Marquet Noémie
Vinoy Agnès

Nogier Augustin
Yang Yijun

Rapport du projet ProTis
Apprentissage de la programmation

2020

Marquet Noémie
Vinoy Agnès
Nogier Augustin
Yang Yijun

Rapport du projet ProTis
Apprentissage de la programmation

Table des matières

1. Description de la problématique

L'objectif de ce projet est d'initier un jeune public à la programmation à travers plusieurs jeux comme Snake ou encore un labyrinthe ou une liste de missions à remplir. Les actions programmées permettent ensuite de diriger un bras robotique selon 2 axes. La commande de ce bras se fait grâce à une carte Raspberry PI.
Pour ce projet il a été important d'avoir une interface accessible à un jeune public et donc très claire et intuitive.

La situation de confinement nous a cependant forcé à modifier certains aspects de notre projet. Il n'était dès lors plus possible d'avoir accès au matériel. Ainsi, la réalisation des circuits a dû être réalisée par simulations et les performances n'ont à aucun moment pues être testées de manière réelle, aucun d'entre nous ne possédant de carte Raspberry PI ou de matériel similaire. Cet aspect n'a pas été facile à gérer et des compromis sur la robustesse de la réalisation de notre projet ont dû être effectués au vu des outils à dispositions. Cet aspect du projet sera développé par chacun dans sa partie et dans le retour d'expérience.

2. Schéma de l'architecture

On peut résumer l'architecture de ce projet à 3 blocs :

- Une interface graphique
- Une chaîne de transmission aux moteurs
- Un asservissement du déplacement des bras

Sur le schéma précédent, on peut voir l'architecture global de notre projet ainsi que les interactions entre les différents blocs. Pour être plus efficace, nous nous sommes répartit les différentes tâches.
Noémie et Agnès se sont chargées de l'interface graphique. Tandis que Augustin a simulé et écrit le code importé sur la Rasberry PI. Enfin, Yijun a simulé les moteurs et la plateforme de translation.

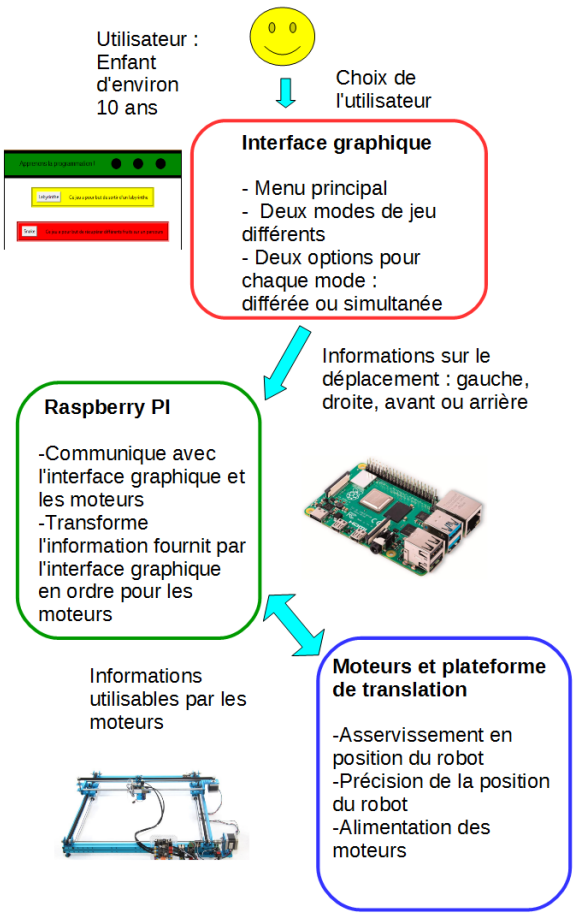


Figure 1 : Représentation de l'architecture de notre projet

3. Structure de l'interface graphique et description ergonomique

Le premier choix à faire pour la réalisation de cette interface a été celui du support à utiliser. La question se posait principalement entre utiliser Matlab et son module GUIDE ou python et son module tkinter.

Par soucis de modularité entre notre interface et la commande des moteurs, nous avons collectivement décidés d'utiliser python et donc tkinter.

La prochaine étape a été de choisir quelles activités implémenter qui soient à la fois ludiques et liées à la découverte de la programmation et complémentaires au déplacement d'un bras robotique. Nous avons choisi de créer 3 jeux même si, au final, seuls 2 ont été réalisés : un Snake, un labyrinthe et un jeu de mission (celui-ci, par manque de temps, a été abandonné).

