créé des petits disques en carton afin de diriger celle-ci.



Le système des trois roues supportant la courroie

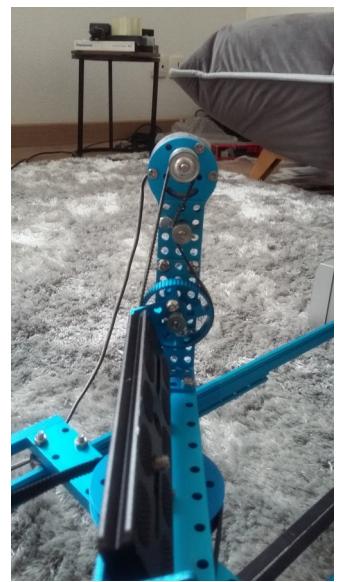


Le moteur pas à pas (42BYGHW609D4P1-X3 - Makeblock 81042) permettant de faire tourner l'ensemble

d) Les mouvements de tilts

Pour les mouvements de tilts, nous avons créé un petit système composé d'une roue et d'un moteur à courant continu (même référence que celui utilisé pour le système de translation). Cette combinaison roue/moteur fait tourner le socle de l'appareil fixé entre la roue et une autre pièce métallique de l'autre côté. C'est l'axe qui demande le plus de contrainte, notamment lorsque l'on doit soulever l'appareil. La vitesse du moteur sera ainsi paramétrée pour réussir à soulever l'appareil. Nous avons testé le système avec l'appareil le plus lourd que nous avions, objectif déployé au maximum : Un Sony A77 Mark II muni d'un objectif SIGMA 70-300mm APO DG (pour un ensemble pesant 1,5 kg).

Avec la puissance standard que nous avons choisie (50 % sur notre échelle), nous avons réussi à soulever l'appareil (Nous avons créé la possibilité de sélectionner 25, 50 ,75 ou 100 % selon une échelle de puissance que nous avons établi préalablement). La vitesse est dissymétrique entre un tilt-up et un tilt-down : il faut augmenter celle-ci lors du tilt-up.



La combinaison roue/moteur permettant les mouvements de tilts



Le Sony A77 Mark II avec son objectif SIGMA

e) La carte Arduino-Uno : ORION (et Raspberry Pi)

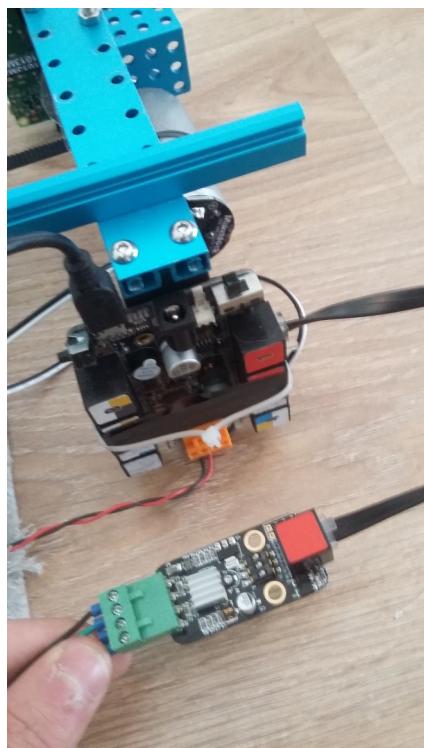
Pour communiquer avec les différents moteurs présents dans le système, nous utilisons l'environnement Arduino. Nous disposons d'une carte ORION (basé sur le modèle Arduino-Uno) pour réaliser ces communications. Cette dernière était fixée sur un des pieds (coté moteur système de translation), avec du châssis afin d'éviter un contact avec la partie métallique du système. Ainsi tous les moteurs sont reliés à cette carte. Cette dernière peut être alimentée par l'intermédiaire de six piles AA ou par une alimentation secteur délivrant une tension de 9,6 Volts. Il est à noter que pour le moteur pas à pas, il est nécessaire de brancher un contrôleur supplémentaire (Stepper Driver).

Au début, nous avions également installé un Raspberry Pi 3 b+ (sur l'autre pied côté moteur). Ce dernier faisait fonctionner le système mais avec pas mal de peine. Nous l'avons délaissé par la suite, un modèle supérieur pourrait néanmoins convenir selon nous.

À noter que nous avions fait le test en VNC car le projet de base était de contrôler le tout à distance sur un réseau local et que nous n'avions plus eu le temps de trouver une autre solution que celle-ci.



Carte Orion



Module additionnel pour le moteur pas à pas (Stepper Driver)



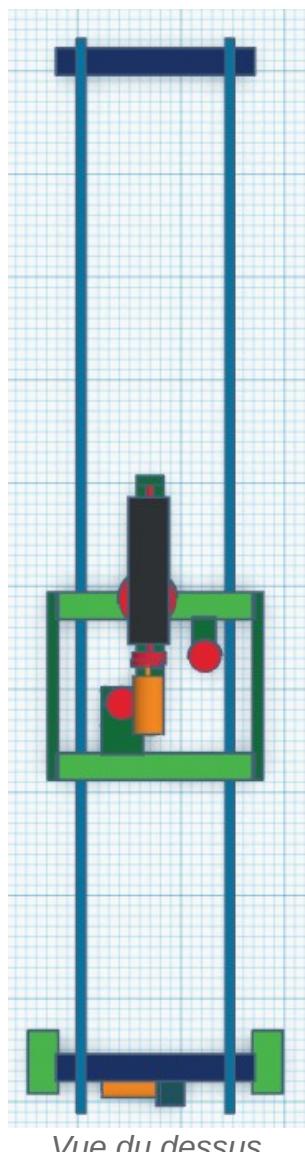
Le Raspberry 3 b+

f) Représentation 3D à l'échelle 1/10 du système sur Tinkercad

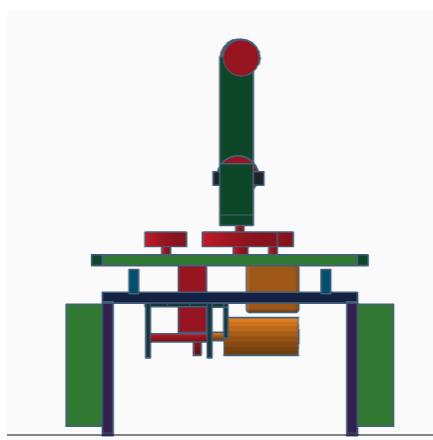
Afin de mieux se représenter le système dans sa globalité, nous avons réalisé une maquette 3D de celui-ci via Tinkercad, une application en ligne disponible gratuitement. Ainsi donc, en suivant le lien :<https://www.tinkercad.com/things/jIelHqzOsAS-slider/edit?sharecode=yKQX-MbYHr0QF9LJIOcdLgyEdSpOj036C3q-pTF6hQ> (ou <https://tinyurl.com/Slider-TYPE2>).

Pour l'utilisation de la fenêtre, cliquez sur l'élément voulu : les dimensions de celui-ci apparaîtront. Vous pouvez aussi placer l'outil règle pour mesurer les distances entre plusieurs éléments. Il est à noter que la représentation du slider est simplifiée, respectant un certain code couleur avec les moteurs en orange et les parties liées aux courroies en rouge.

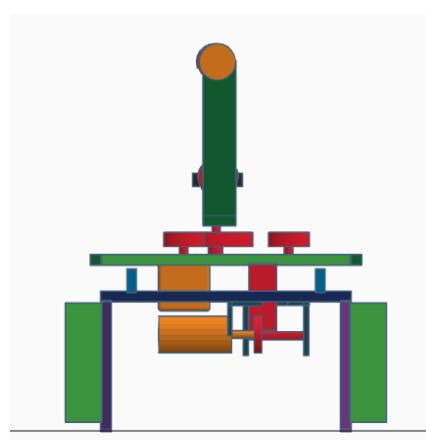
Voici quelques illustrations que vous pourrez retrouver sur la représentation 3D :



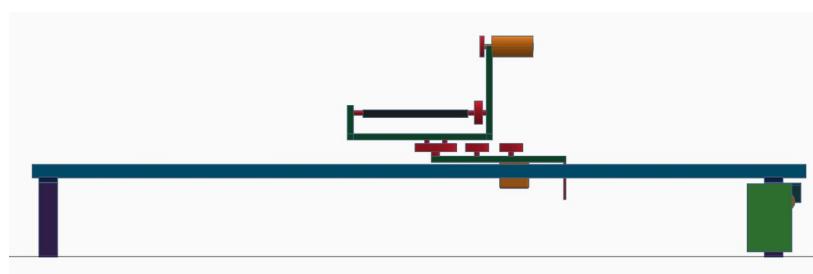
Vue du dessus



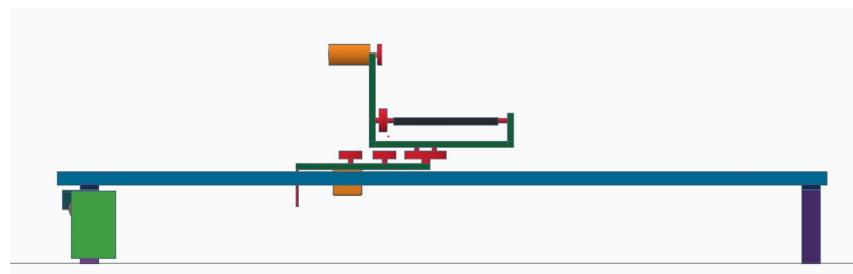
Vue de derrière



Vue de devant



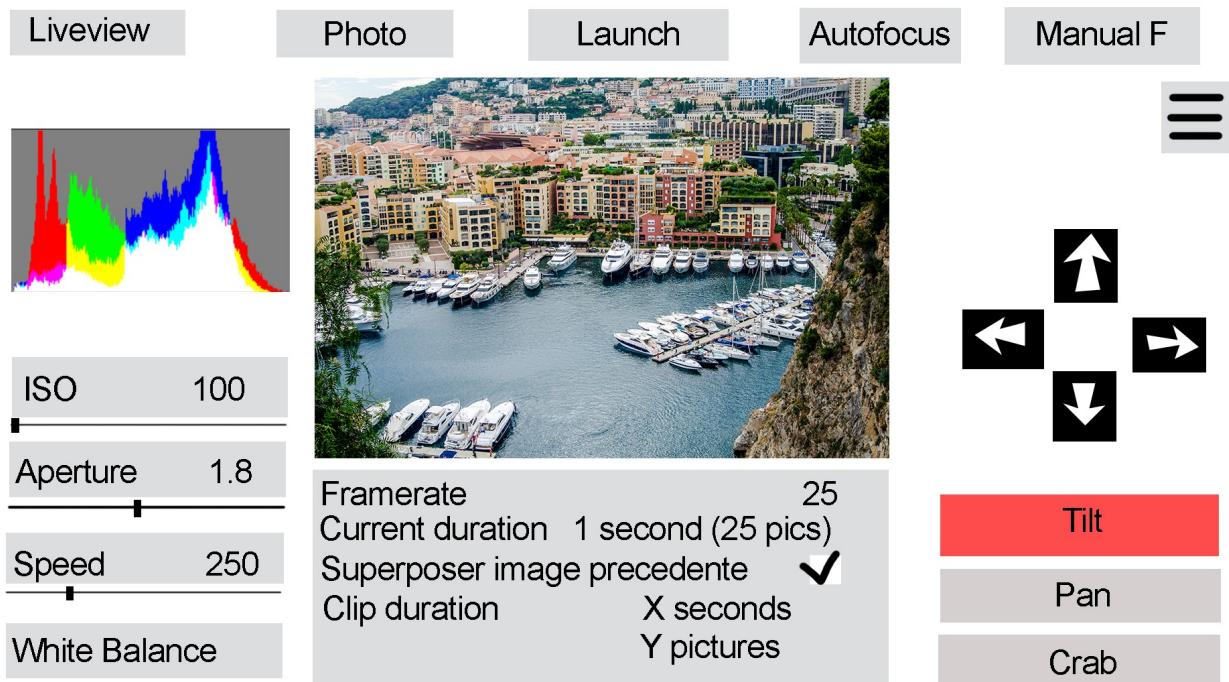
Vue de gauche



Vue de droite

III-Interface graphique

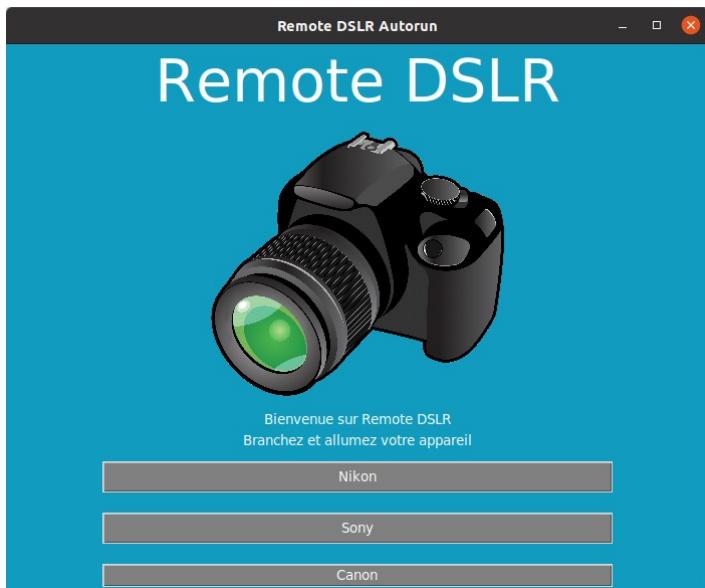
Voici l'esquisse de l'interface qu'on avait imaginée au début du projet :



Le temps imparti pour pour la réalisation du projet ne nous a pas permis d'implémenter l'histogramme pour analyser la photo en temps réel.

