

PROJET ARDUINO - ELEC4

Année scolaire 2018-2019

CONTROLEUR À DISTANCE

Etudiants : Léo MARACHE, Oualid BEN MOHAMED

Encadrants : Pascal MASSON, Nassim ABDERRAHMANE

SOMMAIRE

Introduction	4
Chapitre I : Cahier des charges.....	5
I.1. Objectifs	5
I.2. Matériel	6
Chapitre II : Fonctionnement global.....	8
II.1 Fonctionnement de la partie gestuelle	8
II.2 Fonctionnement de la partie Bluetooth	Erreur ! Signet non défini.
Chapitre III : Plannings	13
III.1 Planning initial	13
III.2 Planning final	13
Conclusion.....	15
Bibliographie	16

Introduction

Dans le cadre du module d'électronique de peip2, il nous a été demandé de réaliser un projet utilisant nos connaissances d'Arduino. Nous nous sommes donc tout d'abord interrogés sur le sujet que nous allions choisir. Dans un premier temps, notre binôme a réfléchi aux possibles applications du matériel déjà disponible. Nous avions notamment à notre disposition un certain nombre de moteurs, des écrans ou encore des capteurs à ultrasons. Nous nous sommes rapidement mis d'accord pour ne pas réaliser de voitures télécommandées, cette thématique ayant été décliné à de multiples reprises. D'autre part nous voulions choisir un sujet sur lequel l'industrie travaille actuellement.

Aussi nous nous sommes intéressés au secteur des téléviseurs qui voit se développer depuis quelques années la reconnaissance gestuelle. En effet de nos jours, certains peuvent être contrôlés à partir de gestes réalisés par le téléspectateur. Typiquement, on peut désormais changer de chaîne d'un revers de main. Samsung, qui a lancé la fonctionnalité sur le marché, met en avant le pragmatisme du service. Que ce soit en pleine préparation d'un repas ou lorsque la télécommande n'est pas à proximité, il est pratique de pouvoir tout de même contrôler la télévision.

Ainsi nous avons opté pour construire une fonctionnalité similaire. Dès le début, il nous a paru évident d'utiliser les capteurs à ultrasons déjà rencontrés pour la reconnaissance gestuelle. Bien que notre technologie n'ait rien à voir avec ce que propose l'industrie, nous étions très motivés par la finalité, pouvoir nous aussi contrôler un système à partir de gestes simples.

L'idée du projet définie, il nous fallait maintenant étudier sa faisabilité. L'un des points critiques sur lequel nous devions tout de suite nous renseigner était celui de la programmation du système. Nous avons très vite compris que nous ne pourrions pas reprogrammer le logiciel d'un téléviseur. Ce dernier est un logiciel propriétaire et donc fermé. Mais existe-t-il cette possibilité sur les systèmes d'exploitation de nos ordinateurs ? La réponse est positive. À l'aide de commandes dédiées nous pouvons contrôler certaines fonctions basiques d'un PC.

Notre objectif esquisssé, il nous a fallu alors nous attarder sur l'obligation d'utiliser des communications radio fréquence. Assez logiquement, nous avons fait le lien avec une autre tendance sur le marché des téléviseurs : le déploiement des applications mobiles type télécommandes TV. En effet, la majorité des marques proposent de nos jours des applications dédiées pour contrôler leurs téléviseurs. Cela s'explique par leur récente mise en réseau. Avec l'arrivée d'internet dans nos salons, les services TV en ligne ont connu une forte progression ces dernières années si bien que les menus sont plus nombreux et complexes que jamais. Or les applications mobiles sont bien plus commodes qu'une télécommande traditionnelle pour naviguer dans ces menus et y faire une recherche. C'est pourquoi nous avons décidé de réaliser une télécommande pour smartphone fonctionnant via Bluetooth avec un PC. L'objectif final étant de contrôler la souris via son smartphone. Cela constitue donc la deuxième partie du projet.

Chapitre I : Cahier des charges

I.1. Objectifs

L'objectif de notre projet est de pouvoir contrôler un ordinateur à distance, donc sans utiliser ni le clavier ni la souris. Pour parvenir à nos fins, nous avons développé deux solutions.

La première consiste à réaliser des gestes simples détectables par deux capteurs à ultrasons. Cette détection se fait par rapport au nombre de mains détectées et de leur distance par rapport aux capteurs. Avec cette solution nous avons accès aux mêmes fonctionnalités offertes par le clavier. C'est-à-dire que l'on peut assimiler à un geste n'importe quel raccourci clavier.

Mais se pose alors le problème du nombre de gestes distinguables. En effet, l'interprétation d'un geste se basant sur les seuls critères du nombre de mains et de leur éloignement, on ne peut qu'en différencier qu'un effectif limité. De plus, dans cette version, nous ne pouvons pas déplacer la souris.

