

Rapport de projet

Arduino

Connect Four



Sommaire

I. Cahier des charges

II. Schéma électrique

III. Algorithme de fonctionnement et explications brèves du code

IV. Coût du projet :

A. Matériel

B. Ingénieur

V. Plannings : prévisionnel et réel

VI. Les problèmes rencontrés

VII. Conclusion

A. Rendu final

B. Apports du projet

C. Perspectives d'améliorations

Bibliographie

Ressources pour le code :

[https://github.com/gcc-mirror/gcc/blob/master/libstdc%2B%2B-v3/
libsupc%2B%2B/initializer_list](https://github.com/gcc-mirror/gcc/blob/master/libstdc%2B%2B-v3/libsupc%2B%2B/initializer_list)

I. Cahier des charges

Cahier des charges :

* **Contexte :** Dans le cadre du projet Arduino il fallait choisir un sujet qui utilise les fonctionnalités d'Arduino, tout en restant réalisable en 8 séances de 3h chacune. Nous avons choisi un puissance 4 électronique qui se joue contre une intelligence artificielle.

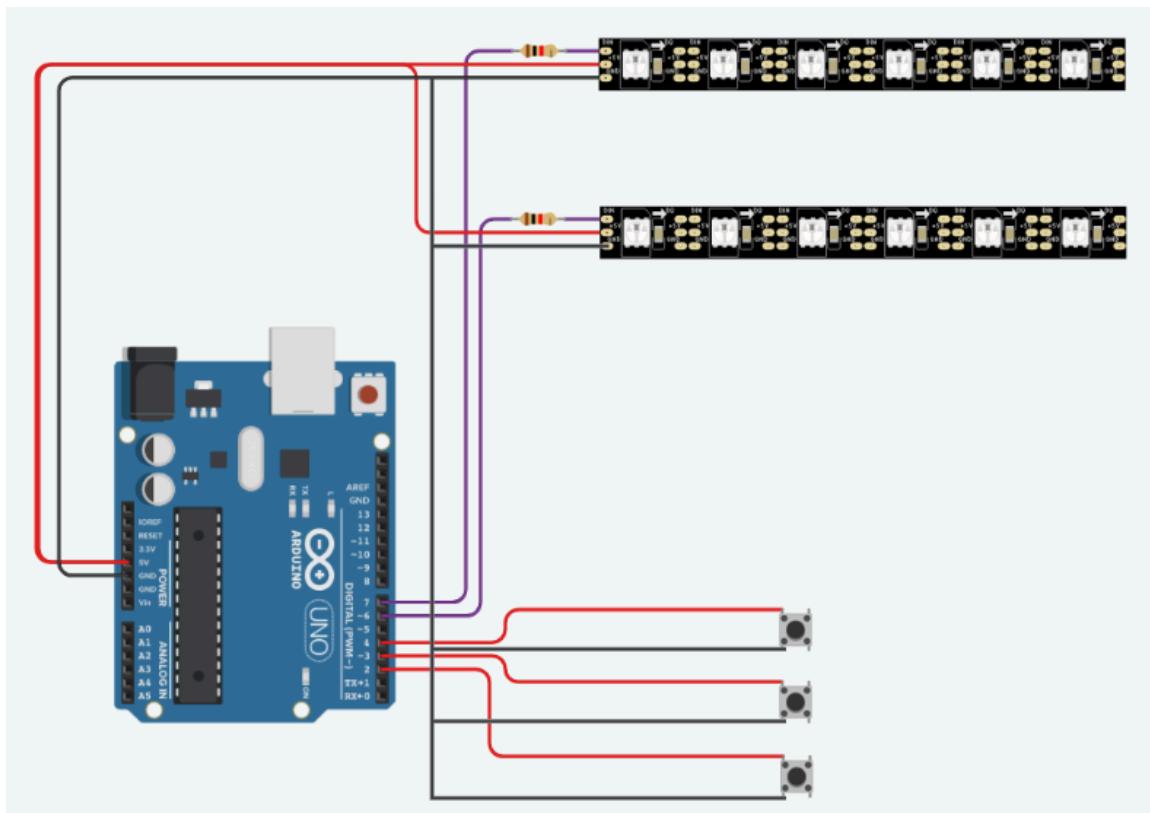
* **Objectifs :** - réaliser quelque chose de ludique

- moderniser un jeu classique
- jouer une partie contre une intelligence artificielle
- veiller à ce que la manière de jouer reste intuitive