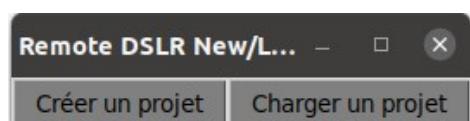
Voici l'interface de notre projet final :

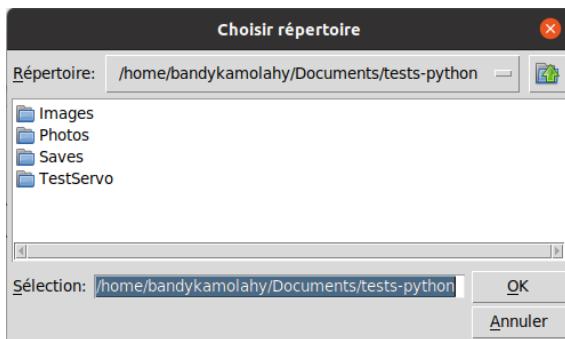
a) Au démarrage



Au lancement de l'application, une première fenêtre s'ouvre, nous permettant de choisir et de renseigner la marque de l'appareil que nous connectons à l'ordinateur. Même si gPhoto2 peut détecter la marque et le modèle, cette première fenêtre est une nécessité technique car on interroge l'appareil une première fois sur les possibilités qu'il offre. Il ne s'agit pas d'un d'une partie superflue. Et le code permet de réduire l'encombrement et la complexité du script principal

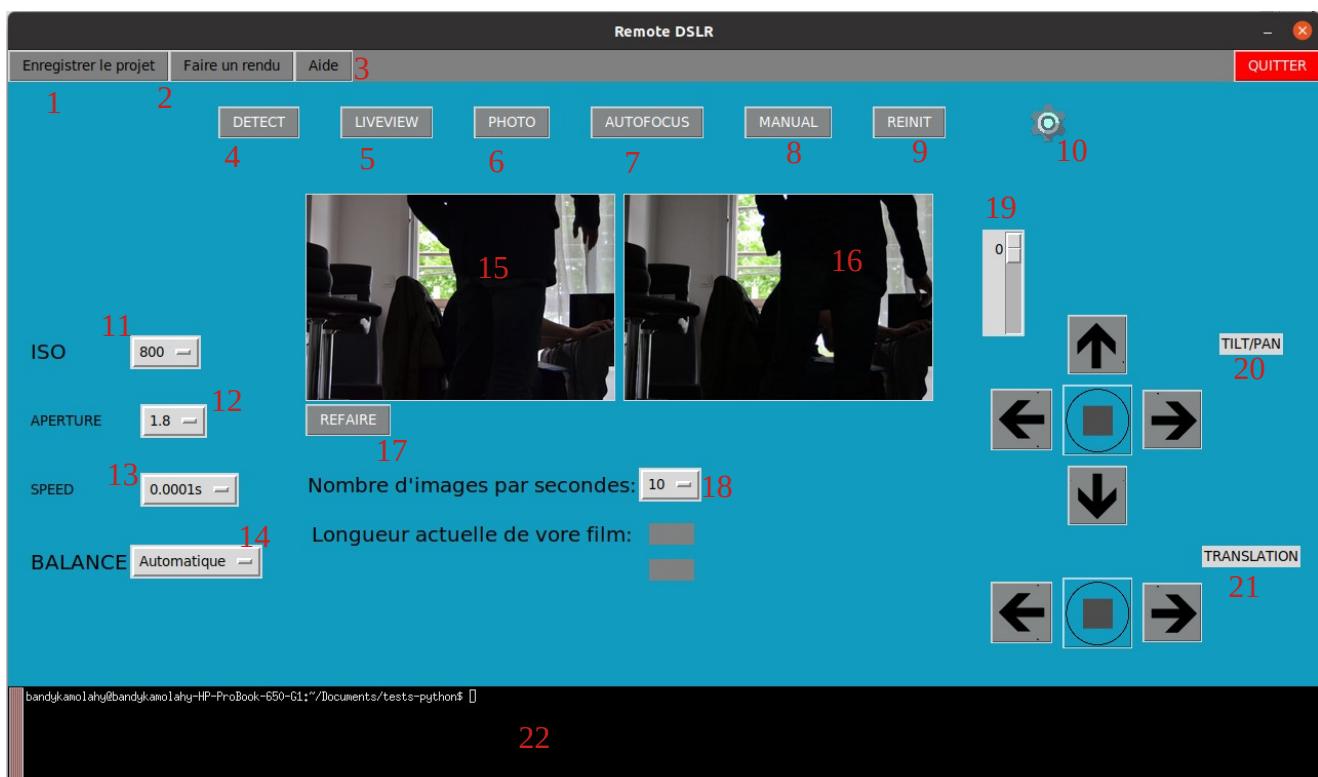
Dès qu'on a lancé l'application, on ouvre une petite boîte de dialogue qui nous dit de choisir si on veut créer un nouveau projet ou en charger un déjà existant



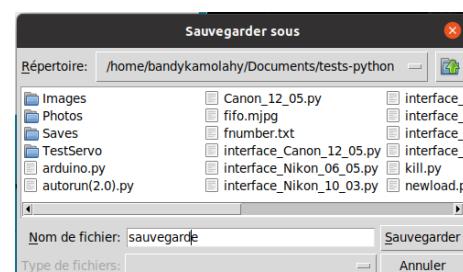


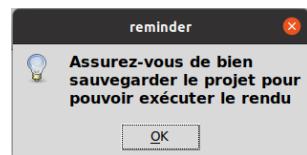
On est ensuite redirigés vers une boîte de dialogue pour choisir un dossier de destination si c'est un nouveau projet ou bien un fichier «.pickle» si on a choisi de charger un projet précédent.

b) Présentation de l'interface utilisateur

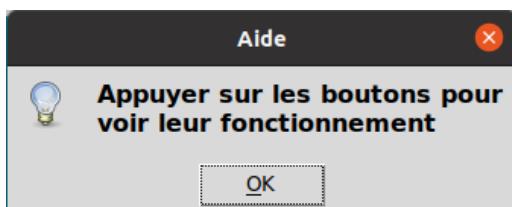


1-Sauvegarde du projet en cours : ouvre une boîte de dialogue pour choisir l'emplacement et le nom de la sauvegarde
Il sera enregistré dans un fichier «.pickle» que l'on pourra recharger plus tard

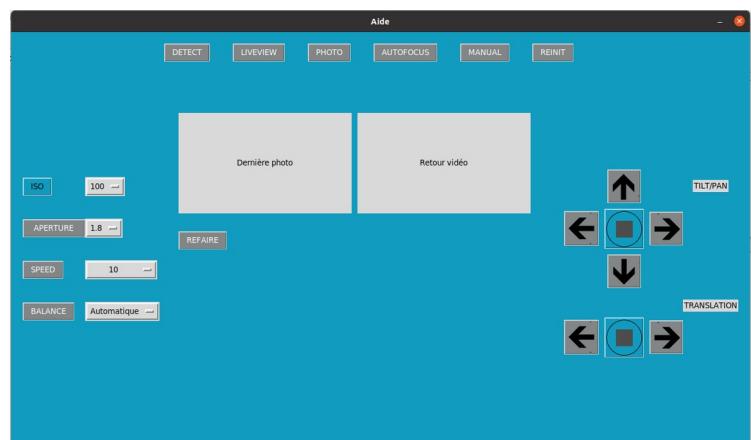




2-Faire un rendu : ouvre une boîte de dialogue pour faire un rendu brut du projet en cours. On choisit le codec, l'encapsuleur, le framerate, l'emplacement et le nom du fichier de destination



3-Aide : Ouvre une fenêtre d'aide qui ressemble à l'interface utilisateur. Il suffit de cliquer sur un bouton pour comprendre comment il fonctionne et à quoi il correspond.



Modèle

Nikon DSC D7000 (PTP mode)

Port

usb:003, 005

4-Detect : Affiche l'appareil connecté dans le terminal qui aura lancé l'application

5-Liveview : active/désactive le liveview

Acquisition d'aperçus d'image comme vidéo de « stdout ». Ctrl-C pour abandonner.
Ctrl-C reçu... Quitter.
Acquisition de film terminée (241 trames)

Acquisition de film terminée (79 trames)
Le nouveau fichier est à l'emplacement /capt0000.jpg de l'appareil
Enregistrement du fichier en /home/bandykamolahy/Documents/tests-python/capt004.JPG
Effacement du fichier /capt0000.jpg de l'appareil
Acquisition d'aperçus d'image comme vidéo de « stdout ». Ctrl-C pour abandonner.

6-Photo : prend la photo et l'enregistre sur l'ordinateur.

7-Autofocus : active le mode autofocus

Le mode d'autofocus doit être choisi au préalable

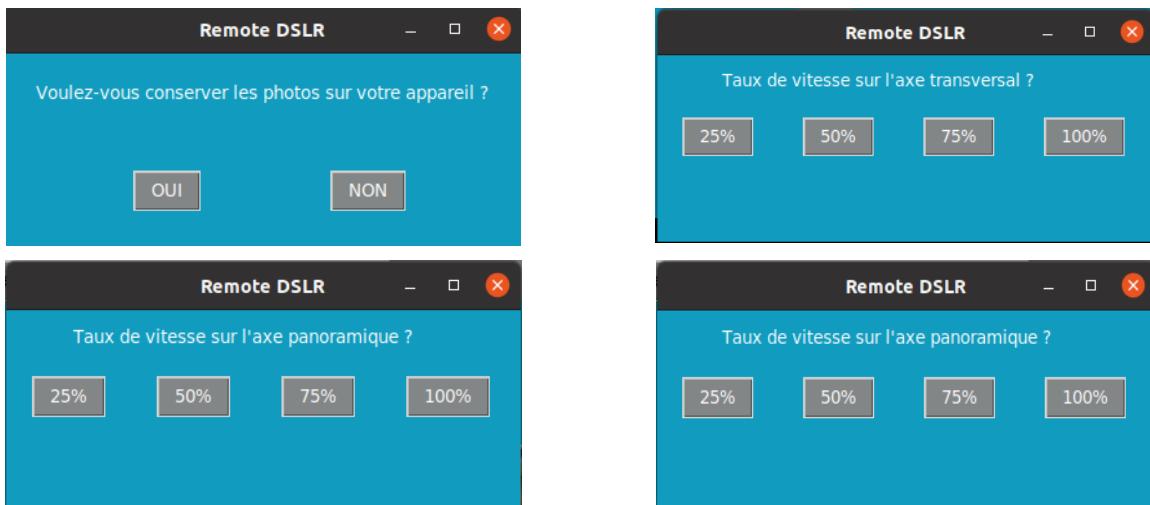
8-Manual

Désactivation du mode autofocus

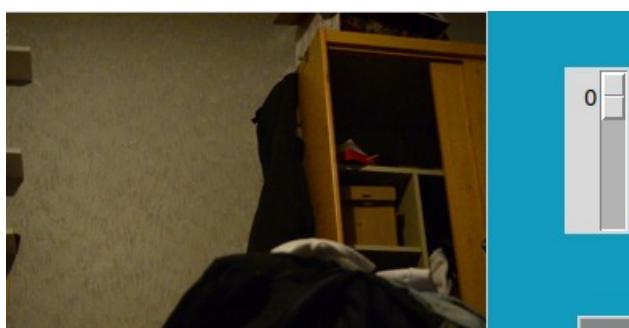
9-Reinit : demande à l'utilisateur de relancer le programme en cas de bug

Réinitialise les paramètres de l'appareil photo :
ISO=100
Aperture=le plus grand possible
Shutterspeed=1/100

10-Paramètres : permet de choisir si on veut garder une trace de la photo sur l'appareil, permet de donner la puissance et la vitesse allouées aux moteurs pour les mouvements de caméra



| | |
|--------------------|--|
| 11-ISO | Menu déroulant pour choisir l'ISO |
| 12-Aperture | Menu déroulant pour choisir l'ouverture |
| 13-Shutterspeed | Menu déroulant pour choisir le temps d'exposition |
| 14-White Balance | Menu déroulant pour choisir la balance des Blancs |
| 15-Affichage Image | Image précédemment prise |
| 16-Affichage Vidéo | Liveview et superposition avec l'image précédente |
| 17-Refaire | Reprendre une photo ratée |
| 18-Framerate | Choix du framerate pour estimer la longueur courante du film |



19-Alpha Level : permet de choisir l'opacité de la superposition de l'image précédente sur le liveview. Cet outil est très efficace car il peut aussi servir de prévisualisation de l'enchaînement entre l'image et du liveview. Le slider est gradué de 0 à 10 tel qu'à 0, on n'a que le liveview et à 10 uniquement l'image précédente.