Aussi, toute application de cette alternative est limitée à un cadre précis. Il faut vouloir contrôler certaines fonctionnalités en particulier et au sein d'un même logiciel. Ainsi un cadre parfaitement approprié à cette alternative est celui d'un logiciel de lecture d'objets multimédia. Finalement, la première partie de notre projet consiste à contrôler les paramètres principaux d'un logiciel de lecture vidéo/musique. Ces derniers sont :

- Mettre sur pause ou lancer la lecture de l'objet multimédia
- Reculer ou avancer dans sa lecture
- Diminuer ou augmenter son volume sonore

On a donc accès à ces fonctionnalités à l'aide des trois gestes ci-dessous.



Figure I.1.1. Geste pour mettre sur pause ou lancer la lecture de l'objet multimédia.

Figure I.1.2. Geste pour rembobiner ou accélérer sa lecture.

Figure I.1.3. Geste pour abaisser ou augmenter son volume sonore.

La deuxième solution consiste à utiliser un smartphone Android relié par Bluetooth au module HC-06. Nous avons donc logiquement opté pour l'application Bluetooth Electronics déjà étudiée. Dans cette version, l'utilisateur peut donc accéder à son ordinateur même s'il en est éloigné de 10 mètres. De plus, nous pouvons cette fois manipuler à distance la souris en plus du clavier. Cette solution offre donc plus de possibilités d'applications même si comme nous le verrons plus tard, on ne peut déplacer la souris que de manière ponctuelle et de fait sans grande précision.



Figure I.1.4. Notre télécommande Bluetooth pour ordinateur.

I.2. Matériel

Notre projet nécessite peu de matériel et a l'avantage de ne pas exiger des pièces supplémentaires par rapport à celles déjà disponibles en cours d'Arduino.

On a donc besoin :

- D'un ordinateur
- D'une carte Arduino Uno
- D'une plaque de test
- De deux capteurs à ultrasons HC-SR04
- Du module Bluetooth HC-06
- D'un smartphone Android

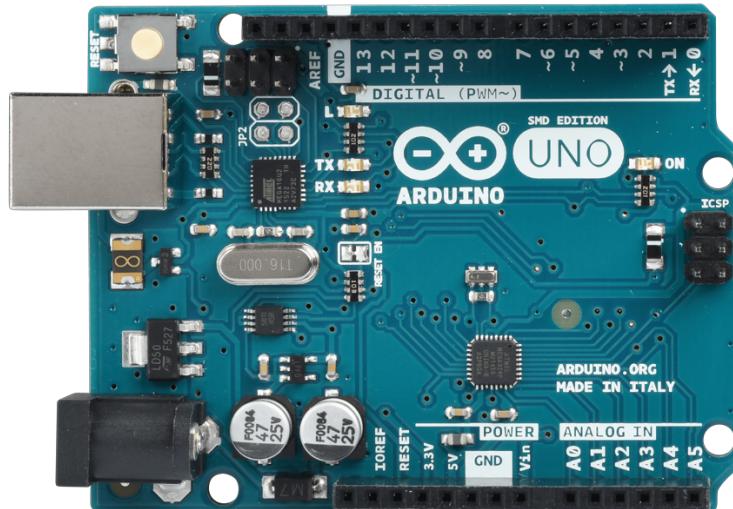


Figure I.2.1. Carte Arduino Uno.



Figure I.2.2. *Module HC-SR04.*

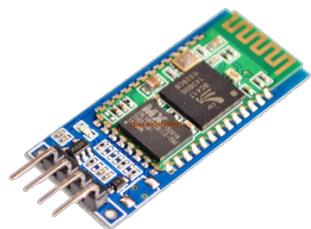


Figure I.2.3. *Module Bluetooth HC-06.*

Chapitre II : Fonctionnement global

Expliquons maintenant comment fonctionne globalement le projet.

Il requiert deux algorithmes qui tournent simultanément :

- Le premier concerne l'Arduino. Il permet dans sa première version de détecter les gestes que l'utilisateur réalise devant les capteurs à ultrasons. Quant à sa variante pour la seconde partie du projet, il interprète cette fois les données reçues par l'application Bluetooth Electronics. Dans les deux cas, il informe un second algorithme.
- Ce dernier est un script Python. Pourquoi ? Parce que Python possède une librairie *pyautogui* qui permet de simuler des actions sur le clavier et pavé tactile. En fonction des données envoyées par l'Arduino, ce programme appelle la fonction correspondante de *pyautogui*. C'est donc cet algorithme qui permet de contrôler le système d'exploitation de l'ordinateur.

