Institut Supérieur
d'Électronique de Paris

TIPE

Bras mécanique et sudoku

					3			
5			7	1		2		
	4			9		8	6	
		3						8
	5	2	6			9	3	
9		7		3			5	
			3		5			
6				2		1		
	9		1				4	

Laurent Tainturier & Alphonse Terrier
supervisé par
M. Patrick COUVEZ

2016-2017

Table des matières

Introduction

[1]

1. Sauf mention contraire, les images et figures ont été réalisées par nos soins.

Chapitre 1

Présentation et faits sur le sudoku

Le sudoku est un jeu sous forme de grille inspiré du carré latin et défini en 1979 par Howard Garns.

Cette grille est carrée et est divisée en n^2 régions de n^2 cases. Elle possède ainsi n^2 colonnes, n^2 lignes et n^4 cases. Dans la version la plus commune : $n = 3$. On appelle section, une ligne, une colonne ou une région

Une grille de sudoku est préremplie, le but du jeu étant de la compléter selon la règle suivante :

– Chaque section doit contenir au moins une fois tous les de 1 à n^2.

Une grille est considérée comme sudoku seulement si sa solution est unique.

Le minimum de cases remplies au préalable pour espérer que la dite solution soit bien unique est de 17, cela a été prouvé par une équipe islandais en 2012.

Chapitre 2

Électronique

2.1 Moteurs pas-à-pas

Nous avons utilisés dans ce projet deux moteurs pas-à-pas qui présentaient, par rapport à d'autres types de moteurs, les avantages suivants :

- une précision bien supérieure à celle de moteurs à courant continu ;
- un couple bien plus important que celui de servomoteurs ;
- mis sous tension, un déplacement fortuit du moteur n'est pas possible.

Les moteurs pas-à-pas sont notamment utilisés dans les systèmes nécessitant une grande précision comme les imprimantes 3D dans lesquelles ils sont très largement employés.

2.2 Drivers l293d

Pour piloter les moteurs, nous utilisons des drivers l293d, qui intègrent un pont en H permettant le contrôle des moteurs dans les deux sens de rotation.

2.3 Schéma électrique

Le schéma ci-joint représente les principales connections au sein de notre montage, même s'il ne présente ni l'écran LCD, ni le détail de l'alimentation (notamment du convertisseur de tension).

Les moteurs pas-à-pas sont constitués de deux bobines qui sont reliées à deux drivers l293D . Ceux-ci sont pilotés par quatre sorties GPIO qui envoient des impulsions au driver qui alimente les bobines en 12V à tour de rôle, ce qui permet à la fois de contrôler le sens et la vitesse de de rotation des moteurs, en choisissant à quelle fréquence envoyer les impulsions.

2.4 Premiers montages

Nous avons réalisés nos premiers montages avec une breadboard facilitant les tests. Une simulation du schéma a également été réalisée sous *Fritzing*