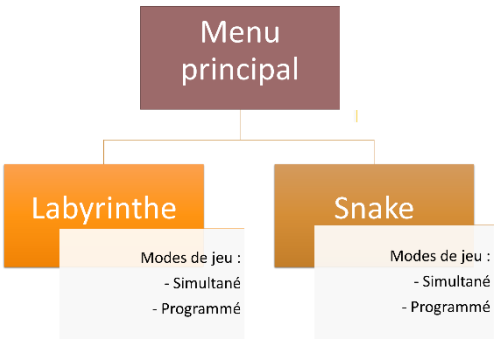


Figure 2 : Schéma de l'arborescence de l'interface graphique

Pour structurer notre interface et la rendre accessible à un jeune publique, nous avons opter pour l'implémentation d'un menu principal qui par la suite donnerait accès aux jeux. On peut présenter l'architecture de l'interface graphique avec le schéma de la figure 2.

I) Le menu principal

Le menu est assez simple : il est constitué d'un bandeau « titre » et de 2 bandeaux « jeu ». Chaque bandeau « jeu » est constitué d'un bouton créant une nouvelle fenêtre et d'une courte description du jeu proposé.

L'aspect de cette interface est visible dans la figure 3.

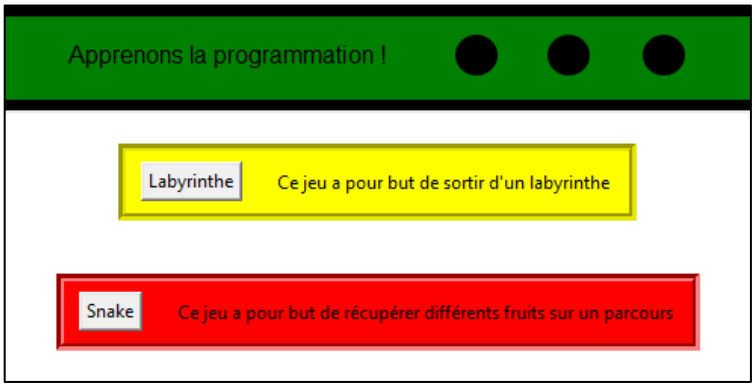


Figure 3 : Interface du menu principal

A la base, nous avions prévu d'insérer un bandeau sous forme d'image pour que l'apparence soit plus attrayante et travaillée. Cependant, nous avons rencontré un problème de compatibilité entre nos différentes versions de python, certaines gérant mal l'importation d'image. Pour faciliter le travail collectif, nous avons donc choisi de rester sur une apparence plus simple.

La fenêtre du menu constitue la fenêtre principale pour Python et lorsque l'on appui sur les boutons des fenêtres « Top-Level » sont ouvertes et passant la fenêtre principale en arrière-plan.

Un problème que nous n'avons toujours pas réussi à régler, est la gestion des événements dans les fenêtres « Top-Level ». Le code qui est donc attaché à ce document dans la partie 7, est donc constitué de 3 codes différents : un pour le menu principal qui permet d'ouvrir les fenêtres mais pas de jouer, un pour le Snake et un pour le labyrinthe.

II) Mode de jeu "Snake"

Le premier jeu proposé au joueur est un jeu inspiré du jeu bien connu Snake où un serpent chercher à collecter le plus de fruit possible. Nous avons réalisé un mode de jeu semblable pour notre module d'apprentissage de la programmation.

Dans ce mode, un carré bleu "le serpent" ou le robot se déplace jusqu'à une "pomme" (le rond rouge) à l'aide des flèches du clavier. Lorsqu'il arrive sur la pomme il peut la manger avec la touche Echap. Cependant, le joueur ne peut pas manger la pomme s'il n'est pas à côté. Une fois la pomme mangée, elle est replacée à un autre endroit aléatoirement. Le score est affiché en haut de la fenêtre et les consigne en bas.

Ce mode de jeu dispose d'une version simultanée et d'une version différée. Dans le mode simultané, le robot ou le serpent se déplace au fur et à mesure que les ordres du joueur sont donnés. Dans le mode différé, les ordres sont enregistrés et ne sont exécutés que si le joueur presse la barre d'espace. Pour passer d'un mode de jeu à l'autre, il faut changer une variable dans le code. Le mode différé permet à l'enfant d'anticiper les mouvements du robot et donc de mieux appréhender la programmation.