20&21-Touches directionnelles : donnent des indications à l'utilisateur sur les mouvements de caméra

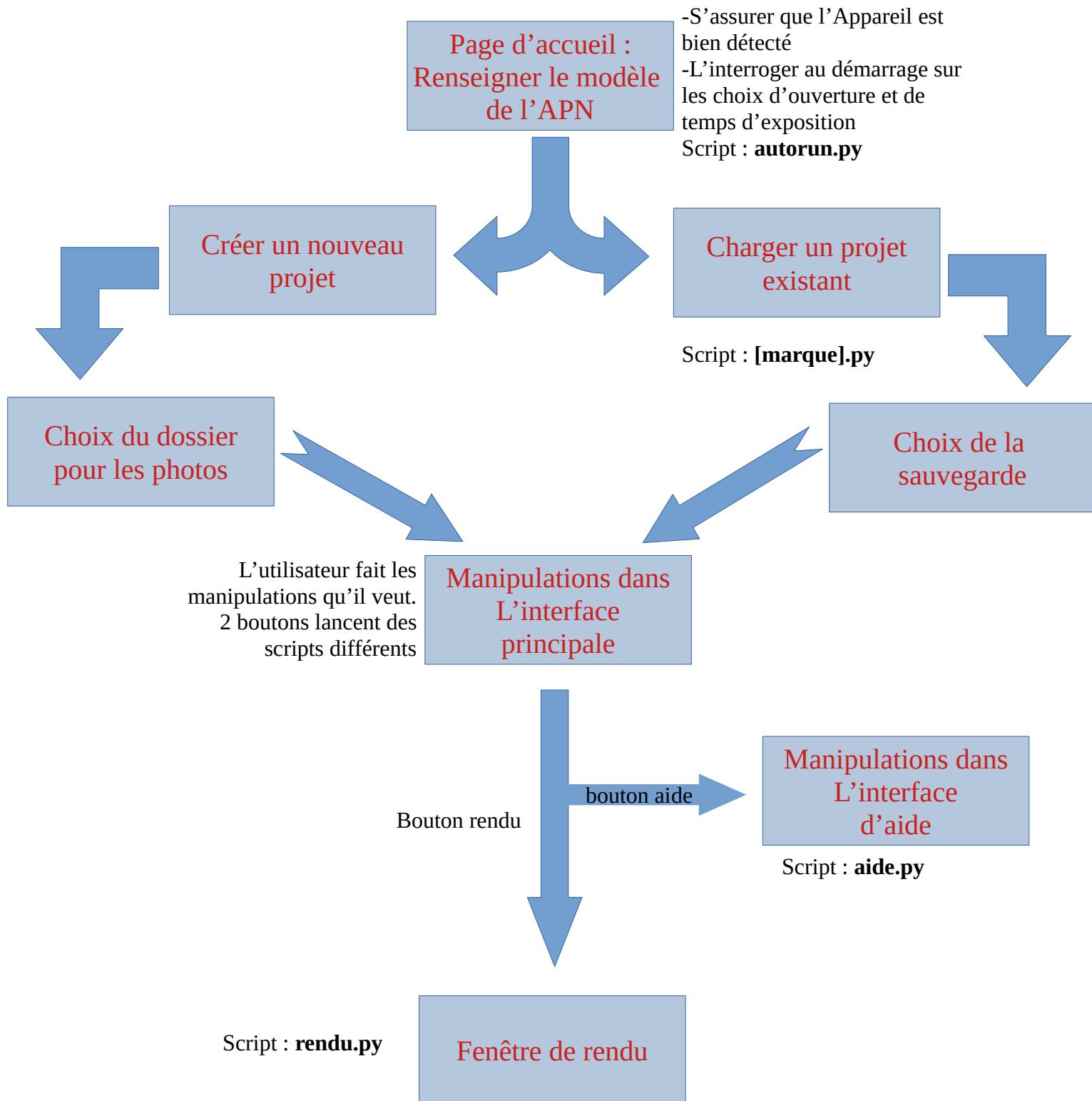


22-Terminal : Inspiré de l'IDE Arduino, le but est de pouvoir vérifier la sortie des commandes ici (pas encore disponible) ; il permet aussi des manipulations en lignes de commandes si pour une quelconque raison, l'utilisateur en a besoin

```
bandykamolahy@bandykamolahy-HP-ProBook-650-G1:"/Documents/tests-python$
```

c) Structuration du logiciel

Voici un schéma simplifié expliquant le fonctionnement global de notre logiciel



IV-gPhoto2 sous Linux/UNIX



Gphoto2 est une application qui s'utilise en ligne de commandes, programmée avec la bibliothèque libgphoto2. Elle sert à contrôler un appareil photo numérique via un ordinateur qui tourne sous un système d'exploitation de type UNIX. Il s'agit d'un logiciel libre disponible sous la licence publique GNU.

Pour l'utiliser, on a installé l'OS Ubuntu, dans la version la plus récente à savoir 20.04.2 LTS en dual boot sur nos PC. De plus le Raspberry Pi 3 B+ que l'on nous a confié tourne sous Raspbian (OS avec un kernel Debian spécialement développé pour les Raspberry).

Il a donc fallu se familiariser avec les manipulations en lignes de commandes avant toute chose. On a déjà eu des cours dessus en L2 mais c'est la première fois que l'on travaille exclusivement sous un système de type UNIX.

a) Installation de gPhoto2

Il est toujours avisé de s'assurer que le système est à jour

```
~$ sudo apt-get update
```

Pour installer l'application gPhoto2, il suffit d'être connecté à internet et taper la ligne de commande suivante :

```
~$ sudo apt-get install gphoto2
```

Aujourd'hui, la version la plus récente de gPhoto2 est la version 2.5.23

Pour vérifier la version de gPhoto2, il faut taper la commande :

```
~$ gphoto2 -v
```

Pour installer gPhoto2 sur un Raspberry, il convient parfois de procéder différemment en passant par le snap (installation du logiciel avec toutes ses dépendances) gPhoto2 :

```
~$ sudo apt install snapd
~$ sudo snap install core
~$ sudo snap install gphoto2
```

Dans de rares cas, il faudra installer toutes les bibliothèques manuellement, les étapes sont listées en annexe.

b) Commandes essentielles de gPhoto2

| Commande | Actions |
|--------------------------|---|
| \$ gphoto2 --auto-detect | Donne le nom, le fabricant et le port d'un appareil connecté en USB au PC. |
| \$ gphoto2 --list-config | Montre tous les registres utilisables et modifiables sur l'Appareil photo. La sortie de cette commande dépend entièrement de l'appareil connecté |

| | |
|---|---|
| \$ gphoto2 --get-config [conf] | Cette commande peut donner l'état courant d'un paramètre de l'APN, et selon l'appareil, liste les choix possibles pour le paramètre concerné (ISO, aperture/f-number , shutterspeed...) |
| \$ gphoto2 --capture-movie [i] | Capture une vidéo pendant une durée [i] qui peut être soit en secondes soit en frames. On peut aussi rediriger la sortie standard de la commande vers un FIFO |
| \$ gphoto2 --set-config(-index) [conf]=value | Modifie la valeur d'un paramètre de l'appareil photo en lui spécifiant la valeur absolue ou un numéro de choix |
| \$ gphoto2 --capture-image-and-download (--keep) --filename [path+name] | Prend une photo, l'importe vers l'ordinateur (on peut laisser ou non une copie sur la mémoire de l'appareil photo) en lui spécifiant un nom et un chemin |

c) Autres commandes essentielles de l'environnement Linux

Mis à part les commandes classiques de bases telles que **cd**, **ls** ou **mkdir...** il existe plusieurs commandes à connaître :

- la commande **| grep [mot]** permet de faire une recherche dans la sortie de la commande précédant le pipe
- la commande **ps aux** permet de lister les process actifs ainsi que leur PID

Ex : la commande **ps aux | grep gphoto2** permet de lister tous les process actifs liés à gPhoto2

Il faut savoir faire la différence entre les différents signaux de terminaison : **kill**, **pkill**, **pkill -9**, **killall**

Sous Linux, il y a des différences à connaître entre un SIGKILL et un SIGTERM :

-Un SIGTERM est un signal de terminaison qui est envoyée à un processus pour lui dire de s'arrêter « proprement ». C'est-à-dire que le processus peut s'arrêter immédiatement tout comme il peut prendre un petit délai avant de s'arrêter tout en libérant les ressources utilisées. La gestion de ce signal est faite de manière de la façon suivante : l'OS ne fait qu'envoyer le signal mais l'arrêt est effectué par le processus lui-même. Il arrive souvent que ce signal soit ignoré, surtout dans le cas d'un processus qui gère un périphérique externe.

-Un SIGKILL est un genre de signal de priorité « forte », qui ne peut ni être retardé ni être ignoré. Il est moins sûr d'utilisation car il force l'arrêt du processus et si jamais ce-dernier possédait un processus fils, alors il pourrait ne pas être informé de l'arrêt de son parent. Le signal est directement envoyé au noyau et c'est l'OS qui arrête le processus.

| SIGTERM | SIGKILL |
|-------------------|----------------|
| kill (-15) [pid] | kill -9 [pid] |
| killall [nom] | pkill -9 [nom] |
| pkill (-15) [nom] | |

En règle générale, killall est un peu plus sûr que pkill car il va chercher exactement le nom du processus dans les 15 premiers caractères des intitulés de chaque processus.

La commande **pkill -9** n'est pas si dérangeant dans notre cas car elle va tuer tous les process qui comportent le nom spécifié, donc, y compris les process enfants.

-mkfifo [nom].extension permet de créer un FIFO (First Input First Output). Il s'agit d'une structure de données qui fonctionne comme une file d'attente, le premier à entrer est aussi le premier à sortir. Un fifo permet de mémoriser temporairement des données en attente de traitement.

Remarque : On s'est aperçu que le choix du signal peut avoir des effets totalement différents d'un appareil photo à un autre ; c'est pourquoi vous trouverez aussi bien des SIGTERM que des SIGKILL dans nos codes.

c) Une question de modèle

Pour écrire un script qui fonctionne bien, il est intéressant de connaître tous les registres manipulables par gPhoto2 sur l'appareil qui est connecté. Cette étape est primordiale dans notre projet. Même si les paramètres généraux se manipulent de la même façon sur la plupart des appareils photos numériques supportés par l'application, il existe des subtilités qu'il faut connaître d'un appareil à un autre. Voici des exemples :

-le paramètre d'ouverture chez Nikon et Sony est identifié par « **f-number** », tandis que chez Canon, on utilise « **aperture** »

-la sortie des commandes **\$ gphoto2 --get-config [paramètre]** donne non seulement l'état courant du paramètre mais aussi une liste de tous les choix de valeurs chez Nikon et Canon ; chez Sony, elle ne donne que la valeur courante

Ainsi, pour bien commencer, il convient de lancer les commandes suivantes et vérifier la sortie :

```
$ gphoto2 --list-config
```

(Tous les registres affichés sont manipulables avec les commandes **\$ gphoto2 --set-config** et **\$ gphoto2 --get-config**.)