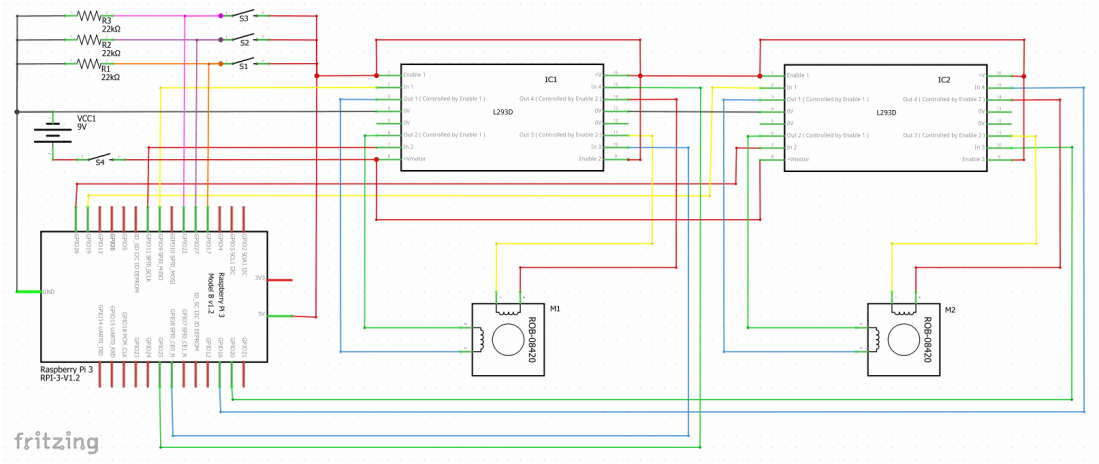

FIGURE 2.1 – Schéma électrique réalisé avec *Fritzing*

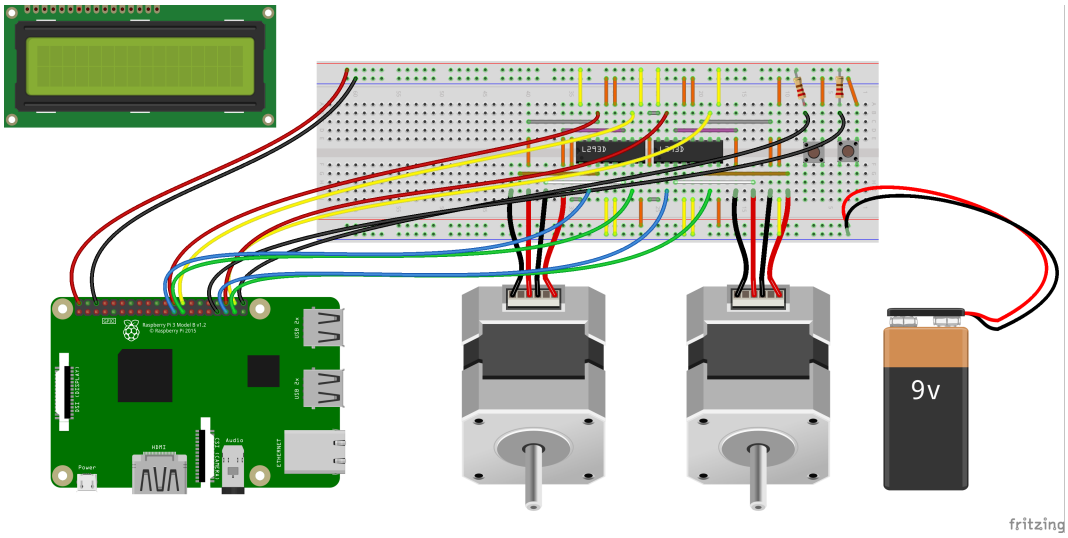

FIGURE 2.2 – Premier montage

2.5 Circuit imprimé

Pour rendre notre bras mécanique plus compact et ainsi rentrer dans le cadre de l'optimalité, nous avons souhaité remplacé la breadboard par une solution bien plus compacte, à savoir un circuit imprimé. Celui-ci sera enficher directement sur les ports GPIO de la Raspberry Pi. Nous avons de nouveau utilisé pour le réaliser le logiciel *Fritzing*, permettant la réalisation du circuit sur deux couches (symbolisées par les couleurs orange et jaune), permettant un cablage plus facile, car permettant les croisements. Nous avons utilisé *Fritzing* car il été assez facile de faire faire fabriquer le circuit imprimé pour une dizaine d'euros. Nous avions hésité à réaliser entièrement ce circuit par nous même, mais du fait de la présence de deux couches, cela se serait révéler très difficile.