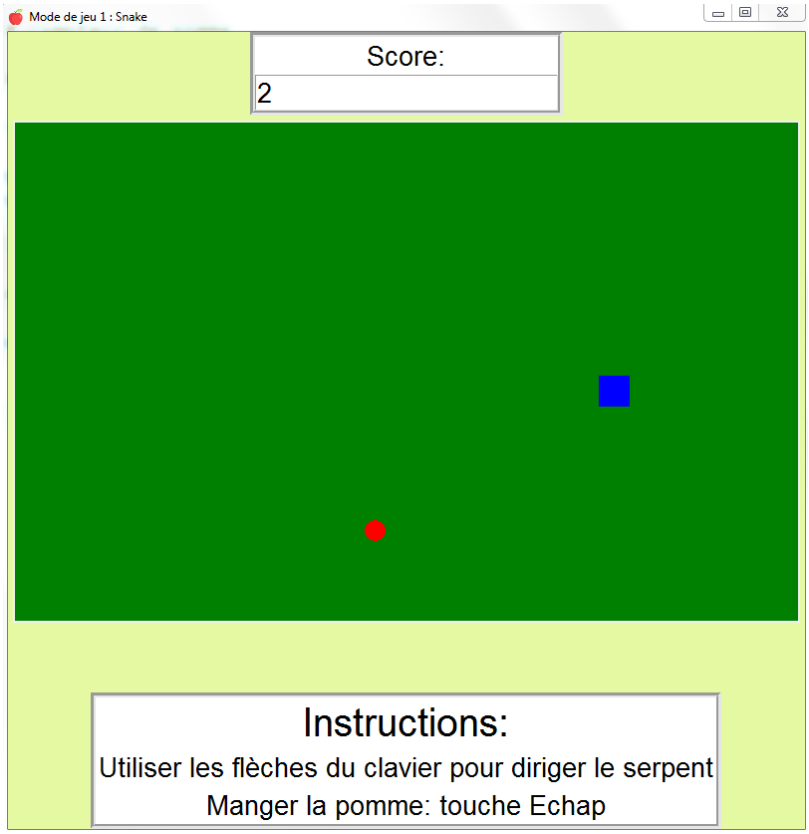


Figure 4 : Plateau du jeu Snake

Il manque encore de nombreuse fonctionnalité au jeu qui n'ont pu être mises en place par manque de temps :

- L'allongement du serpent à mesure que le score augmente
- Fin du jeu si le serpent se mord la queue ou rentre dans le bord de la fenêtre
- Bouton permettant de choisir le mode simultané ou le mode différé.

Dans l'hypothèse où nous aurions pu avoir accès au matériel électronique et à la plateforme de translation. Un petit objet représentant la pomme aurait pu être disposer sur le plateau et le joueur aurait pu déplacer le robot jusqu'à cet objet.

III) Le labyrinthe

Le second jeu actuellement proposé est un labyrinthe.
Le but du jeu est assez évident : le pion représentant le joueur (rouge) doit atteindre le drapeau (rectangle vert) comme on peut le voir sur la figure 5.

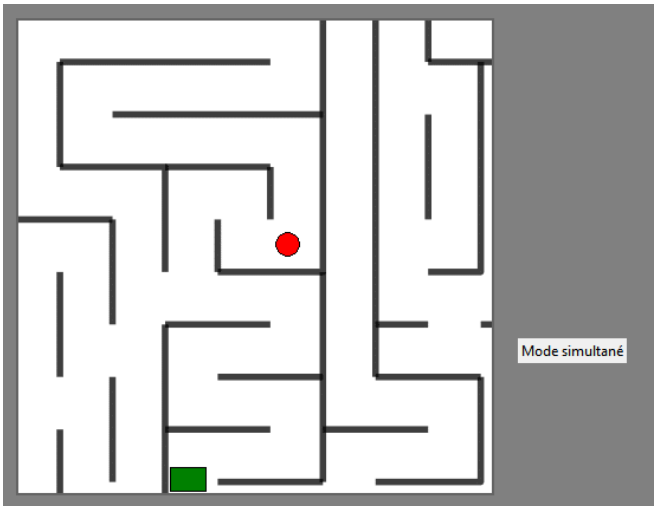


Figure 5 : Plateau du labyrinthe en mode simultané

La position de l'arrivée, représentée par le rectangle vert, peut être modifiée dans le code en changeant les valeurs des variables *x_a* et *y_a* aux lignes 116 et 117. De même, on peut changer la position initiale du pion aux lignes 148 et 149 en modifiant les valeurs de *xpp* et *ypp*.