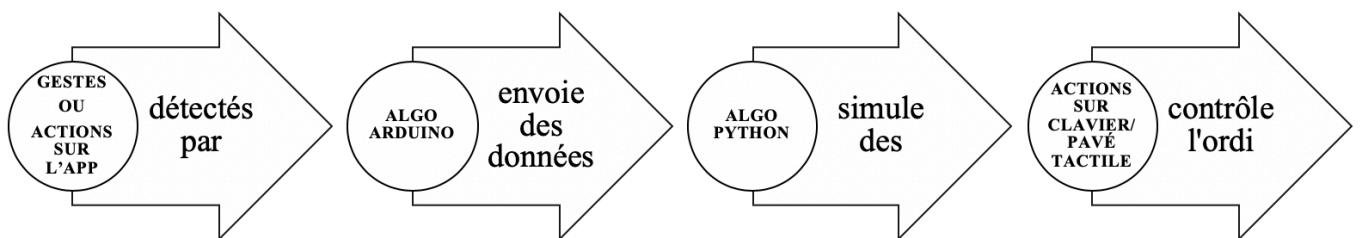


Figure II. Schéma du fonctionnement global du projet.

II.1. Fonctionnement de la partie gestuelle

II.1.1. Le programme Arduino

Comme expliqué dans le chapitre I, on a choisi d'interpréter trois gestes. Pour ce faire, on calcule en permanence la distance séparant les mains des capteurs. Pour le calcul des distances on utilise la fonction *ping_cm* de la librairie *NewPing*.

On distingue alors trois cas :

- Les distances calculées des deux capteurs (respectivement notées G et D dans la **Figure II.1.1**) sont comprises entre un et quinze centimètres. Cela signifie que l'utilisateur souhaite mettre sur pause ou lancer la lecture du média. On envoie alors au script python la chaîne de caractère “PLAY/PAUSE” via la fonction *print*.
- La distance calculée du module de droite (D) est comprise entre quinze et trente-cinq centimètres. À priori l'utilisateur s'apprête à changer le volume sonore. Mais ce n'est pas forcément le cas. Par exemple l'utilisateur peut-être simplement déplacé un objet devant le capteur. Ainsi pour ne pas mal interpréter un geste, on vérifie à nouveau cette distance après une demi-seconde. Si elle est toujours inférieure à trente-cinq centimètres, c'est qu'il s'agit bien de modifier le volume sonore. Pour que l'utilisateur puisse alors ajuster de manière continue la position de sa main, on la calcule de nouveau au sein d'une boucle. Si D est inférieure à vingt-cinq centimètres on envoie au script Python la chaîne de caractères “VOL--” et “VOL++” si elle y est supérieure.
- La distance calculée du module de gauche (G) est comprise entre quinze et trente-cinq centimètres. On suit alors les mêmes étapes que pour le module de droite. La seule différence étant les chaînes de caractères envoyées qui sont dans ce cas “REWIND” et “FORWARD”.

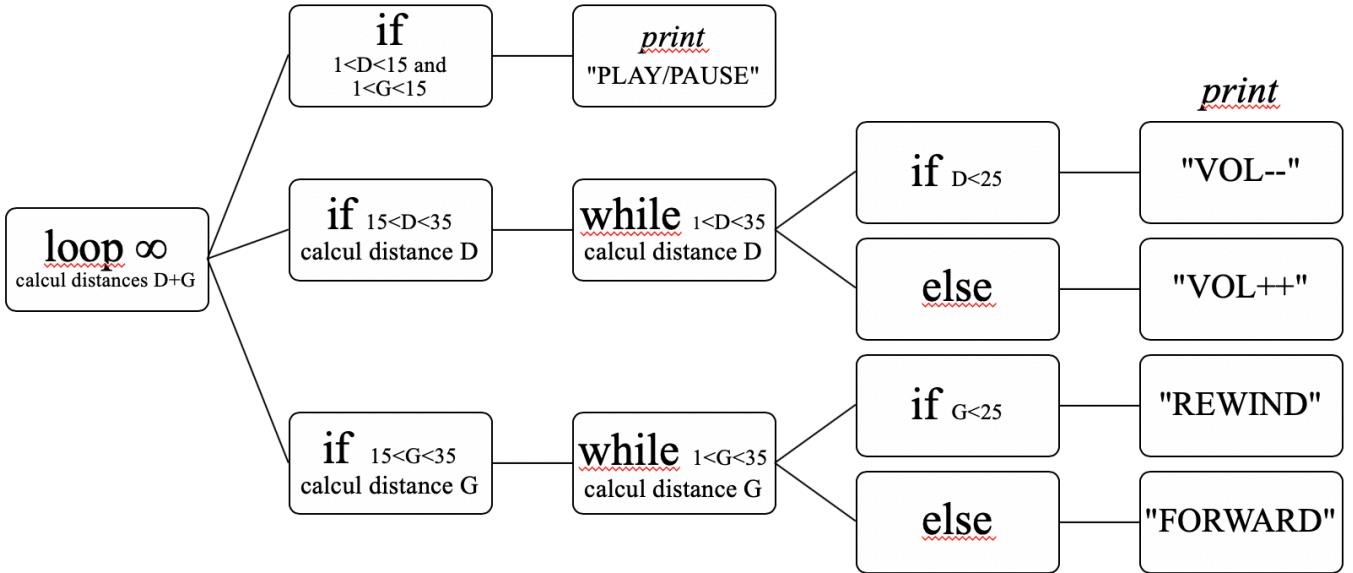


Figure II.1.1. Schéma de l'algorithme Arduino.