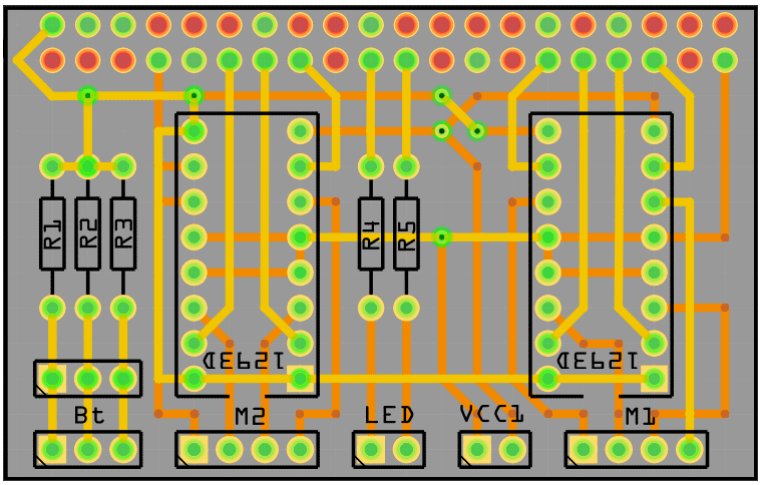

FIGURE 2.3 – Circuit imprimé réalisé avec *Fritzing*

2.6 Écran LCD

Pour rendre le bras mécanique autonome, nous avons intégré un écran LCD permettant à l'utilisateur de se repérer dans l'évolution des différents scripts sans disposer nécessairement d'un ordinateur ou d'un écran à proximité.

FIGURE 2.4 – Écran LCD affichant le message de bienvenue

Chapitre 3

Mécanique

3.1 Choix concernant la structure du bras

La premier défi qui s'est imposé à nous a été le choix d'une structure adéquate : solide et pratique. Notre choix s'est alors porté sur les Makerbeam. MakerBeam est un système de construction Open-Source basé sur des profilés ALU en T.

L'écriture se fera en coordonnées polaires et non pas en coordonnées cartésiennes pour les raisons suivantes :

- un système en coordonnées cartésiennes aurait été trop encombrant et un moins "portable" (dans un contexte notamment d'optimalité, comme exigé par le programme)
- c'est un défi technique de fabriquer un bras de ce type en coordonnées polaires

Ainsi, un moteur pas-à-pas assurera la rotation θ du bras. Ce moteur s'auto-entrainera autour d'une roue dentée de 128 dents comme celle-ci et un roulement permettra la rotation de l'entièreté du bras :

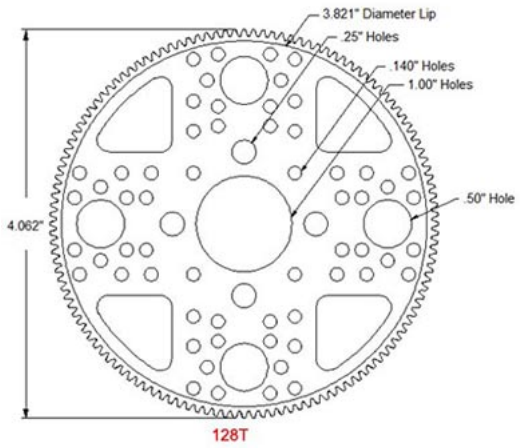

FIGURE 3.1 – Schéma de la roue dentée

De même la translation r a été assurée par un deuxième moteur pas-à-pas et un système courroie/poulie. Cette translation s'est faite le long de deux axes de 272 mm et de 8 mm de diamètre.