\$ gphoto2 --get-config iso (ou équivalent dans la liste de configurations)

\$ gphoto2 --get-config shutterspeed (ou équivalent dans la liste de configurations)

\$ gphoto2 --get-config f-number (ou équivalent dans la liste de configurations - Ex: aperture si boîtier Canon)

\$ gphoto2 --get-config whitebalance (ou équivalent dans la liste de configurations)

Si une des sorties de ces commandes donne tous les choix possibles que peut prendre le paramètre concerné, alors on pourra en créer un dictionnaire utilisable pour les menus déroulants.

Par exemple, le Sony A77 qu'on a à disposition ne nous fournit pas les choix, il faut dans ce cas là utiliser une liste par défaut utilisable par tous les appareils photos numériques en mode manuel.

Pour utiliser un nouvel appareil, il est préférable de suivre les étapes précédentes avant de lancer l'application. Sinon la version **Sony.py** permet une utilisation moins sensible car elle utilise des listes de choix standards.

V-Scripts Python : les fonctions de base

Le principe de nos scripts est d'exploiter les commandes de gphoto2 et non la librairie libgphoto2. On a bien envisagé cette possibilité mais les swig python (piggyphto et python-gphoto2) de librairie ne sont plus à jour depuis longtemps, sont incomplets et manquent cruellement de documentation.

Avant tout, il faudra installer toutes le bibliothèques nécessaires pour le script. Il faudra faire attention à la version de python qui est installée sur notre ordinateur ; **s'il s'agit de Python 3, il faudra écrire les « 3 » sans les parenthèses, sinon, on ignore les « (3) »**

-*Installation de l'utilitaire python*

```
$ sudo apt-get install python-setuptools
```

-*Installation de l'utilitaire pip*

```
$ sudo apt-get install python(3)-pip
```

Si pip ne fonctionne pas, essayer pip3 (python3)

-*Installation de pynput pour lecture du clavier*

```
$ pip(3) install pynput
```

-*Liaison sérielle pour commander l'arduino*

```
$ pip(3) install pyserial
```

-*Installation de pillow*

```
$ pip(3) install pillow
```

-*Installation d'ImageTk pour PIL : bibliothèque graphique basique de traitement d'images*

```
$ sudo apt-get install python-pil.imagetk (python2)
```

```
$ sudo apt-get install python3-pil python3-pil.imagetk (python3)
```

-*Installation de OpenCV : bibliothèque graphique de traitement d'images en temps réel*

```
$ sudo apt-get install python3-opencv
```

```
$ pip(3) install opencv-python
```

-*Installation de Tkinter : module pour créer l'interface utilisateur*

```
$ sudo apt-get install python(3)-tk
```

-*Installation de Pickle: module python pour sauvegarder des données en mode binaire*

```
$ pip(3) install pickle-mixin
```

a) Une fenêtre et des widgets pour commencer

On commence par importer la bibliothèque **tkinter**

```
from tkinter import *  
from tkinter import filedialog  
from tkinter import messagebox
```

```
#utile pour décrire une fenêtre  
#utile pour les boîtes de dialogue de chemins  
#utile pour les boîtes de dialogue sans actions particulières
```

Cette bibliothèque est facile à utiliser pour décrire une fenêtre avec des widgets tels que des zones de textes, des boutons, des menus déroulants et des sliders.

Il faut bien comprendre que Tkinter agit comme une sorte de boucle d'affichage car le programme relit tout le temps le programme pour voir ce qui a changé depuis le dernière affichage.

Voici un exemple de fenêtre très basique avec des boutons et du texte :

```
window = Tk()  
window.title("Remote DSLR")  
window.geometry("400x150")  
window.config(background="#119CBF")  
  
question=Label(window, text="oui ?")  
question.place(x=200,y=10)  
  
quart = Button(window, text="25%", command=quart)  
quart.grid(row=0, column=0)  
  
demi = Button(window, text="50%", command=demi)  
demi.grid(row=0, column=1)  
  
window.mainloop()
```

#on crée une fenêtre appelée « window »
#le titre de « window » est « Remote DSLR »
#la fenêtre a une dimension 400x150
#la couleur de fond (ici en une nuance de bleu)

« question » est un texte (label)
placé dans « window »
#en position x=200 et y=10

« window » est divisée en grille de 2 colonnes
#quart est un bouton dans la 1ère et active une
#fonction « quart » définie au préalable
#demi fonctionne pareil avec et placé dans la
#2ème colonne

#fin de la boucle définissant la fenêtre

On a alors la fenêtre suivante :



On définit un slider de la façon suivante :

```
Alpha = Scale(window, from_=0, to=10, command=change_alpha)  
Alpha.place(x=950, y=175)  
#il s'agit d'un slider associé à la fonction change_alpha() avec une plage allant de 0 à 10
```

Pour définir un menu déroulant, voici la méthode :

```
variable = StringVar()  
variable.set(liste[0])  
menu_deroulant = OptionMenu(window, variable, *liste)  
menu_deroulant.grid(row=0, column=1, sticky=W)  
variable.trace("w", callback)  
# « menu_deroulant » permet de choisir la valeur de « variable » parmi les valeurs de contenues dans  
« liste »  
#au démarrage, variable = liste[0]  
# « trace » permet d'associer « variable » à une fonction « callback »
```

Pour créer une boîte de message :

```
messagebox.showinfo('Default', 'Pas encore disponible')  
#ouvre une fenêtre d'avertissement intitulé 'default' et affichant 'Pas encore disponible'
```

b) Un dictionnaire pour les valeurs à utiliser

Le Script **autorun.py** est le code source de la page d'accueil, elle n'est pas anodine car elle sert avant tout à :

- s'assurer que gPhoto2 est disponible et prêt à l'emploi
- demander à l'appareil les spécifications qu'il peut prendre en ISO et en temps d'exposition (sauf dans le cas du Sony qui ne les fournit pas par la commande **-get-config**)
- détecter si un appareil est bien connecté et reconnu par gPhoto2

Grâce aux bibliothèques **os** et **subprocess** on peut écrire des lignes de commandes à partir de notre Script python.

```
import os          #os.system() est semblable à une commande qu'on tape directement au Terminal
import subprocess  #independamment de notre script
                   #subprocess crée des processus fils
```

Dans certains cas, comme une prise de photo ou le liveview (décrit plus tard), subprocess ne veut pas lancer la commande nécessaire à appeler gPhoto2, c'est là qu'intervient os.

Voici donc comment on fait pour créer les dictionnaires des valeurs, et comment les utiliser :

- Dans le script « autorun.py »

```
os.system('gphoto2 --auto-detect')
#detection de l'appareil est affichage sur le Terminal

def Nikon():
    os.system('pkill gphoto2') #s'assurer que gphoto2 est disponible et prêt à l'emploi
    time.sleep(0.2)
    os.system('gphoto2 --get-config f-number > ./fnumber.txt ')
    #on stocke la sortie de la commande dans un fichier texte
    #la commande sera « --get-config aperture > ./fnumber.txt » si on a un Canon
    os.system('gphoto2 --get-config shutterspeed > ./shutterspeed.txt ')
    #on stocke la sortie de la commande dans un fichier texte
    window1.destroy()
    os.system('python3 Nikon.py')
    #on ferme la fenêtre d'accueil et on lance le script principal pour le Nikon
```

Remarque : On ne fera pas la même chose avec un Sony de la gamme des A77 car on n'obtient pas la liste des valeurs possibles avec cette commande, il faudra définir nous-mêmes la liste

- Dans le script principal Nikon ou Canon

```
f = open("./fnumber.txt", "r", encoding="utf-8")
gphoto2_data: str = f.read()
f.close()

f_number_dict: dict = {}
my_key: str
my_value: str

for line in gphoto2_data.split("\n"):
    if line == "END" or line == "":
        continue
    line_list: list = line.split(":")
    line_list[0] = line_list[0].strip()
    line_list[1] = line_list[1].strip()
    if line_list[0] == "Choice":      #si on rencontre le mot clé « choice » on met à jour le dictionnaire
        #on parcourt la chaînes de caractères
        #on ignore la ligne de fin et le caractère d'échappement
        #on sépare les données là où il y a « :»
```

#on ouvre le fichier texte précédent
#on lit et on stocke le contenu en chaîne
#on ferme le fichier

#on définit un dictionnaire pour contenir les données

```

if 'Choice' not in f_number_dict:
    f_number_dict.update({'Choice': {}})
#on crée en fait un autre dictionnaire 'choice' dans le dictionnaire précédent et contiendra 'my_key' et 'my_value'
    my_key, my_value = line_list[1].split(' /')
#on sépare les données là où il y a ' /' (juste un espace dans le cas d'un Canon) et la première partie sera 'my_key' en
#occurrence le numéro de choix et la deuxième 'my_value'
    f_number_dict['Choice'].update({my_key: my_value})
else:
    f_number_dict.update({line_list[0]: line_list[1]})
#si le mot-clé n'est pas 'choice' alors met tout simplement à jour le dictionnaire 'f_number_dict'

array = len(f_number_dict['Choice'])*[0]
#on crée un tableau de 0 de la longueur du dictionnaire f_number_dict['Choice']
for i in f_number_dict['Choice']:
    array[int(i)] = f_number_dict['Choice'][i]
#on doit maintenant convertir le dictionnaire 'choice' en une liste car le menu déroulant de tkinter ne gère pas les
dictionnaires à cause du type des indices

```

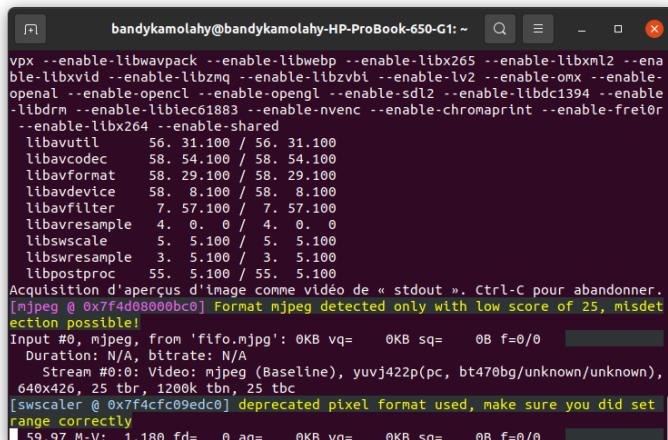
c) le liveview

Il faudra avant toute chose créer un FIFO (cf. partie V-b sur les lignes de commandes) encapsulé en MJPEG (format du liveview) à la racine du projet :

```
$ mkfifo fifo.mjpg
```

On peut ensuite essayer la commande suivante :

```
$ gphoto2 --capture-movie --stdout>fifo.mjpg & ffplay fifo.mjpg
```



Le liveview est un mjpeg à 25 images par seconde.

Ainsi, dans notre script, on a :

-une initialisation du liveview

```
os.system('gphoto2 --capture-movie --stdout>fifo.mjpg &')
```

-une variable **globale** pour vérifier si le liveview est actif ou non

```
global active
active = True          #le liveview est activé au lancement
```

-une fonction **switch()** pour activer et désactiver le liveview

```
def switch():
    global active          #active ou not active désigne l'état du liveview
    if active:              #on désactive si il est déjà actif
        os.system('pkill gphoto2')
    #étrangement pour un Nikon D7XXX, killall fonctionne mais pas avec les Canon et Sony à disposition
    #selon les différentes documentations lues, il serait préférable dans la mesure du possible d'utiliser killall
    time.sleep(0.5)
    active = False         #on change le statut du liveview
else:
    os.system('gphoto2 --capture-movie --stdout>fifo.mjpg &')
    #sinon on relance le liveview
    active = True
    show_frame()
```

-Maintenant, pour intégrer le liveview à notre fenêtre tkinter, il faut importer les bibliothèques opencv pour la vidéo, pillow pour l'image superposée et numpy pour la vérification si une image n'est pas vide

```
from PIL import Image, ImageTk
import cv2
import numpy as np
```

-On définit le fichier qui doit être lu par opencv

```
cap = cv2.VideoCapture('fifo.mjpg')
```