Nous avons volontairement distingué la plage de distances dédiée à la reconnaissance du premier geste des autres pour éviter de mal interpréter les intentions de l'utilisateur. En effet, il nous est impossible de ramener nos mains devant les capteurs de manière parfaitement synchrone. De fait, l'une des mains est forcément repérée en première et déclenche à tort l'envoie des chaînes "VOL++"/"VOL--" ou "REWIND"/"FORWARD".

Notons aussi que les gestes doivent être séparés. Prenons un exemple. Si l'on veut augmenter le son avec la main droite puis ramener l'autre face au capteur de gauche pour mettre sur pause, cela ne fonctionne pas. En effet, la main droite ayant été détectée, la condition de la boucle while est vérifiée. On ne peut donc pas en sortir. On doit d'abord baisser la main droite et après seulement ramener les deux.

Nous avons bien sûr développé une version qui résout ce problème mais nous ne l'avons pas gardé car elle alourdissait inutilement le code. Surtout qu'après réflexion, cette fonctionnalité paraît superflue. L'utilisateur a plus tendance à effectuer des gestes rapides et pratiques pour lui. Laisser sa main droite en l'air puis ramener sa main gauche semble peu intuitif.

II.1.2. Le programme Python

Comment est-ce que le script Python fait-il pour recevoir des données du programme Arduino ? En fait il lit simplement le contenu des *print* d'arduino sur le port série grâce à la fonction python *readline*. On affecte à une variable le résultat de cette fonction. On vérifie alors quelle chaîne de caractères vient d'être reçue et l'on fait appel à la fonction *press*. C'est elle qui simule une pression sur une touche du clavier. Prenons un exemple, à la réception de "PLAY/PAUSE", on fait l'appel de fonction suivant : *pyautogui.press("space")*. En effet la touche "space" est dédiée à la mise en pause ou lancement de la lecture d'un objet multimédia.

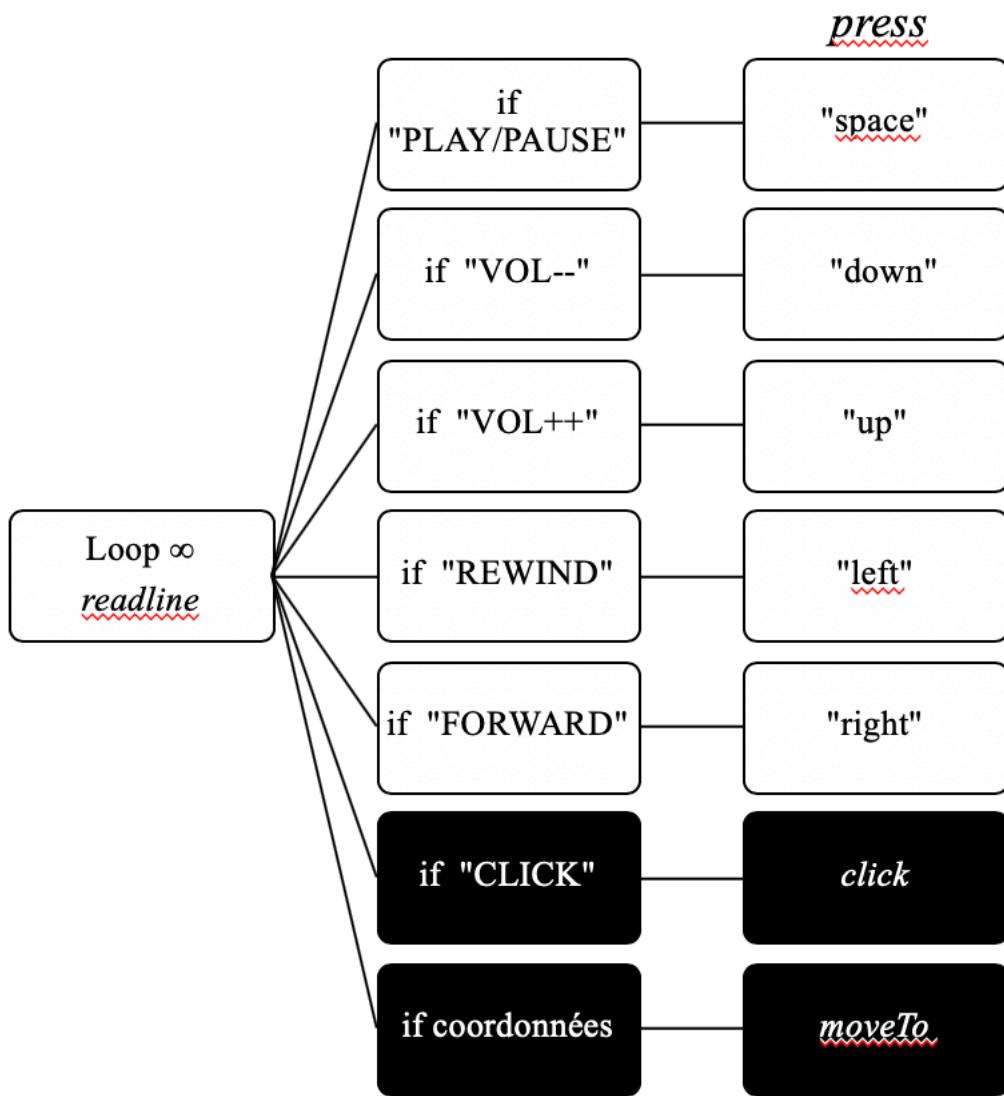


Figure II.1.2. Schéma de l'algorithme Python.