Ce jeu est peut être joué de 2 façons différentes : en mode « simultané » où le pion se déplace à chaque action de l'utilisateur et en mode « différé » où on enregistre les commandes de l'utilisateur jusqu'à ce que celui-ci appui sur un bouton pour effectuer les actions listées.

Le choix de ces modes de jeu ne peut pas encore se faire directement depuis l'interface en changeant la valeur de la variable v à la ligne 12 : 0 pour le mode « simultané » et 1 pour le mode « différé ». Une amélioration évidente serait que cette variable puisse être changée depuis le jeu en lui-même. Une étiquette à la droite du plateau permet de savoir dans quel mode le joueur se trouve.

```
(86, 10)
(105, 10)
(124, 10)
(143, 10)
(162, 10)
(181, 10)
(200, 10)
(219, 10)
(219, 29)
(219, 48)
(219, 67)
(219, 86)
(219, 105)
(219, 86)
```
Figure 6 : Affichage des coordonnées dans Python

Pour le mode « différé » particulièrement, nous voulions afficher la liste des actions effectuées par l'utilisateur avant que celles-ci soient exécutées. Cela lui permettrait de se souvenir de ce qu'il aurait « programmé »et ainsi mieux assimiler la notion de programmation. A défaut d'avoir réussi à implémenter cet affichage, la liste des coordonnées successives s'affiche dans l'affichage de python (voir figure 6)

Pour mieux mesurer chaque déplacement, une échelle est affichée à la droite du plateau comme sur la figure 7.

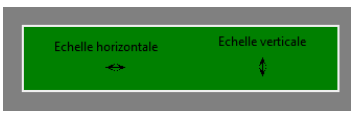


Figure 7 : Echelle de déplacement du joueur dans la labyrinthe

Le programme permet également de changer relativement facilement de plateau de jeu. Il suffit de changer le nom du fichier à ouvrir à la ligne 110 en faisant attention à respecter les bonnes

dimensions. Là encore, nous aurions préféré que cette modification puisse se faire depuis l'interface mais ce n'est pas encore le cas.

4. Liaison entre l'interface graphique et les moteurs

Le rôle de la plateforme de translation est de performer les mêmes mouvements commandés via l'interface graphique que ceux effectués dans les différents modes de jeu. Pour cela, nous utilisons deux moteurs qui permettent au robot de se déplacer librement dans un plan horizontal.

Les modes de jeu consistant en une série de déplacements sur une grille, nous avons choisi d'utiliser deux moteurs pas à pas ; ainsi la précision des déplacements sur la grille est garantie par le nombre de pas effectués par les moteurs.

Parmi les deux types de fonctionnement qu'offrent les moteurs pas à pas, nous avons choisi d'utiliser la configuration en *Full-step,* où une rotation complète du barreau correspond à quatre pas. Comparée à huit pour le fonctionnement en *Half-step,* la précision sur la position finale du robot n'est toutefois pas modifiée, dans la mesure ou une case de la grille correspond à un nombre entier de rotations complètes du barreau, ce qu'on suppose ici.

Cependant, en raison des impossibilités techniques liées au confinement, la construction de la grille est rendue impossible ; le programme effectuant la liaison interface-moteurs associe donc, par convention, un déplacement d'une case à 64 pas en *Full-step* de la part des moteurs (défini avec *StepNumber=64* à la ligne 20).

Le programme en question est donc rédigé sous Python et travaille sur la carte Raspberry Pi, à laquelle l'on branche – l'on aimerait brancher, s'entend – les deux moteurs pas à pas via huit sorties GPIO, que l'on repère par leur *pin* dans le programme (avec *StepPins* à la ligne 32).

Le raccord avec l'interface graphique est réalisé grâce à la transmission d'une liste de déplacements du programme gérant l'interface vers le programme contrôlant les moteurs. Celle-ci prend la forme d'un *Array* de nombres entre 1 et 4, correspondant chacun à un type de déplacement (1=avant, 2=arrière, 3=droite, 4=gauche), et généré à l'issue de la série de déplacements. Le second programme prend alors ces informations pour faire effectuer les mouvements correspondants à la platine de translation.
Note : dans le programme de contrôle des moteurs, on crée une telle liste Liste_deplacements *à la ligne 49, servant au test du programme sans liaison avec l'interface mais obsolète dans le cas où celle-ci est établie.*

Toutefois, ce type de liaison ne permet pour l'instant que d'effectuer les mouvements en fin de séquence, et n'est donc adapté que pour le mode de jeu en différé. Une piste d'amélioration est d'ajuster la façon dont l'information des mouvements est transmise entre ces deux programmes, afin de permettre le déplacement de la platine de translation en temps réel pour le mode simultané.