-On va pouvoir définir la fonction récursive **show_frame()**

```
def show_frame():
    global image
    #image correspond à la photo précédente, si on vient de créer un projet, elle correspond à une image png transparente
    #elle est mise à jour par la fonction qui sert à prendre une photo
    global alpha
    #alpha correspond au niveau de transparence pour la superposition
    #il est mis à jour par la fonction associée au slider
    _, frame = cap.read()          #on lit chaque image de la vidéo
    if np.shape(frame) != ():      #pour éviter un problème, on ne fait rien si l'image est vide
        height, width, depth = frame.shape
        prec_img = cv2.resize(image,(width,height))
    #on scale la photo pour qu'elle aie la même dimension que l'affichage du liveview
    frame=cv2.addWeighted(frame,1-alpha,prec_img,alpha,0)
    #somme pondérée par alpha de l'image et du liveview
    cv2image = cv2.cvtColor(frame, cv2.COLOR_BGR2RGBA)
    img = Image.fromarray(cv2image)
    intermediaire = img.resize((300, 200))
    imgtk = ImageTk.PhotoImage(image=intermediaire)
    Imain.imgtk = imgtk
    #on affiche chaque image du liveview grâce à imagetk de la bibliothèque PIL qui peut dialoguer avec Tkinter
    Imain.configure(image=imgtk)
    Imain.after(10, show_frame)      #la fonction se rappelle elle-même toutes les 10 ms
```

d) Créer, charger, sauvegarder

-Il faut importer 1 bibliothèque supplémentaire pour sauvegarder un projet en cours

```
import pickle
```

-Grâce à filedialog dans la bibliothèque Tkinter, on peut demander à l'utilisateur de renseigner un chemin ou un fichier à charger

```
def commencer():          #cette fonction est lancée avant la boucle tkinter principale
```

```
    def new():           #nouveau projet
```

```
        global image
```

```
        image = cv2.imread('./Images/prev_depart.png')
```

```
#initialisation de l'image superposée au liveview comme une image full transparente
```

```
        global path          #contendra le chemin vers le dossier du projet
```

```
        global frame         #numéro de la prochaine photo initialisé à 1
```

```
        frame = 1
```

```
        path = filedialog.askdirectory()
```

```
#cette fonction renvoie le chemin choisi sous forme d'une chaîne de caractères stockée dans « path »
```

```
        NoL.destroy()
```

```
    def load_project():      #charger un projet
```

```
        global image
```

```
        global frame
```

```
        global path
```

```
        dico: dict={} 
```

```
#les données stockées pendant la sauvegarde sont agencées en dictionnaire donc on les récupère dans un dico
```

```
        file = filedialog.askopenfile(mode="rb", defaultextension=".pickle")
```

```
#un fichier ‘.pickle’ est une structure de données en binaire qu’on vient lire
```

```
        dico=pickle.load(file)
```

```
#la fonction pickle.load() permet de charger le contenu d’un fichier ‘.pickle’
```

```
        file.close()
```

```
        path=dico["chemin"]
```

```
#on récupère le chemin vers le projet
```

```
        frame=dico["numero_image"]
```

```
#on récupère le numéro de la prochaine photo pour ne pas écraser une photo déjà présente
```

```
        print(dico)
```

```
        image = cv2.imread(path+'/prev.JPG')
```

```
#image contiendra la dernière photo prise du projet
```

```
        NoL.destroy()
```

```
NoL = Tk()
```

```
NoL.title("Remote DSLR New/Load")
```

```
NoL.config(background='#119CBF')
```

```
NEW = Button(NoL, text="Créer un projet", bg='grey', fg='black', command=new)
```

```
NEW.grid(row = 0, column = 0, sticky=W)
```

```
LOAD = Button(NoL, text="Charger un projet", bg='grey', fg='black', command=load_project)
```

```
LOAD.grid(row = 0, column = 1, sticky=W)
```

```
reinitialisation()          #cette fonction ne fait que réinitialiser les paramètres de prise de vue
```

NoL.mainloop()

#cette dernière ligne empêche la boucle principale de se lancer avant d'avoir choisi entre créer et charger un projet

#elle évite de gros bugs liés aux threads

-Il y a deux variables à mémoriser pour sauvegarder le projet : le chemin et le numéro de la photo.
On va les agencer dans un dictionnaire et les stocker sur un fichier en mode binaire

```
def saveas():
    global frame
    global path
    dico: dict = {}                      #on crée le dictionnaire
    dico["chemin"] = path                 #on enregistre path comme définition du mot « chemin »
    dico["numero_image"] = frame         #on enregistre frame comme définition du mot « numero_image »
    filename = filedialog.asksaveasfilename(defaultextension=".pickle")
    #on demande à l'utilisateur l'emplacement et le nom qu'il veut donner à la sauvegarde
    output_file = open(filename, "wb")
    #on ouvre le fichier qu'on vient de créer et on y stocke les données en mode binaire
    pickle.dump(dico, output_file)
    output_file.close()
    #il ne reste plus qu'à le charger et à reprendre ou on était ;)
```

e) prendre une photo

On a maintenant tous les outils pour commencer notre stop motion

Voici la fonction de base qui sert à prendre la photo :

```
def picture():
    global active
    global frame
    global path
    global image
    if active :
        #on doit s'assurer qu'il n'existe pas déjà un fichier du même nom, ça ferait totalement planter le programme
        os.system('rm -f '+path+'/capt%03d.JPG' %frame)
        #si le liveview est actif, il faut l'arrêter avant de prendre les photos. On est en liaison USB
        #le multi-tâche est très limité donc on peut pas faire le liveview et prendre les photos
        subprocess.run(['killall','gphoto2']) #Cas du Nikon
        subprocess.run(['pkill','-9','gphoto2']) #Cas du Sony et du Canon
        #le SIGKILL fonctionne aussi avec le Nikon mais trop brutal, on entend le miroir qui subit une grosse pression
        #nous pensons que Gphoto2 gère aussi le firmware de l'appareil Nikon (à vérifier) donc on doit fermer une tâche propre à l'APN
        #quand on essaie de le Kill
        #du côté ordinateur, pkill -9 ne pose pas trop de soucis mais sur un Nikon D7XXX, c'est très contraignant
        #en revanche, killall ne fonctionne pas sur les Sony et Canon mais pkill -9 ne leur pose aucun problème
        time.sleep(0.5)
        os.system('gphoto2 --capture-image-and-download --filename '+path+'/capt%03d.JPG' % frame)
        #on prend la photo et on l'enregistre dans le chemin « path » avec le nom « capt » et le numéro de la photo prise « frame »
        os.system('gphoto2 --capture-movie --stdout>fifo.mjpg &')
        #on relance le liveview
        os.system('cp '+path+'/capt%03d.JPG ' %frame +path+ '/prev.JPG')
        #on copie l'image qui vient d'être prise en tant que « prev.JPG » pour le rétro-affichage
    else:
        os.system('rm -f '+path+'/capt%03d.JPG' %frame)
        os.system('gphoto2 --capture-image-and-download --filename '+path+'/capt%03d.JPG' % frame)
        os.system('cp '+path+'/capt%03d.JPG ' %frame +path+ '/prev.JPG')
        frame += 1                      #on incrémente le numéro de la photo
        image = cv2.imread(path+ '/prev.JPG')
        #on met à jour l'image qui se superposera au liveview
        review()
        #on rafraîchit à jour le rétro-affichage
```

Remarques :

-La fonction repicture() qui sert à refaire la photo est basée sur cette fonction picture() précédée d'une décrémentation de la variable frame

-On peut choisir de garder une trace de la photo sur l'APN en rajoutant « --keep » dans la ligne de commande. La version complète du code comprend un moyen de mettre ou non ce paramètre dans les **Settings** du logiciel

-Pour des raisons qui nous échappent, nous ne pouvons pas faire appel à switch() dans la fonction picture().

Nous avons émis l'hypothèse que switch() change l'état booléen de « active » et nous fait quitter la condition avant même de pouvoir effectuer la prise de photo mais cela nous semble improbable car le programme est censé être lu de façon séquentielle.

f) paramètres

Les fonctions servant à modifier les paramètres de prise de vue sont les différentes procédures appelées **callback**.

-Prenons par exemple la liste des ouvertures qu'on a définie précédemment dans la partie sur les dictionnaires :

```
array[int(i)] = f_number_dict['Choice'][i] #array contient donc tous les choix d'ouverture possibles
```

-voici la fonction pour que le choix sur le menu déroulant devienne effectivement l'ouverture de l'appareil :

```
def callback_a(*args):
    global active
    value = a.get()
    # « a » est la variable récupérée à partir du menu déroulant, elle est de type chaîne de caractères
    if active:
        #comme précédemment avec la prise de photo, on doit arrêter le liveview pour pouvoir changer l'ouverture
        subprocess.run(['killall','gphoto2'])
        time.sleep(1)
        os.system('gphoto2 --set-config-value f-number=' + value )
    #on peut changer l'ouverture avec la commande --set-config(-value) f-number
    time.sleep(1)
    os.system('gphoto2 --capture-movie --stdout>fifo.jpg &')
else:
    os.system('gphoto2 --set-config-value f-number=' + value )
```

-du côté interface utilisateur, on a :

```
a = StringVar()      #on définit « a » comme une variable chaîne de caractère
a.set(array[0])       #on initialise la valeur de « a » comme étant la première valeur dans « array »
aperture_menu = OptionMenu(left_frame, a, *array)
# « aperture_menu » est un menu déroulant avec les valeurs contenues dans « array »
aperture_menu.grid(row=1, column=1, sticky=W, padx=10)
a.trace("w", callback_a)
#on rattache enfin « a » à la fonction callback_a() qui saura quoi en faire
```

Certains appareils, dont le Sony A77, vont faire les paramètres par palier, ils vont passer par toutes les valeurs intermédiaires avant d'atteindre la valeur cible ce qui entraîne un temps d'attente qui peut être long. Dans leur cas, la bibliothèque accède aux valeurs par balayage de listes.

g) Contrôle de l'Arduino (côté PC)

-La commande **ls /dev/tty*** liste toutes les liaisons séries sur l'ordinateur. Si un appareil Arduino est connecté, alors on trouvera facilement le port associé.

-Maintenant, on souhaite commander l'arduino avec notre Script python. Pour cela, il nous faut inclure 2 autre bibliothèques

```
from pynput.keyboard import Listener, Key    #écouter le clavier de l'ordinateur
import serial                            #faire la transmission de données vers l'Arduino
```

-On a choisi d'envoyé un seul caractère en écriture binaire à la fois. Chaque caractère code une action à faire pour la carte Arduino :

| Code | Tâche | Code | Tâche |
|------|-------------------------------------|------|--------------------------------|
| 0 | Tilt bas | c | Vitesse de translation à 75 % |
| 1 | Arrêter un mouvement de tilt | d | Vitesse de translation à 100 % |
| 2 | Tilt haut | e | Vitesse de panoramique à 25 % |
| 3 | Panoramique gauche | f | Vitesse de panoramique à 50 % |
| 4 | Panoramique droite | g | Vitesse de panoramique à 75 % |
| 5 | Translation sens direct | h | Vitesse de panoramique à 100 % |
| 6 | Translation sens inverse | i | Puissance de tilt à 25 % |
| 7 | Arrêt d'un mouvement de translation | j | Puissance de tilt à 50 % |
| a | Vitesse de translation à 25 % | k | Puissance de tilt à 75 % |
| b | Vitesse de translation à 50 % | l | Puissance de tilt à 100 % |

-Voici le code pour lier les mouvements de caméra aux touches du clavier :

try:

```
ser = serial.Serial('/dev/ttyUSB1',9600)      # il faut mettre le bon port
ser.timeout=1
```

```
def on_press(key):
    if key == Key.up: # si le bouton directionnel haut est pressé
        print("tilt up")
        ser.write(b'2')
    elif key == Key.down: # si le bouton directionnel bas est pressé
        print("tilt down")
        ser.write(b'0')
    elif key == Key.left: # si le bouton directionnel gauche est pressé
        print("pan left")
        ser.write(b'3')
    elif key == Key.right: # si le bouton directionnel droit est pressé
        print("pan right")
        ser.write(b'4')
    elif key == Key.enter: # si la touche entré est pressée
        print("Go forward")
        ser.write(b'5')
```

```

elif key == Key.shift_r: # si la touche shift de droite est pressé
    print("Go backward")
    ser.write(b'6')

def on_release(key):
    if key == Key.up:
        print("tilt stop")
        for i in range (0,20):
            ser.write(b'1')
    if key == Key.down:
        print("tilt stop")
        for i in range (0,20):
            ser.write(b'1')
    if key == Key.enter:
        print("trans stop")
        for i in range (0,10):
            ser.write(b'7')
    if key == Key.shift_r:
        print("trans stop")
        for i in range (0,10):
            ser.write(b'7')

def ecoute_clavier():
    with Listener(on_press=on_press,on_release=on_release) as listener:
        listener.join()

except:
    print("Micro-contrôleur slider non détecté")

```

Un souci lié à la liaison sérielle fait que l'envoi d'un seul caractère en une fois n'est pas capté par la carte Orion que nous utilisons. Lorsque l'utilisateur reste appuyé, cela ne pose pas de problème car le signal se renvoie plusieurs fois. Par contre lorsqu'il lâche le bouton, il faut envoyer plusieurs fois le signal pour s'assurer que la carte le reçoit correctement.