3.2 Difficultés rencontrées et solutions mises en place

Nous avons notamment rencontré des difficultés concernant le roulement utilisé. En effet, nous avions au préalable utilisé un roulement à billes comme celui-ci :

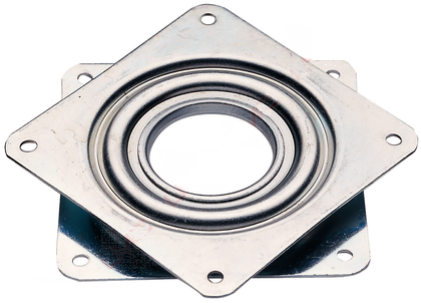

FIGURE 3.2 – Roulement défectueux

Malheureusement, ce roulement tournait très mal et saccadait. De plus, du jeu était présent. Après quelques heures de recherche, nous avons donc choisi le roulement à billes suivant, qui tourne très bien mais qui tout de même possède lui aussi un peu de jeu :

FIGURE 3.3 – Roulement finalement choisi

3.3 Conception du chariot en 3D

Nous avons pris la décision de concevoir certaines pièces de notre projet en 3D, celle-ci nécessitant de la précision. Le logiciel professionnel de modélisation SolidWorks, développé par DassaultSystèmes, a été utilisé pour développer ces pièces.

Les mises en plan des pièces composant le chariot sont disponibles en annexe **??** à partir de la page **??**.

La partie principale du chariot devait pouvoir accueillir un servomoteur, permettant l'abaissement et le soulèvement du stylo :

Ce stylo sera fixé sur la pièce présentée en page**??** dont l'impression 3D est montrée ci-dessous :

L'ensemble chariot/support du stylo imprimé en 3D est alors montrée ci-dessous :

Le chariot permet, de façon modulaire, d'accueillir un support caméra se divisant en deux pièces présentées en page **??** et en page **??**.

Chapitre 4

Informatique

4.1 Présentation globale

Tous les algorithmes développés dans le cadre de ce projet sont disponibles en annexe. Ils ont été, pour la plupart, développés sous Python 3, les autres sous Python 2 car certaines bibliothèques dont nous avions besoin, notamment pour la reconnaissance, n'était disponible que sous Python 2.

Nous avons développé une programmation modulaire, permettant de travailler simultanément sur le projet, sans pour autant poser de problème de logistique. Ainsi nous avons dissocié tous les scripts ; que ce soit la reconnaissance, la résolution, l'affichage, la gestion de la caméra ou des servo-moteurs, etc. Pour cela, nous avons développé une relation qualifiable de maître-esclave entre nos scripts. Chacun des scripts dépend d'un fichier principal, appelé *main*, qui récupère les informations des autres scripts et donne les ordres adéquats à ceux-ci, selon la situation. Ainsi, les scripts ne sont pas reliés les uns aux autres mais seulement à ce script principal, ce qui permet d'ajouter ou d'enlever très facilement tel ou tel script, sans pour autant altérer le fonctionnement de l'ensemble, permettant ainsi de tester chacun des scripts très facilement.

Il nous apparaissait de plus nécessaire de développer une programmation permettant l'exécution simultanée de plusieurs instructions, notamment pour les moteurs. En effet, les moteurs doivent absolument fonctionner de manière synchrone afin des déplacements les plus précis possibles. C'est pour cela que nous nous sommes basés sur ce qui pourrait être qualifiée de "programmation parallèle" qui permet l'exécution de plusieurs processus simultanément. Cela est rendu possible en Python par l'utilisation de la bibliothèque *threading* depuis laquelle nous utilisons *Thread* (signifiant *processus* en français).

4.2 Résolution du sudoku

Pour résoudre une grille de sudoku, un joueur utilise différentes méthodes de résolution, qui sont purement algorithmiques. Il utilise en grande majorité deux méthodes qui peuvent permettre de résoudre une grande partie des grilles disponibles sur le marché. On appelle ces deux méthodes *inclusion* et *exclusion*. Cependant les grilles de niveau supérieur font appel à des méthodes plus subtiles et souvent plus difficiles à appliquer en pratique qui se basent généralement sur des *paires* ou des *triplets*. Les grilles les plus difficiles peuvent

imposer au joueur de faire un choix entre plusieurs possibilités pour espérer mener à son terme la résolution. C'est sur ce constat que ce base la méthode dite de *backtracking* (ou *retour sur trace*), qui permet de résoudre toute grille de sudoku, même si celle-ci n'est pas valide et possède plusieurs solutions. Bien sûr cette liste n'est pas exhaustive et il existe d'autres méthodes de résolution, même si ces méthodes ne sont pas forcément très employés faciles d'emploi.