II.2. Fonctionnement de la partie Bluetooth

Une fois la première partie du projet achevée, on a attaqué celle qui consiste à utiliser un smartphone pour contrôler l'ordinateur.

II.2.1. Le programme Arduino

Dans cette version, nous pouvons toujours modifier les trois paramètres de la partie gestuelle, mais aussi déplacer la souris et cliquer. La télécommande Bluetooth que nous avons développée compte ainsi six boutons :

- Un pour mettre sur pause ou lancer la lecture
- Un pour diminuer le volume
- Un pour augmenter le volume
- Un pour reculer dans la lecture
- Un pour avancer dans la lecture
- Et un pour cliquer

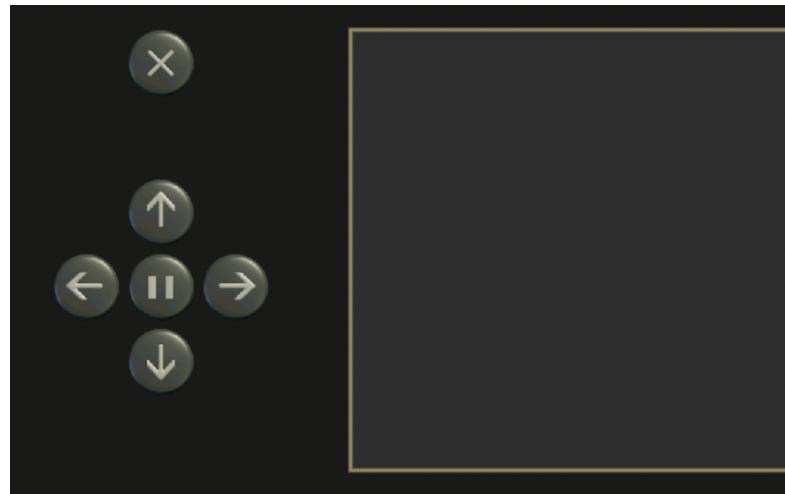


Figure II.2.1.1. La télécommande Bluetooth.

Ces six boutons marchent de la même manière. Lorsqu'ils sont touchés, l'application Bluetooth Electronics envoie un message au module Bluetooth. Ce dernier le transmet à la carte Arduino qui le communique au programme Arduino via un port série. Dans notre cas, ce message est simplement un caractère. Enfin, en fonction du caractère reçu, l'algorithme ci-dessous envoie à son tour le message associé au script Python.