Remarque :

Par soucis de gain d'espace et de pertinence, les fonctions gérant la puissance et et la vitesse ne sont pas décrites dans ce rapport.

Nous avons découvert la gestion d'erreurs grâce à la fonction « try...except » pendant ce projet :

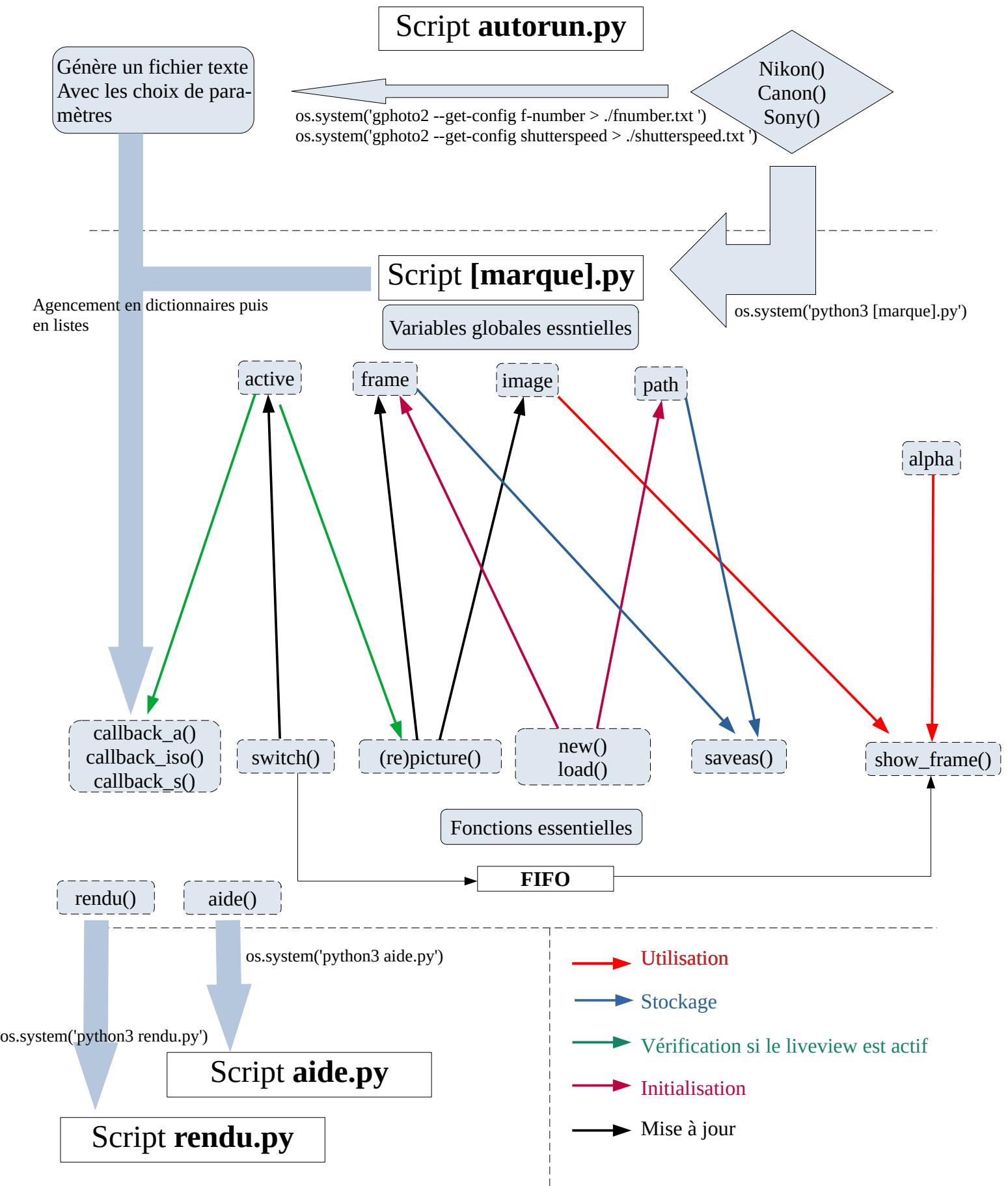
```

try :
    une action
    raise Exception (si la sortie de la commande ne nous convient pas et si la commande le crée
                    pas l'exception automatiquement)
except :
    une autre action

```

Cela nous a permis de passer une commande et éviter de faire planter le programme en cas d'erreur. Cette fonction nous a été très pratique

h) interaction entre les fonctions



i) rendu

-Le script de rendu utilise le fichier de sauvegarde « .pickle » pour retrouver le chemin vers le projet : on ouvre le fichier comme on le fait lorsqu'on veut charger un projet.

-il y a des menus déroulants comme pour les paramètres de prise de vue : codec, framerate, encapsuleur

-On utilise filedialog pour demander à l'utilisateur l'emplacement et le nom de son média final

-Pour faire le rendu, on utilise ffmpeg comme moteur.

```
def rendu():
    global path      #contient le chemin vers les photos
    global fr        #framerate du projet choisi par l'utilisateur
    global encod     #codec du projet choisi par l'utilisateur
    global encap     #format du projet choisi par l'utilisateur
    nom = filedialog.asksaveasfilename(defaultextension=encap)
    os.system("ffmpeg -r "+fr+" -f image2 -i "+path+"/capt%03d.JPG -r "+fr+" -codec:v
              "+encod+" "+nom)
#on envoie la ligne de commande pour faire un rendu de vidéo à partir d'images fixes sous ffmpeg
```

j) Threading

Lorsqu'on travaille avec des boucles et des fonctions récursives ou simplement des processus longs, il faut faire très attention à ce qu'aucune procédure ne gêne le fonctionnement d'une autre ni ne perturbe la boucle principale du programme. Tkinter travaille comme une boucle (**window.mainloop()**) or dans notre script, d'autres fonctions créent une boucle, c'est le cas de **show.frame()** qui définit une récursion, ou **ecoute_clavier()** qui capte en permanence les touches appuyées sur le clavier. La captation de vidéo en elle-même est un processus long à elle seule.

Il faut donc trouver un moyen de les faire fonctionner en arrière plan.

-Dans le cas des fonctions gérées par le script, on peut en faire des **threads**, c'est-à-dire, des unités de traitement indépendants afin qu'elles ne bloquent pas le fonctionnement du reste du programme. Pour ce faire il faut inclure la bibliothèque suivante :

```
import threading
```

Ensuite, on définit lance les procédures en tant que threads :

- 1) **try:**
 threading.Thread(target=ecoute_clavier).start()
 except:
 print(1)
- 2) **threading.Thread(target=show_frame).start()**

-Pour le liveview dont la sortie la fonction est une ligne de commande, il suffit de rajouter une esperluette dans la commande pour le lancer en arrière :

```
os.system('gphoto2 --capture-movie --stdout>fifo.jpg &')
```

k) Autofocus

Pour demander à l'appareil photo de faire le point automatiquement, il faut au préalable activer le mode autofocus en lui précisant le mode adopté si c'est AF-S (Sélectif), AF-A (Automatique) ou AF-C (Continu).

Ensuite il suffit de lancer la mise au point avec la commande --set-config autofocusdrive=1 (Nikon et Canon) ou --set-config autofocus=1 (Sony)

```
def af():
    if active :
        subprocess.call(['killall','gphoto2'])
        time.sleep(1)
        os.system('gphoto2 --set-config autofocusdrive=1')
        time.sleep(3)
        os.system('gphoto2 --capture-movie --stdout>fifo.mjpg &')
    else:
        os.system('gphoto2 --set-config autofocusdrive=1')
        switch()
```

Encore une fois, killall ne fonctionnera pas sur les appareils Sony et Canon, il faudra un SIGKILL (pkill -9) dans leur cas.

Il est à noter que le script que l'on a écrit pour le Sony A77 présente quelques subtilités particulières car la mode choisi est la mise au point continue :

Pour la mise au point automatique, nous utilisons d'abord la commande *--set-config focusmode=3* qui va sélectionner le mode continu de l'autofocus, puis nous utilisons *gphoto2 --set-config autofocus=1* pour activer le mode autofocus en continu (ce n'est pas le mode C de l'autofocus, mais ça va demander à l'appareil de toujours chercher le point, au détriment de toute autre opération).. En sélectionnant le mode continu de l'autofocus, nous pouvons avoir, comme son nom l'indique, une mise au point qui est continuellement mise à jour. Ensuite, si au contraire nous ne voulons désactiver la mise au point automatique, nous appliquons d'abord la commande *--set-config autofocus=0* pour arrêter l'autofocus en continu et conserver le dernier point, puis avec *--set-config focusmode=0*, on désactive totalement l'autofocus et permettons de tourner la bague de mise au point pour faire sa netteté sans avoir peur d'abîmer le moteur autofocus. Il est à noter que, lorsque l'autofocus est actif, un procédé tel que le *global active* fut mis en place afin de le désactiver avant tout opération. En effet, lorsque celui-ci est actif, il bloque toute modification de paramètres.

Il faut faire très attention lorsqu'on choisit de travailler en mise au point continue car elle peut potentiellement poser des problèmes de communication avec l'appareil photo qui est souvent occupé !!

VI-Côté Arduino

Dans notre cas, nous utilisons une carte Orion de Makeblock qui se programme exactement de la même façon qu'une carte Arduino Uno classique car elles sont toutes les deux munies du microprocesseur Atmega328p. Il existe aussi un shield pour une carte Arduino Uno pour l'utiliser comme une carte Orion.

Le microcontrôleur reçoit par transmission série, les codes que le script python envoie et les traduit pour effectuer le travail correspondant.

Pour une liaison correcte, il faut que la fréquence de réception en Bauds soit égale à la fréquence d'émission côté PC, à savoir 9600 Bds.

```
#include "MeOrion.h" //Librairie pour utiliser la board Orion
#include <Wire.h> //Librairie pour une liaison I2C
#include <SoftwareSerial.h> //Librairie pour établir une conversation série

MeStepper stepper(PORT_1); //Indication que le moteur du pan est un stepper, qu'il est connecté
                           au port 1 et qu'il sera nommé stepper
MeDCMotor motor1(M1); //Indication que le moteur du tilt est un moteur continu, qu'il est connecté
                        au port M1 et qu'il sera nommé motor1
MeDCMotor motor2(M2); //Indication que le moteur du transversal est un moteur continu, qu'il est
                        connecté au port M2 et qu'il sera nommé motor2
int trans = 150; //Variable indiquant la vitesse initiale du moteur transversal
int pan = 5; //Variable indiquant le nombre de pas initial à effectuer pour le moteur du pan
int tilt = 50; //Variable indiquant la vitesse initiale du moteur du tilt

//Configurations de base
void setup() {
    Serial.begin(9600); //On lance la liaison série à une vitesse de 9600 Bauds
    stepper.setMaxSpeed(2000); //On indique la vitesse maximale de notre moteur pas à pas
    stepper.setAcceleration(7500); //On indique le niveau d'accélération que prendra notre
                                   moteur pas à pas avant d'atteindre sa vitesse maximale
}

//Boucle du programme
void loop() {
    if(Serial.available()) //On teste si la liaison série est
    {
        char a = Serial.read(); //On stocke les informations lues depuis la liaison série dans une
                               variable caractère "a"
        switch(a) //Selon la valeur de "a", nous appliquons différents cas
        {
            case 'a': //Cas où l'utilisateur a choisi la vitesse transversale à 25%
            trans = 100; //On indique la nouvelle vitesse transversale
            break;
            case 'b': //Cas où l'utilisateur a choisi la vitesse transversale à 50%