* **Fonctions :** - puissance 4 électronique

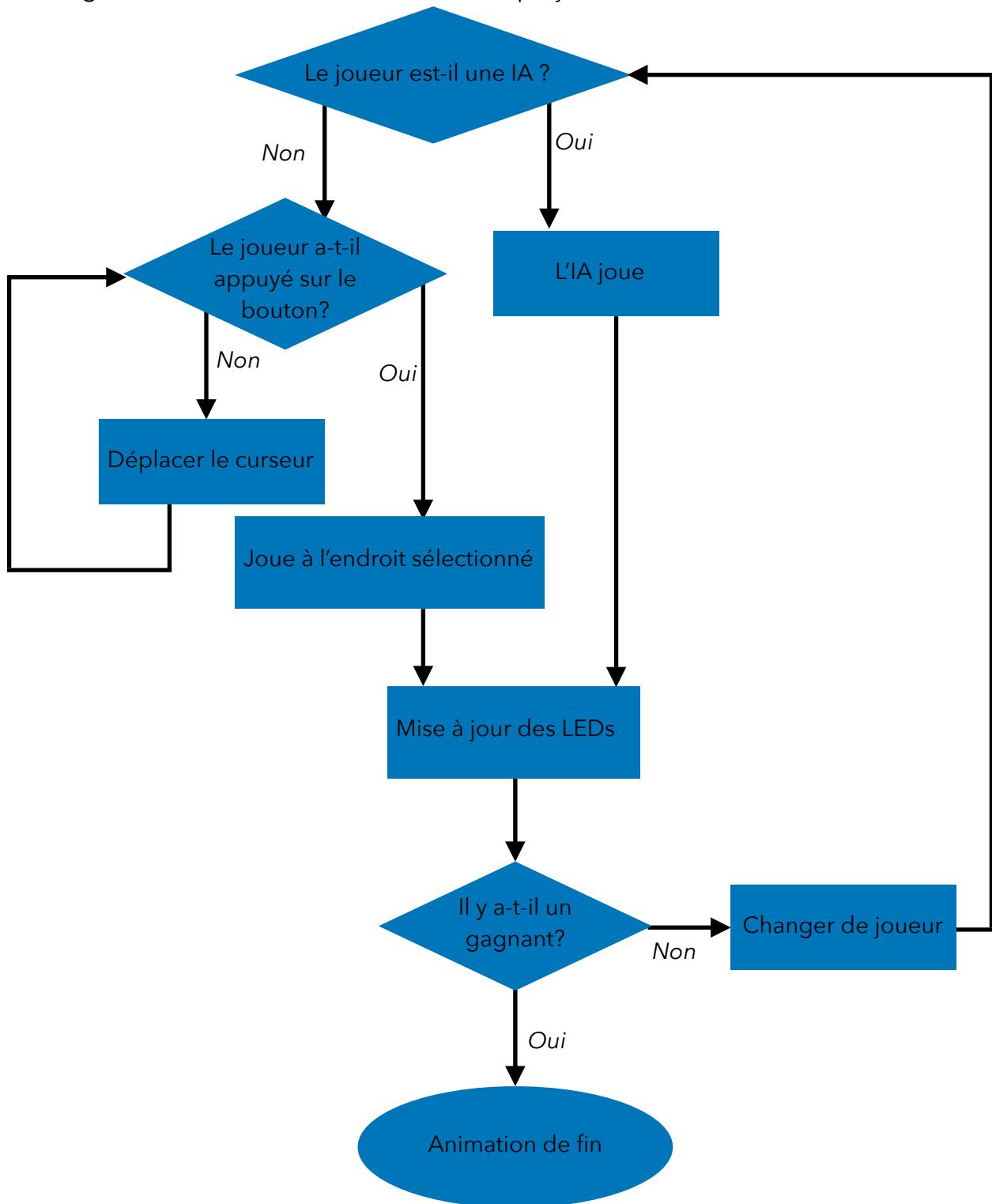
- partie qui se joue contre un ordinateur
- commandes : 3 boutons → 1 qui permet de décaler la colonne sélectionnée d'un rang vers la gauche, 1 autre d'un rang vers la droite et le dernier de valider la position

II. Schéma électrique



III. Algorithme de fonctionnement et explications brèves du code

Voici l'algorithme de fonctionnement de notre projet.



Quelques mots à propos du code de notre projet :

- La structure du code est orientée objet.