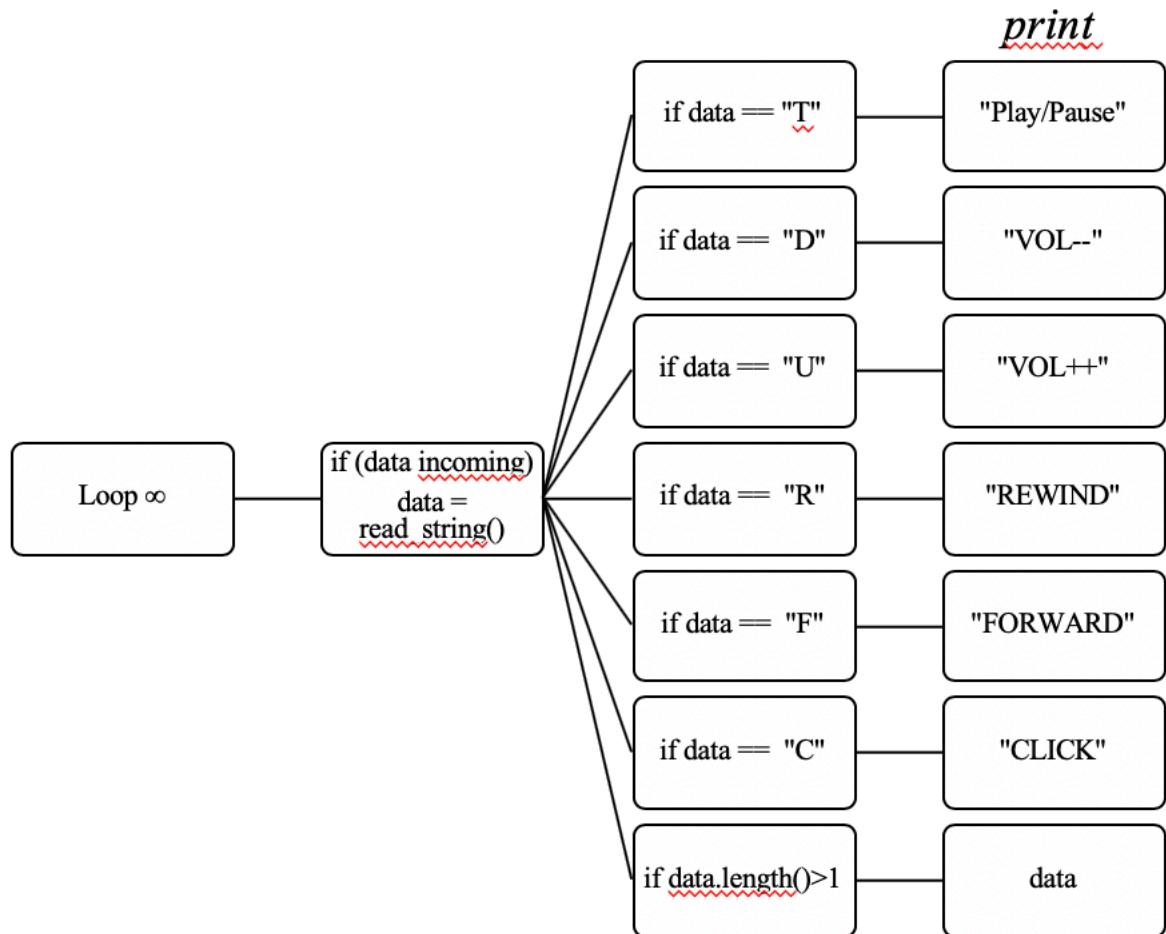


Figure II.2.1.2. Schéma de l'algorithme Arduino.

Comment accéder aux données reçues par la carte Arduino ? Comme le montre la **Figure II.2.1.2**, on vérifie si le buffer série de la carte Arduino contient des données à l'aide de la fonction *available*. Si c'est le cas on lit la chaîne de caractères avec *read_string*. Si l'on a appuyé sur un bouton, cette chaîne n'est constituée que d'un seul caractère.

Mais si sa longueur est supérieure à un, c'est qu'il s'agit des coordonnées du point où l'on souhaite déplacer la souris. Comment sont envoyées ces coordonnées ? A l'aide d'un pavé sur notre télécommande Bluetooth. Le point d'origine du pavé est situé à son extrémité supérieure gauche. Lorsqu'on appuie sur un point du Pad, Bluetooth Electronics envoie une chaîne de caractère de la forme “MX_Y_” où :

- “M” indique que le pavé a été touché une fois.
- “X_” renseigne l'abscisse du point
- “Y_” renseigne l'ordonnée du point
 (“_” étant un nombre)

Enfin, cette chaîne est directement envoyée au script Python qui sait la traiter.

II.2.2. Le programme Python

Pour les boutons ci-contre , une chaîne de caractères est donc reçue par le script qui fait appel à la fonction *press* sur la touche clavier correspondante. Par exemple si l'on a appuyé sur le bouton , notre script reçoit la chaîne “VOL++” et fait l'appel *pyautogui.press("up")*. La touche directionnelle “haut” permet en effet d'augmenter le volume.

Pour le bouton , la chaîne de caractères reçue est “CLICK” et la fonction python appelée en conséquence est *click*.

Enfin lorsqu'on a cliqué sur le pavé de la télécommande Bluetooth. La chaîne reçue est de la forme “MX_Y_”. Or la procédure python utilisée pour déplacer la souris en un point (X, Y) prends seulement ses coordonnées en paramètres. Pour ce faire, nous avons utilisé la fonction *split*.