```

```

trans = 150; //On indique la nouvelle vitesse transversale
break;
case 'c': //Cas où l'utilisateur à choisi la vitesse transversale à 75%
trans = 200; //On indique la nouvelle vitesse transversale
break;
case 'd': //Cas où l'utilisateur à choisi la vitesse transversale à 100%
trans = 250; //On indique la nouvelle vitesse transversale
break;
case 'e': //Cas où l'utilisateur à choisi la vitesse du pan à 25%
pan = 2; //On indique le nouveau nombre de pas à effectuer
break;
case 'f': //Cas où l'utilisateur à choisi la vitesse du pan à 50%
pan = 5; //On indique le nouveau nombre de pas à effectuer
break;
case 'g': //Cas où l'utilisateur à choisi la vitesse du pan à 75%
pan = 10; //On indique le nouveau nombre de pas à effectuer
break;
case 'h': //Cas où l'utilisateur à choisi la vitesse du pan à 100%
pan = 15; //On indique le nouveau nombre de pas à effectuer
break;
case 'i': //Cas où l'utilisateur à choisi la vitesse du tilt à 25%
tilt = 25; //On indique la nouvelle vitesse du pan
break;
case 'j': //Cas où l'utilisateur à choisi la vitesse du tilt à 50%
tilt = 50; //On indique la nouvelle vitesse du pan
break;
case 'k': //Cas où l'utilisateur à choisi la vitesse du tilt à 75%
tilt = 75; //On indique la nouvelle vitesse du pan
break;
case 'l': //Cas où l'utilisateur à choisi la vitesse du tilt à 100%
tilt = 100; //On indique la nouvelle vitesse du pan
break;
case '4': //Cas où l'utilisateur appuie sur la touche du pan droite
stepper.move(pan); //Le moteur du panoramique effectue le nombre de pas dans le sens
horaire
break;
case '3': //Cas où l'utilisateur appuie sur la touche du pan gauche
stepper.move(-pan); //Le moteur du panoramique effectue le nombre de pas dans le sens
trigonométrique
break;
case '0': //Cas où l'utilisateur appuie sur la touche du tilt bas
motor1.run(tilt); //Le moteur du tilt tourne dans le sens horaire selon la vitesse
sélectionnée
break;
case '2': //Cas où l'utilisateur appuie sur la touche du tilt haut
motor1.run(-tilt-15); //Le moteur du tilt tourne dans le sens trigonométrique selon la
vitesse sélectionnée
break;
case '5': //Cas où l'utilisateur appuie sur la touche du transversal avant

```

```

motor2.run(trans); //Le moteur transversal tourne dans le sens horaire selon la vitesse
électionnée
break;
case '6': //Cas où l'utilisateur appuie sur la touche du transversal arrière
motor2.run(-trans); //Le moteur transversal tourne dans le sens trigonométrique selon la
vitesse sélectionnée
break;
case '1': //Cas où les touches tilt haut ou bas sont relâchées
motor1.stop(); //Le moteur du tilt arrête de tourner
break;
case '7': //Cas où les touches transversale avant ou arrière sont relâchées
motor2.stop(); //Le moteur du pan arrête de tourner
break;
}
}
stepper.run(); //ré-arme/actualise le stepper moteur du pan
}

```

Remarque : Dans un premier temps, nous initialisons les vitesses pour chacun des moteurs.

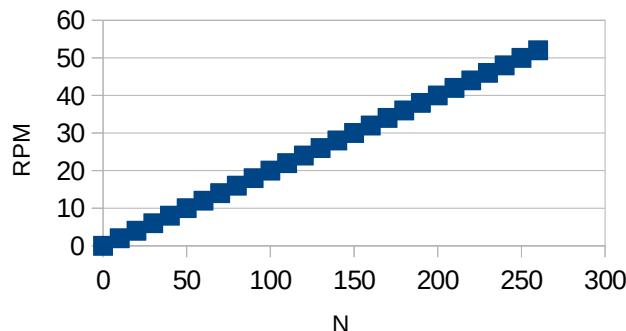
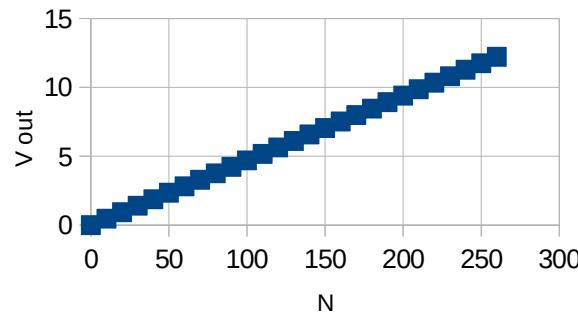
Les vitesses pour les moteurs DC sont codées sur 8 bits, autrement dit, la valeur minimale théorique est de 0 et la valeur pleine échelle est de 255.

Soit U la tension maximale requise par le moteur DC (dans notre cas 12 V)

$$\text{Le quantum est } q = \frac{U}{2^8} = 0,047 \text{ V} \quad \text{et} \quad v = \frac{50 \text{ rpm}}{2^8} = 0,2 \text{ rpm}$$

Avec N le mot code qu'on applique, on a la fonction de transfert suivante :

$$V_{\text{OUT}} = 0,047 \text{ V} * N \quad \text{Vitesse} = 0,2 \text{ rpm} * N$$



Conclusion

Ce projet a été très enrichissant pour nous car il a été l'occasion de mettre en pratique toutes les connaissances en programmation et en électronique. On a rencontré pas mal de problèmes tout au cours de la réalisation de notre système et on a essayé d'apporter des solutions le plus efficacement possible. On a pu revoir et approfondir la programmation en langage Python qu'on a abordée en fin de Licence 1 et on a mis en pratique la programmation du module Arduino dont on continue encore d'apprendre les subtilités et les possibilités.

On a développé un prototype qui bien-sûr peut encore être amélioré. Dans un premier temps, nous avions choisi de travailler en Python pour la vitesse d'exécution ainsi que pour la large gamme de bibliothèques que le langage offre. De plus, il nous a été conseillé d'utiliser ce langage car c'est un moyen rapide de prototyper un logiciel. Cependant, on s'est heurtés plus tard aux limites du langage car les déclinaisons de la librairie « libgphoto2 » pour python ne sont plus vraiment à jour et souffrent d'un gros manque de documentation. Néanmoins, le logiciel fonctionne assez bien dans l'état actuel. Pour aller plus loin et faire un logiciel plus performant, il serait plus avisé d'écrire le programme en C/C++ ou en JAVA qui possèdent la librairie à jour et bien documentée.

Un autre aspect qu'on aurait aimé améliorer si on avait le temps est la fluidité du mouvement pour permettre un travelling « smooth » tout en prenant des photos. De plus, l'utilisation du système, pour le moment, n'est pas très intuitive, il faut prendre le temps de s'y habituer. Même pour nous, les mouvements de caméra n'ont pas été très faciles à utiliser au début. Pour l'améliorer, ce serait bien d'utiliser un support mécanique spécialement fait pour accueillir l'appareil ainsi que les cartes Arduino et Raspberry car le prototype met en œuvre beaucoup de solutions assez rudimentaires.

Pour résumer, grâce à ce projet nous avons pu expérimenter une première approche d'un projet d'ingénierie. On a fait énormément de rétro-ingénierie sur des logiciels tels que Entangle ou Linux StopMo... La documentation et les recherches sur les projets similaires déjà menés ont occupé une grande partie de notre projet. On a fait énormément de changement sur notre système avant d'en arriver à ce que nous vous présentons ici. Pour un prototype, nous estimons que le système fonctionne assez bien, mais force est de constater qu'il y a encore beaucoup à améliorer.

Merci d'être nos bêta-testeurs aujourd'hui :)

Annexe

Matériel utilisé

| | |
|--|---|
| <p>Raspberry Pi 3B+ (ou plus récent)</p> <p>1.4GHz 64-bit quad-core processor, dual-band wireless LAN, Bluetooth 4.2/BLE, faster Ethernet, and Power-over-Ethernet support (with separate PoE HAT)</p> |  |
|  | <p>Carte Orion (ou Arduino Uno avec shield Orion)</p> <ul style="list-style-type: none">-Alimentation: 7 - 12 V-Connectique pour programmation : micro USB-Microcontrôleur: ATmega328-Mémoire Flash 32 KB (ATmega328)-SRAM: 2.5 KB)-EEPROM: 1 KB-Clock Speed 16 MHz |
| <p>Nikon D7000</p> <p>Canon EOS 1200D</p> <p>Sony Alpha 77</p> |  |
|  <h1>Linux</h1> | <p>Un ordinateur qui tourne sous une distribution Linux/UNIX</p> |
| <p>Un kit de robotique Makeblock</p> <p>2 moteurs à courant continu 12V – 50 rpm 1 moteur pas à pas avec un driver Pièces mécaniques</p> |  |

Installation de Raspbian

Site : raspberrypi.org/software/

Download du software **Raspberry Pi Imager**
Installation sur Micro SD du **Raspberry Pi OS**

Mise à jour de l'OS, dans le terminal : sudo apt-get update
sudo apt-get upgrade

Installation manuelle de Gphoto2 et de libgphoto2

Installation des packages permettant la compilation de gphoto2 :
sudo apt-get install git make autoconf libltdl-dev libusb-dev libexif-dev libpopt-dev libxml2-dev
libjpeg-dev libgd-dev gettext autopoint

Import du code de la librairie libgphoto2:
sudo git clone <https://github.com/gphoto/libgphoto2.git>

Compilation de la librairie libgphoto2 :

```
cd ~/libgphoto2
sudo autoreconf --install --symlink
sudo ./configure
sudo make
sudo make install
```

Clonage code du logiciel de gphoto2 sur Raspberry Pi
cd ~ git clone <https://github.com/gphoto/gphoto2.git>

Compilation du logiciel de gphoto2 :

```
cd ~/gphoto2
sudo autoreconf --install --symlink
sudo ./configure
sudo make
sudo make install
```

Vérification du répertoire de la librairie :

```
sudo nano /etc/ld.so.conf.d/libc.conf
# libc default configuration /usr/local/lib
```

Rafraîchissement de la mémoire de cache :
sudo ldconfig

Génération des règles udev :

```
/usr/local/lib/libgphoto2/print-camera-list udev-rules version 201 group plugdev mode 0660 |
sudo tee /etc/udev/rules.d/90-libgphoto2.rules
```

Génération du dossier de base de données hardware pour udev :

```
/usr/local/lib/libgphoto2/print-camera-list hwdb | sudo tee /etc/udev/hwdb.d/20-gphoto.hwdb
```

Spécifications techniques ORION

Makeblock Orion Main Control Boards

Overview

Makeblock Orion is a main control board upgraded and improved for teaching and entertainment on the basis of Arduino Uno. With powerful driving ability and maximum output power of 36W (3A), it can drive four DC motors simultaneously. Well-designed color system is used with sensor modules perfectly, and eight user-friendly independent RJ25 ports implements circuit connection easily. In addition, it supports most Arduino programming tools (Arduino/ArduBlock), and provides the GUI programming tool (mBlock) upgraded from Scratch and mobile APP to meet the needs of various users.

Technical specifications

- Output voltage: 5V DC
- Input voltage: 6V-12V DC
- Maximum input current: 3A
- Communication mode: UART, I²C,digital I/O, analog input
- Control chip: Atmega 328P
- Dimension: 80 x 60 x 18 mm (L x W x H)

Functional characteristics

- Easy to connect with a variety of sensors, electronic modules, and drive modules
- Support the DC motor, stepper motor, servo driver, and encoder motor driver
- Drive two motors directly
- Supply 5V voltage and 2A current
- Onboard buzzer
- Over-current and over-voltage protection
- Fully compatible with Arduino
- Easy to use RJ25 cable
- Provide specific Arduino library functions in Makeblock to simplify programming
- Support mBlock (upgrade of Scratch 2.0) and applicable to users of all ages

Introduction to ports

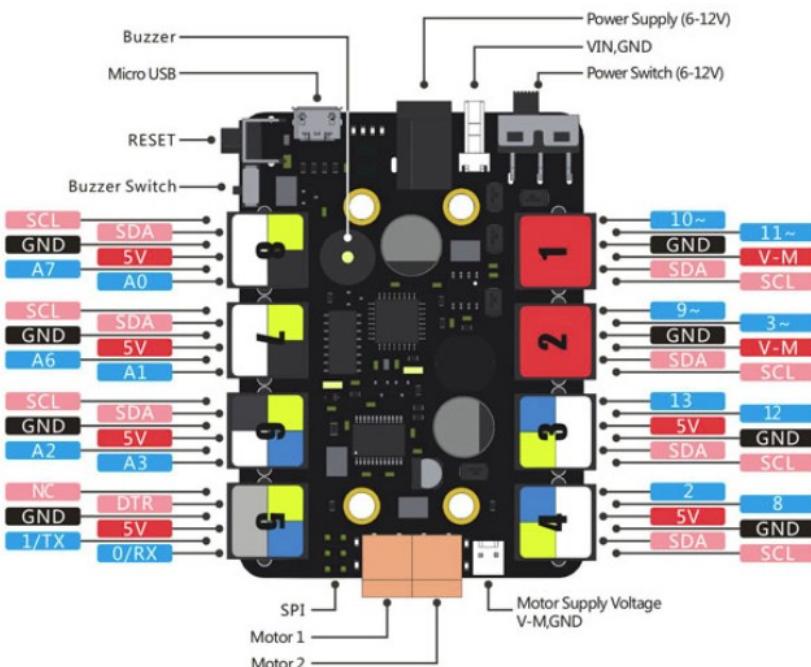
Makeblock Orion provides eight RJ25 ports identified by labels of six different colors. Their colors and functions are as follows:

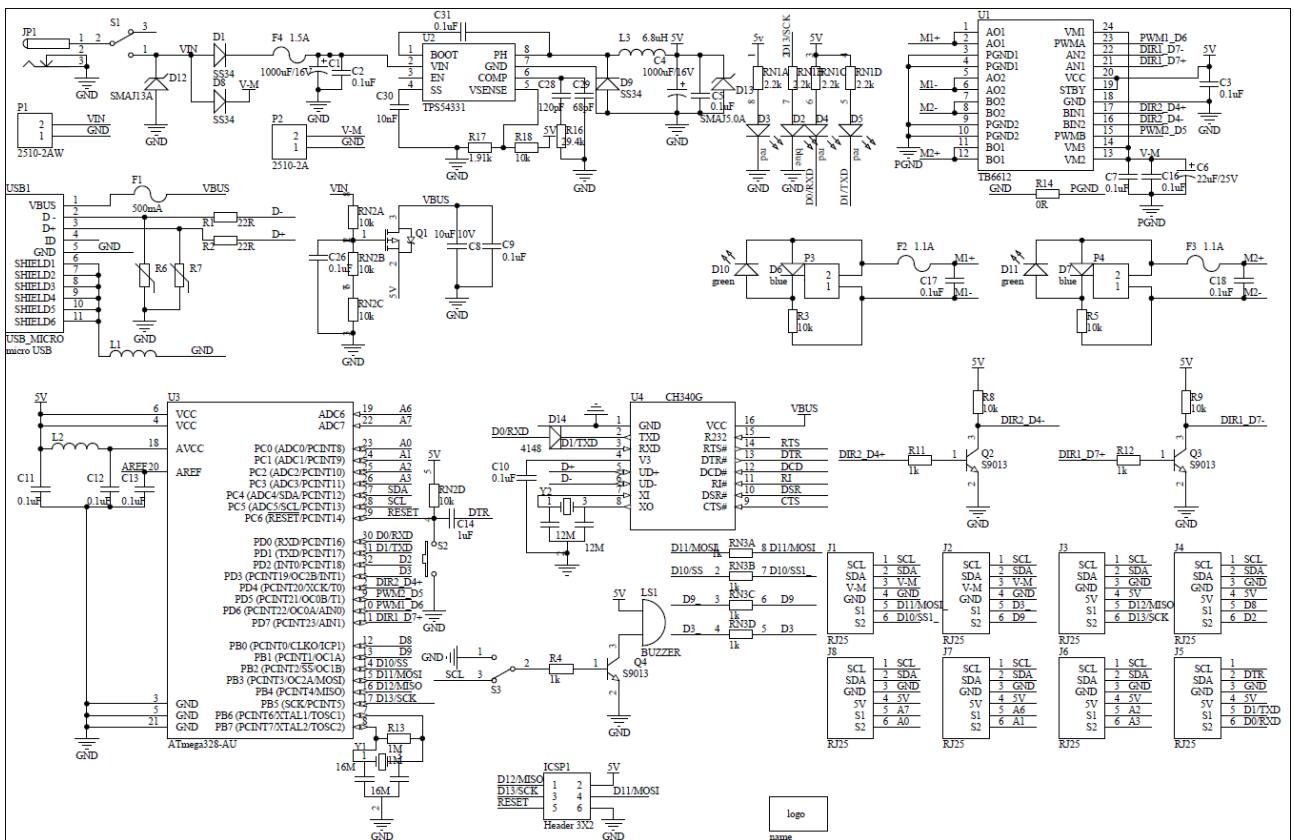
| Color | Function | Module using this port |
|-------|--|--|
| | Red means the output voltage is 6-12V, and it is usually connected to the motor driver module of 6-12V voltage | <ul style="list-style-type: none">• Me DC Motor Driver• Me Stepper Motor Driver• Me Encoder Motor Driver |
| | Single-digital port | <ul style="list-style-type: none">• Me Ultrasonic Sensor• Me RGB LED |
| | Double-digital port | <ul style="list-style-type: none">• Me 7-Segment Display• Me PIR Motion Sensor• Me Shutter Cable• Me Line Follower• Me IR Receiver |
| | Serial port of hardware | <ul style="list-style-type: none">• Me Bluetooth Module (Dual Mode)• Me WiFi Module |
| | Analog signal port | <ul style="list-style-type: none">• Me Light Sensor• Me Potentiometer• Me Joystick• Me 4-Button• Me Sound Sensor |
| | I ² C port | <ul style="list-style-type: none">• Me 3-Axis Accelerometer and Gyro Sensor• Me Compass |

The output voltages of yellow, blue, gray, black, and white ports are constant 5V DC current. These ports are usually connected to

| Port No. | Color | Type of compatible modules | Modules using this port |
|----------|-------|---|---|
| 1 & 2 | 2 1 | (6-12VDC) driver module | <ul style="list-style-type: none"> • Me Dual Motor Driver • Me Stepper Motor Driver • Me Encoder Motor Driver |
| 3 & 4 | 4 3 | Single-digital port Double-digital port I ^C port | <ul style="list-style-type: none"> • Me Ultrasonic Sensor • Me RGB LED • Me Limit Switch • Me 7-Segment Display • Me PIR Motion Sensor • Me Shutter • Me Line Follower • Me IR Receiver • Me 3-Axis Accelerometer and Gyro Sensor |
| 5 | 5 | Single-digital port Double-digital port Serial port of hardware | <ul style="list-style-type: none"> • Me Ultrasonic Sensor • Me RGB LED • Me Limit switch • Me 7-Segment Display • Me PIR Motion Sensor • Me Shutter Cable • Me Line Follower • Me IR Receiver • Me Bluetooth Module (Dual Mode) • Me TFT LCD Screen |
| 6 | 6 | Single-digital port Double-digital port I ^C port Analog signal port | <ul style="list-style-type: none"> • Me Ultrasonic Sensor • Me RGB LED • Me Limit Switch • Me 7-Segment Display • Me PIR Motion Sensor • Me Shutter • Me Line Follower • Me IR Receiver • Me 3-Axis Accelerometer and Gyro Sensor • Me Potentiometer • Me Joystick • Me 4-Button • Me Sound Sensor |
| 7 & 8 | 8 7 | Single-digital port I ^C port Analog signal port | <ul style="list-style-type: none"> • Me Ultrasonic Sensor • Me RGB LED • Me Limit Switch • Me Potentiometer • Me Joystick • Me 4-Button • Me Sound Sensor • Me 3-Axis Accelerometer and Gyro Sensor |

modules of 5V supply voltage.





Fiche de suivi

DOSSIER DE PROJET LICENCE 3 [Type 2]

- à conserver tout au long de l'avancement du projet -

1. GENRE (Tout projet audiovisuel qui ne soit pas un projet « type 1 »
(Projet technique, informatique, stop motion, animation pâte à modeler, clip etc....))
Projet Ingénierie AV – Vidéo

2. ENCADRANT PEDAGOGIQUE (signature obligatoire avant la soutenance de pitch)

Nom et prénom : Gauthier Sébastien

Signature : 

Adresse mail : sebastien.gauthier@uphf.fr
Contact téléphonique : 03.27.51.15.12

3. ETUDIANT REFERENT DU PROJET :

Nom : ANDRIAMANALINA Rivo

Signature : 

4. DESCRIPTIF DU PROJET

Créer un dispositif de cadrage "semi-automatique" (car nécessité de la présence d'un opérateur)

- Notre but est de permettre le cadrage à distance avec un APN
- faire un RCP qui soit compatible avec tout APN sorti depuis 2012
- si le temps nous le permet**, on aimerait ajouter une fonctionnalité pour tracker la position et les mouvements du sujet, ainsi le cadrage serait totalement autonome.

5. EQUIPE DU PROJET L3 Type 2 :

| NOM | PRENOM | FONCTION | TELEPHONE | ADRESSE MAIL |
|----------------|--------|----------|----------------|-----------------------------|
| ANDRIAMANALINA | Rivo | | 06.10.14.15.15 | rivo.tovonavalona@gmail.com |
| COUARD | Théo | | 06.04.09.92.81 | theo.couard@hotmail.fr |
| HUFTIER | Lucas | | 06.40.05.39.36 | lucas.huftier@gmail.com |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |

6. DEFINITION DES ETAPES DE TRAVAIL

- Construction maquette
- Création du logiciel/Code
- Test(s)
- Rectification des éventuelles erreurs

7. ESTIMATION DES DUREES DE CHAQUE ETAPÉ (en journées):

20 à 30 jours par étape

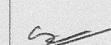
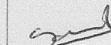
8. PLANNING PREVISIONNEL :

| ETAPE | Travail prévu | Matériel utilisé | Périodes | |
|---------|----------------------------------|--|----------|-----|
| | | | DEBUT | FIN |
| ETAPE 1 | familiarisation avec graphique 2 | ordinateurs, appareils photos/ Raspberry | | |
| ETAPE 2 | construction programme | ordinateurs/ appareils photos | | |
| ETAPE 3 | interface graphique | ordinateurs | | |
| ETAPE 4 | maquette + slider | pièces métalliques, tourneurs, etc. | | |
| ETAPE 5 | finition et correctifs | ordinateurs appareils photos | | |
| ETAPE 6 | | | | |

10. FICHE DE SUIVI (TYPE 2)

Nom de l'encadrant pédagogique principal (rappel) : SÉBASTIEN GAUTHIER.....

Nom des autres enseignants éventuellement consultés sur des points particuliers : PHILIPPE.....
THOMAS, NICOLAS, VIEVILLE.....

| Réunion Statut | Réunion Objet | Date conseillée au plus tard | Date effective | Mise au point de l'avancée (remplie par les étudiants) | Remarques adressées par l'encadrant | Signature de l'encadrant |
|---------------------------|--------------------------------|------------------------------|----------------|--|-------------------------------------|---|
| Réunion(s) facultative(s) | Expliq' sujet | 04/11/2011 | | choisie d'un sujet | |  |
| Réunion obligatoire | Définition du cadre de travail | Mi janvier au plus tard | | projet remis PSCR | |  |
| Réunion(s) facultative(s) | matiriel | 17/01/2011 | | obtention de matériel (vitres métalliques) | |  |
| Réunion obligatoire | Mise au point à mi parcours | Mi mars | | Avancée des codes et programmes + maquette | |  |
| Réunion(s) facultative(s) | code, matériau | 22/03/2011 | | | |  |
| Réunion obligatoire | Examen final du projet | Mi mai | | test vidéos, stop-motion... | |  |
| Rendu du projet | | Fin mai | | | | |

Webographie

| | |
|-------------------------------|---|
| gPhoto2 | http://www.gphoto.org/doc/manual/using-gphoto2.html |
| Python-tkinter | https://docs.python.org/fr/3/library/tkinter.html |
| Python-subprocess | https://docs.python.org/3/library/subprocess.html |
| Python-OS | https://docs.python.org/fr/3/library/os.html |
| Python-Dialog | https://docs.python.org/3/library/dialog.html |
| Python-Pillow | https://pillow.readthedocs.io/en/stable/reference/Image.html |
| OpenCV | https://pypi.org/project/opencv-python/ |
| Piggyphoto | https://github.com/alexdu/piggyphoto |
| Python-gPhoto2 | https://github.com/jim-easterbrook/python-gphoto2 |
| Entangle | https://gitlab.com/entangle/entangle |
| Orion Makeblock | https://www.makeblock.com/project/makeblock-orion |
| Makeblock libraries | https://github.com/Makeblock-official/Makeblock-Libraries |
| Makeblock | https://www.makeblock.com/ |
| Libgphoto2 (C++) | https://github.com/gphoto/libgphoto2 |
| Signaux de terminaison | https://sites.uclouvain.be/SystInfo/notes/Theorie/html/Fichiers/fichiers-signaux.html |
| Python-trace | https://docs.python.org/3/library/trace.html |
| Gstreamer | https://gstreamer.freedesktop.org/ |
| Nikon D7000 | https://www.nikon.fr/fr_FR/product/discontinued/digital-cameras/2015/d7000 |
| Sony A77 | https://www.01net.com/tests/sony-alpha-a77-mark-ii-fiche-technique-23709.html |
| Canon EOS 1200D | https://www.canon.fr/for_home/product_finder/cameras/digital_slr/eos_1200d/ |