- Le code utilise beaucoup de listes, coordonnées et matrices.
- **Classe Array** : inspirée de la classe array de la librairie standard de C++, contenant énormément de choses très utiles mais qui n'est pas dans la librairie Arduino parce qu'elle n'est pas faite pour des appareils avec du matériel informatique si minimal.
- **Classe Initializer_list** : code directement extrait de la librairie standard mais avec des modifications. Permet l'initialisation d'array avec des accolades.
- **Classe Board** : la classe du jeu en lui-même . Dans un premier temps nous avons fait une version qui compile sur architecture 64 bits le temps que l'on construise la structure et une fois que nous avons eu le début de l'écran nous avons adapté le code pour Arduino. Ce qui veut dire un fichier *.ino* qui utilise la classe board qui elle-même utilise array qui utilise *initializer_list*.

Pour faire tourner le jeu on a besoin d'une boucle "infinie" pour que le jeu fonctionne constamment en attendant un input et c'est exactement ce que fait la méthode loop d'Arduino donc nous nous en sommes servi pour gérer l'avancement du jeu. L'idée c'est que l'on attend un input et dès qu'on valide on joue à cet endroit, on affiche l'état du jeu sur les LEDs puis on vérifie s'il y a un gagnant, auquel cas une animation se lance. En ce qui concerne la classe board, la grille de jeu est une matrice 6 lignes 7 colonnes remplie par des valeurs d'une énumération player, soit le joueur 1, soit le joueur 2, soit le joueur "0" quand c'est vide. À chaque fois que l'on joue dans une colonne on place un pion à l'emplacement le plus bas où il n'y a pas déjà un joueur. Et les 2 choses les plus compliquées à mettre en place ont été la vérification du gagnant et l'IA.

Pour la vérification, à chaque fois qu'un joueur joue on fait des segments autour de sa position dans toutes les directions pour que chaque segment n'aille pas plus loin que trois autour de l'origine puisqu'on gagne en alignant 4 pions, et que cela reste dans les limites de la matrice. Finalement, on prend une portion de longueur 4 sur les segments et on la décale jusqu'à atteindre le bout en regardant à chaque fois s'il y a 4 du même joueur. Pour ce qui est de l'IA, lorsque c'est son tour elle utilise un algorithme récursif appelé *minimax* et elle joue toutes les positions possibles, attribue un score en fonction des différents pions présents puis joue toutes les possibilités de l'adversaire en attribuant un score négatif et ainsi de suite jusqu'à une certaine profondeur puis on remonte l'arbre en choisissant le score le plus grand à chaque étape pour qu'au final le résultat de la somme de tous les scores soit le plus grand possible.

IV. Coût du projet :

Le coût d'un projet ne réside pas uniquement dans le coût du matériel mais il faut également prendre en compte le coût « humain ». En effet, la réalisation de ce projet nécessite de la réflexion et de la préparation, c'est-à-dire du temps et comme ce sont des

ingénieurs qui donnent de leur temps et tout travail étant rémunéré il ne faut pas oublier de prendre en compte cela.

A. Matériel

Le matériel que nous avons utilisé est le suivant :

1 carte Arduino Uno



3 boutons poussoirs



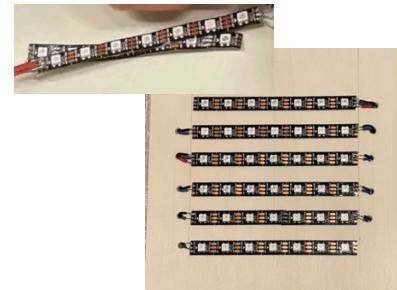
1 batterie portative



2 plaques de bois de 18cmx18cm en 3mm



7 bandes de LEDs



- 2 résistances, 1 feuille de papier brillant diffuseur de lumière, 1 boîte en bois, 6 vis et 4 entretoises, des fils électriques

Matériel	Coût unitaire	Quantité	Coût total
Carte Arduino Uno	~ 30€	1	30 €
Batterie portative	~ 25€	1	25 €
Bandes de LED	(Prix au mètre) ~6,60€	1,1 m	7,26 €
Boutons poussoirs	0,9 €	3	2,7 €
Résistances	~ 0,2€	2	0,4 €
Papier brillant diffuseur de lumière	~ 1€	1	1 €
Plaques de bois de 18cmx18cm en 3mm	~2,1€	6	12,6 €
Vis	0,04	6	0,24 €
Entretoises	0,05	4	0,2 €
Fils électriques	(Prix au mètre) ~1€	~ 2m	2 €
Machine à découpe laser			~ 5€
			86,4 €

B. Ingénieur

Le temps de travail sur le projet est estimé à : 65h. Soit les 8 séances de 3h, (24h) plus du temps en dehors des séances prévues à cet effet pour avancer le travail et surtout écrire le code.