Le test du programme contrôlant les moteurs, ainsi que le raccord avec l'interface graphique, n'ont toutefois pas pu être testés expérimentalement, en raison du manque d'accès au matériel. Si la compilation prouve que la modularité est assurée d'un point de vue syntaxique, le test du contrôle des moteurs par les deux programmes mis ensemble doit cependant être effectué par d'autres moyens.

5. Résultats des simulations des moteurs avec analyse

Parce que nous n'avons pas ces éléments pour construire un prototype exact, un simulateur logiciel pratique devrait être appliqué pour les simulations de moteur et de circuit. Il y a des contraintes pour ce sujet : l'une est la vitesse du mouvement, l'autre est la précision.

Nous avons donc considéré deux types de moteur : le moteur à courant continu et le moteur pas à pas. Notre idée est que les déplacements du platine se feront de case en case, selon un damier imposé.

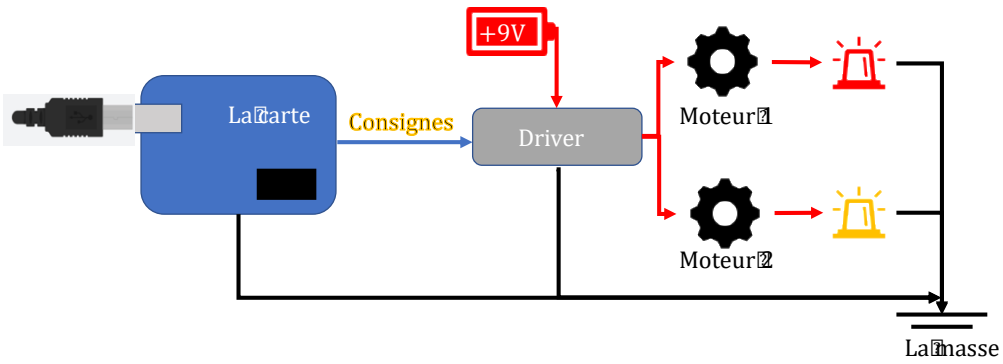
La vitesse du moteur courant continue est contrôlée par une impulsion PWM et généralement assez rapide. Mais il est difficile de contrôler sa position car la durée du courant doit être prise en compte précisément.

Le moteur pas à pas est un meilleur choix au contraire car il est naturellement précis dans le mouvement d'angle. De plus, la carte Raspberry Pi est un bon choix pour piloter le moteur car elle utilise python comme langage. Par conséquent, une combinaison de Raspberry Pi et moteur pas à pas est notre premier choix.

Cependant, comme nous ne trouvons que le simulateur pour Arduino et langage C, on a quand même fait un modèle avec eux avec ces circuits pour bien montrer notre idée. Donc, nous avons à la fois un modèle en Raspberry Pi avec python et Arduino avec C.

Voici le schéma de notre circuit. Pour la simulation, nous avons utilisé la carte Arduino et le pont H comme lecteur. Le moteur choisi ici est le moteur à courant continu. Deux moteurs peuvent effectuer le mouvement séparément selon la consigne itinéraire généré par le bloc précédent, l'un fait en avant et en arrière, l'autre fait à gauche et à droite. Il y a 2 LED après les moteurs surveillent ses états.

Figure 8 : le schéma de notre circuit en bloc



L'image entière est montrée ci-dessous. Le logiciel correspondant va :

a) Recevoir le consigne (sens et nombre d'étapes)
b) Piloter le bon moteur dans la bonne direction
c) Contrôler l'heure actuelle afin que le moteur s'arrête au bon endroit

Ici, l'heure actuelle d'une étape est inconnue. Nous avions prévu d'utiliser Matlab Simulink pour modéliser l'ensemble du moteur afin de trouver la bonne durée, mais nous n'avons pas le temps pour cela.
Il convient de noter que si nous pouvons utiliser le moteur pas à pas en réalité, nous ne pouvons considérer que le "pas d'angle" plutôt que le "pas de temps".