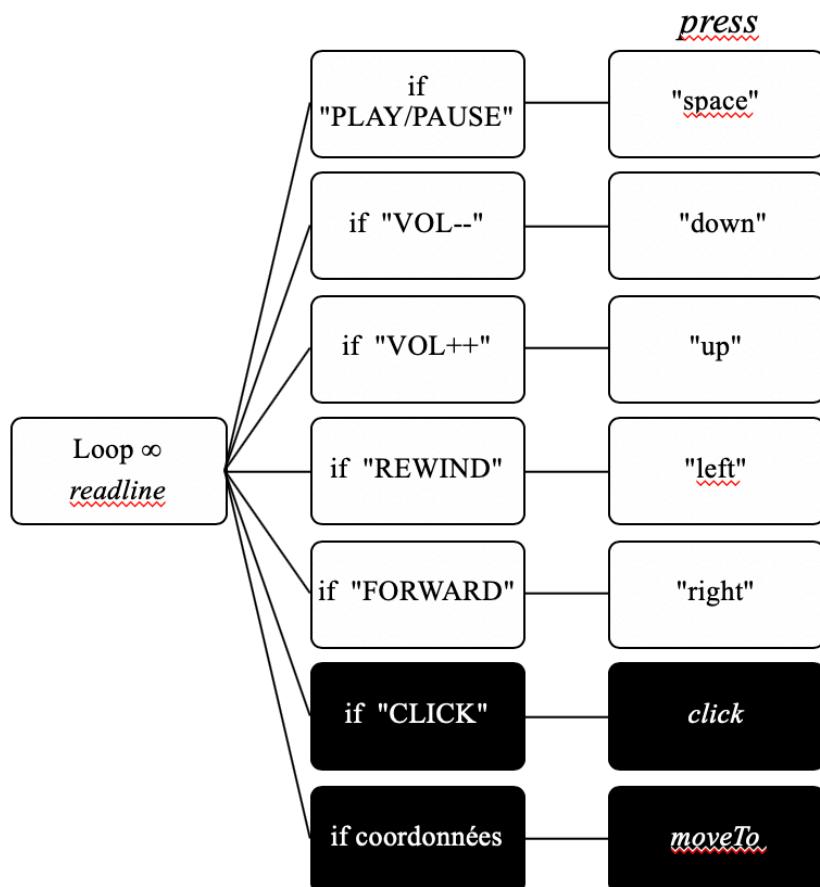


Figure II.2.2.2. Schéma de l'algorithme Python.

Chapitre III : Plannings

III.1. Planning initial

Nous avons disposé de huit séances de TP en plus du travail personnel pour mener à bien le projet. Nous avons planifié de s'occuper de la partie gestuelle pendant les quatre premières semaines et de la partie Bluetooth durant les semaines 6 et 7. L'objectif étant de vérifier le bon fonctionnement de l'ensemble du projet et de corriger les derniers problèmes. Enfin, la dernière séance est dédiée à la préparation de l'oral de fin de projet.

Semaine	Oualid Ben Mohamed	Léo Marache
Première semaine	Création Site GitHub Création du planning Montage du projet Installation python et librairies	Création Cahier de Cahier de charges Recherche de bibliographie Montage du projet Installation python et librairies Création schéma récapitulatif
Deuxième semaine	Compréhension code python Amélioration code python	Compréhension code Arduino (partie Avancer/Reculer vidéo) Amélioration de ce code
Troisième semaine	Compréhension code Arduino (partie Augmenter/Diminuer son de la vidéo) (Amélioration de ce code)	Reprise et réécriture du code Arduino
Quatrième semaine	Relecture du cours sur le Bluetooth Correction de petites erreurs	Relecture du cours sur le Bluetooth Correction de petites erreurs
Cinquième semaine	Réalisation d'un premier bouton	Écriture de l'algorithme de la partie Bluetooth
Sixième semaine	Implémentation de la partie Bluetooth sur Arduino	Implémentation de la partie bluetooth sur Arduino
Septième semaine	Finalisation de la télécommande + vérification de l'ensemble du projet	Finalisation de la télécommande + vérification de l'ensemble du projet
Huitième semaine Fin du projet	Correction de petites erreurs + préparation oral Possible améliorations ? + préparation oral	Correction de petites erreurs + préparation oral Possible améliorations ? + préparation oral

Figure III.1. Planning initial.

III.2. Planning final

Comme prévu, la première partie nous a pris quatre semaines. La seconde en revanche s'est finalement étalée jusqu'au début de la dernière séance.

Nous avons consacré la première semaine aux prérequis du projet :

- Élaboration du cahier des charges :
 - Définition des objectifs
 - Création du planning
 - Début de la bibliographie
 - Définition du matériel nécessaire
- Création du GitHub

La deuxième semaine, nous avons commencé l'écriture des programmes Arduino et Python. Même si l'on a travaillé indépendamment pendant les séances, on mettait régulièrement en commun nos avancées.

La troisième semaine a été dédiée aux ajustements du code Arduino et à la préparation de l'oral de mi-parcours. La première partie du projet est alors fonctionnel mais on teste différentes versions.

La quatrième semaine, nous avons finalisé la première partie du projet et préparé l'oral intermédiaire.

La cinquième semaine, nous avons réellement débuté la partie Bluetooth avec la création d'un premier bouton sur Bluetooth Electronics et la rédaction du code Arduino et Python associés.

La sixième semaine, nous avons finalisé les boutons et tester une première version du contrôle de la souris avec le pavé de l'application Bluetooth Electronics. Cette première tentative s'est soldée par un échec.

La septième semaine, nous avons finalement réussi à contrôler la souris à partir du smartphone mais de manière ponctuelle seulement. Cela à cause notamment des délais de communication du matériel. D'autre part, la vérification du bon fonctionnement de l'ensemble du projet initialement prévue pour cette séance n'a pas pu être faite.

Enfin la dernière semaine, nous avons donc repris l'ensemble du projet et enregistré la démonstration en cas de problème le jour de l'oral. Nous avons donc dû commencer à préparer l'oral chez nous.