Le salaire moyen d'un ingénieur étant d'environ 2 900€/mois à raison de 35h par semaine cela fait environ 140h par mois soit environ 21 euros de l'heure.

Donc : $65h \times 21\text{€} = 1365\text{€}$

On a finalement le coût ingénieur qui est évalué à : $2 \times 1050\text{€} = 2730\text{€}$.

Le coût total de notre projet est donc estimé à environ 2817 €.

V. Plannings : prévisionnel et réel

Voici ci-dessous le planning prévisionnel (représenté avec les lignes de couleur grise) ainsi que la planning réel (représenté avec les lignes de couleurs blanches).

Séance numéro	Logan		Saveria	
1	Recherche du matériel nécessaire	Recherche du matériel + écriture du code	Recherche du matériel nécessaire	Absente (malade)
2	Recherche du matériel + écriture du code	Écriture de la base du code	Reflexion sur la structure	Absente (malade)
3	Écriture de la base du code	Écriture de la base du code	Modélisation de la structure	Réflexion à la conception de la structure
4	Adaptation du code à Arduino	Adaptation du code à Arduino	Construction de la structure	Modélisation de la structure (sur Onshape)
5	Écriture de l'IA	Continuation de l'adaptation à Arduino	Construction de la structure	Découpe laser de la 1ère planche puis fixation des LEDs dessus

Séance numéro	Logan		Saveria	
6	Écriture de l'IA	Écriture de l'IA	Construction	Soudure sur les bandes de LEDs + découpe du socle
7	Assemblage structure - code	Écriture de l'IA + assemblage structure code	Assemblage structure - code	Assemblage de la partie verticale de la structure + collage socle
8	Personnalisation des couleurs	Assemblage de toutes les parties ensemble	Assemblage et personnalisation couleurs	Assemblage de toutes les parties ensemble

VI. Les problèmes rencontrés

Il est évident qu'au cours d'un projet tel que celui-ci tout ne se passe pas sans accroc ou imprévu.

Il est vrai que durant plusieurs séances nous avons été limités par la disponibilité du personnel ainsi que des outils du FabLab. Ce qui est normal puisque nous étions une vingtaine d'élèves avec chacun des besoins et devions tous nous adresser à la même personne. Mais quelques fois pour pouvoir avancer sur la suite, nous avions besoin de réaliser une étape clé sauf que lorsque l'attente était longue cela pouvait être pénalisant pour la suite de nos missions. Par exemple, pour commencer la fabrication de notre structure, il fallait absolument découper la plaque de bois sur laquelle allaient être fixées toutes les bandes de LED pour ensuite pouvoir souder des fils sur chaque bande et ensuite pouvoir tout relier à la carte Arduino pour tester le code. Seulement la machine à découpe laser était très sollicitée par tout le monde et nous avons pu découper notre planche à la presque fin de notre séance... Ce qui engendre donc une autre difficulté qui a été la gestion du temps.

Il se trouve que souvent, certaines missions que nous nous étions fixées avaient une durée bien supérieure à celle que nous avions estimée ce qui engendrait du travail supplémentaire hors des séances. Un exemple concret : l'assemblage à la toute fin. Pour assembler toutes les parties ensemble il fallait d'abord assembler les plaques avec le socle, puis couper les fils à la bonne distance, avant cela il fallait donc passer tous les fils dans le trou dédié à cela sur le socle, sauf que pour savoir la bonne distance il fallait fixer la carte Arduino dans le socle, pour fixer la carte Arduino il fallait savoir comment faire, ensuite la carte Arduino n'était pas au bon niveau donc on a du rajouter un épaisseur de scotch et ainsi de suite avant de réussir à tout assembler.

Enfin, une des dernières difficultés majeures rencontrée était d'adapter le code du jeu qui était écrit en C++ pour compiler sur Arduino. Ayant déjà programmé en C++ mais jamais pour Arduino, nous avions écrit un programme qui compilait grâce à GCC sur architecture 64-bits ce qui a permis de prendre de l'avance le temps que la structure soit construite. Comme la librairie standard de C++ fournie un grand nombre d'outils très utiles nous en avons fait usage. Mais lorsque nous avons voulu faire tourner le programme sur Arduino une fois la structure opérationnelle, nous nous sommes rendu compte que la librairie standard de C++ n'était pas présente, puisqu'Arduino utilise un autre compilateur, AVR-GCC, plus minimaliste puisque les cartes Arduino ont bien moins de puissance que les vrais ordinateurs. Il a donc fallu recréer les outils de cette librairie pour ainsi adapter le code pour Arduino.