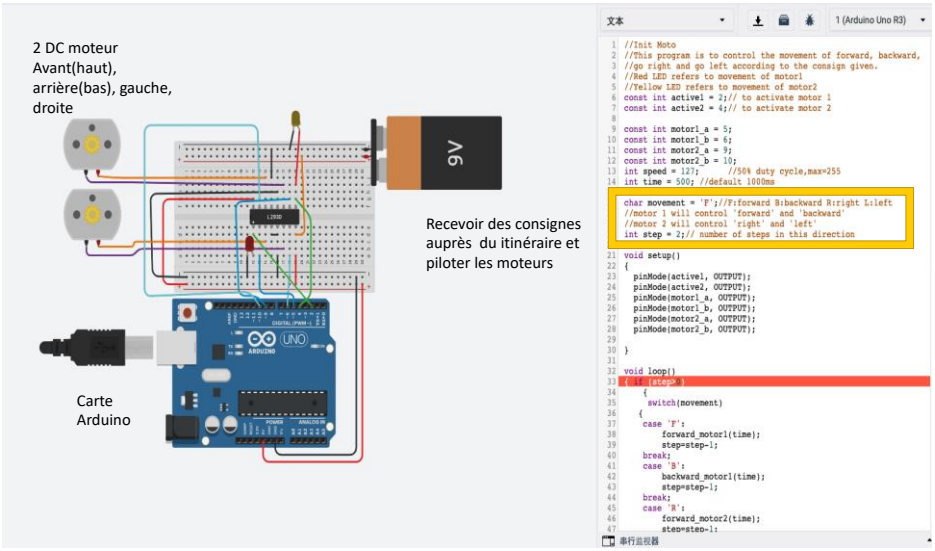


Figure 9 : Notre circuit en simulation et son code correspondant (Arduino+C)

6. Retour d'expérience du travail en équipe dans des conditions particulières

La menée du projet à distance n'a pas été évidente. En effet à chaque problème rencontré, il était très compliqué de communiquer avec les autres membres de l'équipe pour demander conseil. Chaque question nécessitait une description détaillée à l'oral ou à l'écrit vu que la connexion internet ne pouvait pas forcément supporter les partages d'écran.

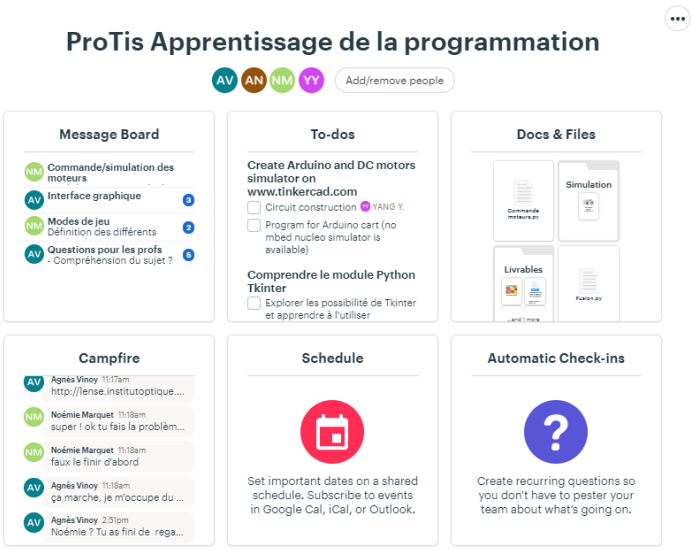


Figure 10 : Page principal de notre basecamp

Pour faciliter nos échanges, nous avons rapidement utilisé l'outil basecamp mis à notre disposition (voir figure 10). Nous nous sommes servis en particulier du « Message board » pour garder nos observations et nos décisions générales en un seul endroit, des « To-dos » pour nous permettre de voir notre progression. Nous l'utilisions conjointement à un tableau excel que nous remplissions à chaque séance pour garder une trace de ce qui avait était fait, par qui, et quand. Ce document était stocké sur un google drive qui a été associé au base camp et qui nous permettait de stocker les documents qui étaient modifiables en simultané. Les autres fichiers ont été rangé dans les différents dossiers du basecamp (voir figure 11). Enfin, nous avons également utilisé le chat de basecamp pour ne pas perdre nos discussions ce qui était le cas lors de la fermeture de la session sur ecampus. Nous avons aussi utilisé une conversation Whatsapp pour pouvoir se joindre rapidement les uns les autres en cas de problème. En effet, nous ne regardions pas toujours le chat du projet Basecamp en dehors des heures de cours.