On a donc atteint nos objectifs dans les temps bien que nous n'ayons pas pu respecter parfaitement le planning initial.

Semaine	Oualid Ben Mohamed	Léo Marache
Première semaine	Création GitHub Création du planning Montage du projet Installation python et librairies	Création Cahier des charges : Recherche de bibliographie Montage du projet Installation python et librairies Création schéma récapitulatif du fonctionnement du projet
Deuxième semaine	Recherche et réflexion sur l'algorithme python Que doit-il faire concrètement ?	Recherche et compréhension de l'algorithme Arduino nécessaire (partie Avancer/Reculer vidéo)
Troisième semaine	Compréhension code Arduino (partie Augmenter/Diminuer le son de la vidéo) (Amélioration de ce code)	Reprise et réécriture du code Arduino tests de différents gestes
Quatrième semaine	Relecture du cours sur le Bluetooth mise en commun sur ce qu'il va falloir réaliser, les limites ... Correction de petites erreurs	Relecture du cours sur le Bluetooth mise en commun sur ce qu'il va falloir réaliser, les limites ... Correction de petites erreurs
Cinquième semaine	Réalisation d'un premier bouton	Écriture de l'algorithme arduino pour la partie Bluetooth du projet
Sixième semaine	Implémentation de la partie Bluetooth sur Arduino + recherche sur un moyen de contrôler la souris	Implémentation de la partie Bluetooth sur Arduino
Septième semaine	Finalisation de la télécommande (travail commun sur le PAD)	Finalisation de la télécommande (travail commun sur le PAD pour contrôler la souris depuis le smartphone)
Huitième semaine Fin du projet	Recherche de solutions pour régler un problème avec le PAD Préparation de l'oral	Correction finale sur le PAD et vérification du fonctionnement global du projet préparation de l'oral et de la présentation powerpoint

Figure III.2. Planning réel.

Conclusion

L'objectif de notre projet, à savoir contrôler un ordinateur à distance par des gestes ou avec un smartphone, a été atteint dans les temps. Nous avons choisi de démontrer l'intérêt du projet dans le cadre de la lecture d'un objet multimédia. Cette situation se prête parfaitement au contrôle à distance.

Durant notre travail, nous avons été confrontés à quelques difficultés. La première a été de d'établir un moyen pour le programme Python de recevoir des données d'un algorithme extérieur. Le projet en a dépendu. En ce qui concerne la partie gestuelle, nous avons tester de multiples versions du code Arduino. Finalement, nous nous sommes restreints à des gestes les plus intuitifs possibles et faciles à réaliser. Dans l'ensemble, nous avons réussi à apporter des solutions aux problèmes rencontrés à l'exception de la souris que l'on pensait pouvoir contrôler de manière continue.

Les pistes d'améliorations concernent entre autres la vitesse de communication du matériel. Cela implique l'utilisation d'une autre carte Arduino et module Bluetooth. Alors nous pourrions déplacer en temps réel la souris. D'autre part, il serait intéressant de pouvoir saisir une recherche internet depuis son smartphone et voir les résultats sur le navigateur web de l'ordinateur. Il faudrait pour cela construire un clavier dans l'application Bluetooth Electronics (ou à défaut en créer une) et appeler la procédure python permettant l'ouverture d'un navigateur web.

Nous avons apprécié travailler sur un sujet qui est au cœur de l'actualité industrielle. Au fil des années, le nombre d'objets contrôlables à distance a en effet fortement augmenté. Comme nous l'avons vu, les moyens sont variés. Ce peut être un contrôle gestuel dans le cas des téléviseurs, par Bluetooth via son smartphone ou encore par commandes vocales. C'est d'ailleurs l'une des perspectives de notre projet. Si on disposait de huit semaines supplémentaires, on travaillerait sur un moyen d'effectuer une recherche internet par commande vocale et d'afficher les résultats sur le navigateur web de l'ordinateur.

Enfin, la partie gestuelle du projet donne un aperçu des applications possibles pour les personnes présentant un handicap physique.

Bibliographie

Librairie pyautogui :

<https://pyautogui.readthedocs.io/en/latest/>

<https://pyautogui.readthedocs.io/en/latest/keyboard.html>

<https://pyautogui.readthedocs.io/en/latest/mouse.html>

Librairie serial :

<https://pyserial.readthedocs.io/en/latest/pyserial.html>

https://pythonhosted.org/pyserial/pyserial_api.html

Détection des gestes :

<http://users.polytech.unice.fr/~pmasson/Enseignement/Elements%20de%20robotique%20avec%20arduino%20-%20Distance%20et%20detection%20d%20obstacles%20-%20Projection%20-%20MASSON.pdf>

Partie Bluetooth :

<http://users.polytech.unice.fr/~pmasson/Enseignement/Elements%20de%20robotique%20avec%20arduino%20-%20Communications%20RF%20-%20Projection%20-%20MASSON.pdf>

<https://www.pythonforbeginners.com/dictionary/python-split>