VII. Conclusion

A. Rendu final

Nous avons réussi à réaliser quasiment tout ce que nous avions l'intention de réaliser. En effet, notre jeu est fonctionnel, nous avons pu jouer plusieurs parties.

Nous sommes plutôt satisfaits de ce que nous avons produit car nous avons réussi à aller jusqu'au bout de notre projet. Le jeu fonctionne correctement, toutes les soudures ont donc tenu et le code était bien fonctionnel puisque le jeu marchait et à chaque fin de partie l'animation se lançait et le jeu s'actualisait pour recommencer une partie.

B. Apports du projet

Ce projet nous a permis tout d'abord d'avoir une première vraie vision globale de ce que peut être la mise en place et la concrétisation d'un projet dans notre futur métier d'ingénieur. Grâce à ce projet, nous avons acquis de nouvelles compétences, notamment en termes de travail de groupe, organisation du travail, gestion des imprévus et méthodologie. De plus, cela nous a permis d'avoir de nouvelles compétences techniques comme celle de réaliser des soudures (aucun d'entre nous n'avait déjà fait cela avant), utiliser de nouveaux logiciels de modélisation,... . Enfin ce projet a également été l'occasion de mettre en pratique les connaissances théoriques que nous avons acquis durant la PeiP.

C. Perspectives d'améliorations

Si nous devions retravailler des éléments sur notre projet ce serait peut être d'une part la structure. Avec plus de temps nous aurions pu imprimer toute la structure en 3D afin d'avoir un socle plus solide et donc durable dans le temps et ce qui aurait également permis un assemblage plus « clean » de toutes les parties.

De plus, l'intelligence artificielle a de temps en temps des bugs et ne joue pas de manière optimale. En effet, certaines fois elle peut gagner en un seul coup mais elle ne joue pas ce coup en question. Il est possible que ce soit du à un stack overflow puisque cela ne c'est jamais produit sur la version ordinateur du jeu et on manipule des très grands nombres pour les coups gagnants au niveau de l'attribution de score pour forcer l'IA à jouer à cet endroit, or les int sur Arduino sont stockés sur bien moins de bits que sur ordinateur à cause de la différence de RAM disponible, il est donc possible que ça vienne de là, auquel cas utiliser des long int à la place pourrait tout régler. Ou un autre bug que nous avons remarqué à la fin de notre projet est que l'intelligence artificielle gagne certaines parties alors que les 4 pions ne sont pas alignés, ce qui paraît bien étrange puisque la méthode qui vérifie si un joueur gagne ne fait pas la différence entre une IA ou un vrai joueur mais impossible de recréer ce mal fonctionnement en tant que joueur. Bien que ce soient des choses qui nous sont arrivées à une faible fréquence, il serait intéressant de se pencher plus sur ces bugs afin d'essayer de les résoudre.

Enfin, afin d'améliorer notre projet et d'apporter de l'intérêt à ce dernier nous pourrions créer un mode de sélection des couleurs afin de pouvoir personnaliser un peu plus l'expérience notamment grâce à une application mobile par exemple. L'application mobile était prévue initialement mais n'a pas été faite par manque de temps, le code a été fait tel qu'il est très simple de changer si les joueurs 1 ou 2 sont des IA ou des vrais joueurs, ainsi si l'application avait été faite on aurait pu choisir de jouer à 2 joueurs ou 1 joueur ou même à 0 joueurs et laisser 2 IA s'affronter. Pour ce qui est de la personnalisation, nous étions face à un problème, c'est que chaque fois que la carte Arduino redémarre toutes les variables sont effacées donc comment stocker le choix des couleurs envoyés depuis l'app ? Après quelques recherches nous avons trouvé la solution : utiliser la EEPROM (Electrically Erasable Programmable Read-Only Memory) d'Arduino. Il s'agit d'une mémoire permanente et qui donc stocke des données même lorsque la carte Arduino est débranchée. Dans le code cela se présente sous la forme d'un tableau d'entiers dans lequel on peut modifier les valeurs assez aisément en indiquant l'adresse. Or les entiers dans ce tableau ne peuvent excéder la valeur 255 mais c'est exactement ce dont nous avons besoin pour stocker les valeurs RGB des couleurs.