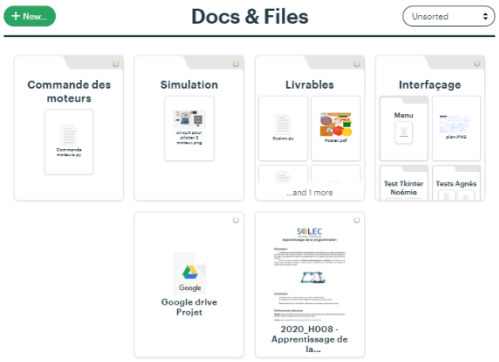


Figure 11 : Organisation de nos documents sur basecamp

Au-delà de notre groupe, nous avons également échangé avec d'autres grâce au forum de discussion sur ecampus. Nous avons ainsi pu échanger des liens de MOOCs intéressants ou encore poster des retours d'expérience et de difficultés rencontrées.

7. Bilan des acquis et des cours suivis avec une biblio argumentée

I) Agnès

Pour apprendre à me servir de tkinter, j'ai suivi plusieurs cours en ligne donc plusieurs vidéos sur YouTube qui permettait de commencer un projet rapidement et ainsi d'assimiler l'utilisation des fonctions les plus importantes immédiatement. Les liens des 2 vidéos sont données ci-dessous.

- Créé Un jeu de Dames en python PARTIE 1 par BacrTutoriel : https://www.youtube.com/watch?v=bzSpU9s2Pto
- Créé Un jeu de Dames en python PARTIE 2 par BacrTutoriel : https://www.youtube.com/watch?v=I7BsPG38gos

Je me suis par la suite surtout servi de 3 sites qui recensait les différentes fonctions associées à la création d'une interface graphique en proposant des exemples et les diverses options de chaque fonctions.

- http://www.xavierdupre.fr/app/teachpyx/helpsphinx/c_gui/tkinter.html#canevas
- https://python.doctor/page-tkinter-interface-graphique-python-tutoriel
- http://tkinter.fdex.eu/doc/caw.html#can-images

J'ai trouvé que les 3 sites étaient assez complémentaires, l'un étant plus complet pour les images, par exemple et l'autre pour les canevas.

Enfin j'ai participé quelques fois au forum de discussions sur la prise en main de tkinter lorsque je trouvais un lien intéressant à partager ou que je rencontrais une difficulté auquel d'autres auraient également pu se heurter tant pour avoir des éléments de réponse que pour proposer un retour d'expérience.

II) Noémie

J'ai moi aussi appris à me servir du module Python Tkinter car j'ai travaillé avec Agnès sur la partie interface graphique du projet.

Tout d'abord, j'ai visionné la vidéo Apprendre le Python : Tkinter de Graven :

https://www.youtube.com/watch?v=N4M4W7JPOL4

Cette vidéo aborde les bases de Tkinter tel que la création d'une fenêtre, les canvas et les boutons. Elle m'a permis de commencer à prendre en main Tkinter. A ce moment, je me suis rendu compte que je n'arrivais pas à afficher les images dans les fenêtres Tkinter. Après plusieurs essais infructueux, je suis arrivé à la conclusion que ma version de Python ou ma version de l'éditeur Pyzo avait un problème. J'ai renoncé à afficher les images dans Tkinter.

Je me suis ensuite demandé si le module de Python Pygame n'était pas plus adapté à notre usage. Pour en avoir le cœur net, j'ai visionné la vidéo du même youtubeur sur le module Pygame :

https://www.youtube.com/watch?v=8J8wWxbAdFg&t=794s

J'en ai conclu que bien que beaucoup plus esthétique que Tkinter, Pygame n'était pas adapté à notre projet d'électronique. En effet, le prise en main est plus compliqué et la communication avec la carte Raspberry Pi aurait été plus compliquée.

Dans la suite de mon apprentissage de Tkinter, j'ai pioché les informations dont j'avais besoin au faire et à mesure que le projet avancé. Ainsi, le site Zeste de savoir m'a permis d'aborder la gestion des événements dans Tkinter :

https://zestedesavoir.com/tutoriels/1729/programmation-avec-tkinter/programmation-evenementielle-avec-tkinter/

Le site suivant m'a fourni les noms des touches du clavier dans Tkinter.

http://tkinter.fdex.eu/doc/event.html

J'ai beaucoup aimé apprendre à me servir de Tkinter même si de nombreuses subtilités de ce module m'échappent encore.

III) YANG Yijun

Je regarde Matlab Simulink pour faire la modélisation d'un moteur à courant continu (dans le 'help' de Matlab). J'étudié l'exemple dans Simscape et vu les résultats de simulation (front montant et descendant de vitesse).

J'ai trouvé un simulateur en ligne pour la carte Arduino avec langage C pour accomplir. Les cours peuvent se trouver dans le même site.

- www.tinkercad.com

IV) Augustin

Si le principe de fonctionnement des moteurs pas à pas m'était déjà familier, j'ai en revanche appris à les commander via Raspberry Pi, et ai donc dû apprendre la syntaxe adéquate en Python. Le site que j'ai pu trouver donne un exemple de programme commandant un moteur pas à pas unipolaire à 4 fils en *Full-step* et *Half-step,* en utilisant la syntaxe et les modules Python appropriés.

Il explique également les broches de la Raspberry Pi, ce qui m'a été utile pour déterminer les branchements du second moteur.

https://www.aranacorp.com/fr/pilotez-un-moteur-pas-a-pas-avec-raspberrypi/

8. Les programmes commenté et les schémas électriques

Les différents programmes Python que nous avons réalisés pendant ce projet sont disponibles dans un dossier compressé joint à ce rapport technique.

9. Les problèmes à régler

Lors de ce rapport, nous avons rencontré de nombreux problèmes techniques que nous n'avons malheureusement pas pu tous résoudre.

Interface graphique :

Pendant la création de l'interface avec l'utilisateur, nous avons rencontré plusieurs problèmes liés à l'utilisation de Tkinter.

- Nous nous sommes aperçus que certains codes qui fonctionnaient sur un ordinateur ne fonctionnaient pas sur un autre. Par exemple, l'affichage des images à poser des problèmes.
- Un autre problème rencontré a été la mise en place d'un menu principal. En effet, la création d'une fenêtre secondaire est possible sous Tkinter mais les différents jeux (Snake et Labyrinthe) ne fonctionnent pas dans cette nouvelle fenêtre. Nous supposons que ce problème est lié à la gestion des variables.

Simulation des moteurs :

Le simulateur en ligne pour la carte Arduino est pour les débutants plutôt qu'un grand projet. Il manque pas mal de composent. L'utilisation de Matlab Simscape est obligatoire si on veut dérouler notre projet. Malheureusement, ce logiciel est tellement complexe en sort que nous avons besoins de beaucoup de temps de l'apprendre.

L'absence de moteurs nous ayant forcés à simuler leurs actions en utilisant d'autres conventions, nous n'avons pas pu tester entièrement la modularité de nos programmes entre eux et n'avons donc pas la garantie d'une chaîne entièrement fonctionnelle entre l'interface et la platine de translation.

10. Améliorations possibles

Au niveau de l'interface, plusieurs améliorations sont envisageables et même nécessaires. Il faudrait par exemple que le choix du mode soit accessible dans l'interface et que l'on puisse pouvoir choisir le plateau sur lequel on veut jouer en chargeant un fichier de son choix ou en choisissant un dans une liste. Il faudrait aussi que l'affichage des actions se fasse directement dans l'interface pour que le joueur y ait facilement accès.

Une autre amélioration serait de placer des « obstacles » sur les murs du labyrinthe pour empêcher certaines actions.

On pourrait également implémenter d'autres structures de programmation plus complexes comme des tests « if » ou des boucles « while » ou « for ». Cela pourrait constituer un mode de jeu avancé.

La liaison entre l'interface graphique et la commande des moteurs n'est pour l'instant adaptée que pour le fonctionnement en différé. Il faut donc travailler sur la façon de mettre en place cette transmission d'information afin qu'elle ait lieu en temps réel, permettant ainsi au mode simultané de fonctionner avec les moteurs.

Enfin, il pourrait être intéressant de faire fonctionner les codes de l'interface graphique et de la carte Rasberry PI ensemble. Assemblage que nous n'avons pas eu le temps de faire.

Conclusion

Ce projet d'électronique a été fortement modifié et altéré par la crise sanitaire en cours. Nous regrettons de ne pas avoir eu accès au matériel d'électronique nécessaire pour tester nos différents programmes. Malgré tout, ce projet a été une expérience intéressante tant du point de vue technique et informatique que du point de vue de l'organisation et du travail d'équipe. Ainsi, travailler en équipe à distance a été un vrai défi. Nous avons appris que cela demande une bonne communication et une bonne organisation. De plus, le partage des informations devient critique dans ce contexte. Nous avons enfin appris à nous former en autonomie à l'aide des outils disponible en ligne.