Inclusion : un seul candidat pour une case On dénombre pour chaque case vierge les chiffres pouvant être positionnés dans celle-ci. S'il n'y a qu'une seule possibilité alors elle est solution et est donc placée dans la case.

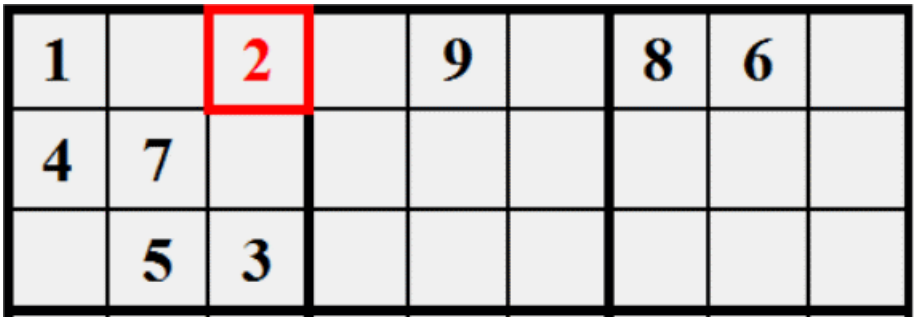

FIGURE 4.1 – Méthode d'inclusion pour une région

Exclusion : candidat unique pour une section Si pour un chiffre donné, on ne peut le placer dans une section que dans une seule case possible, alors ce chiffre est placé dans cette case

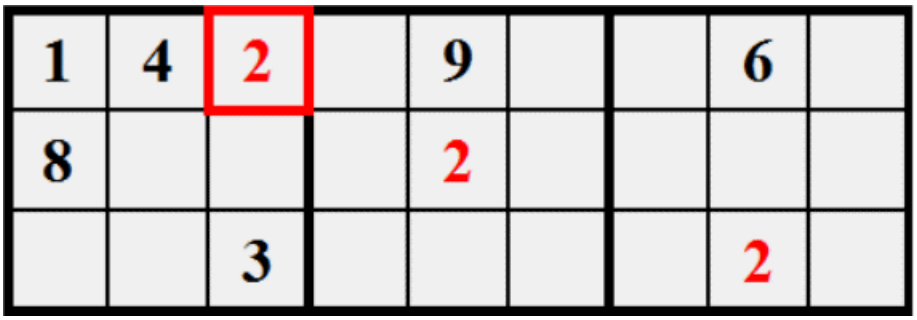

FIGURE 4.2 – Méthode d'exclusion pour une région

Paires Si dans une section donnée, deux cases ne contiennent que deux *candidats identiques*, alors ils ne peuvent être placés que dans ces deux cases, ce qui permet de les éliminer des autres possibilité pour la section

FIGURE 4.3 – Méthode des paires pour une ligne

La méthode des triplets est une généralisation de cette méthode à trois cases et non seulement deux.

Jumeaux Si dans une section, un chiffre ne peut être placé que sur une ligne (resp. colonne), alors ce chiffre peut être supprimé des possibilités pour le reste de la ligne (resp.colonne). Cette méthode ne permet pas toujours de placer immédiatement celui-ci, mais permet de réduire les possibilités pour placer les autres.

18	5	16	268	32	38	36	69	7
7	89	3	236	4	26	1	89	9
2	2	4	~~2~~37	9	~~2~~7	~~2~~36	~~2~~6	~~2~~6

FIGURE 4.4 – Méthode d'exclusion pour une région

Backtracking Cette méthode consiste tout d'abord à lister pour chacune des cases vierges les chiffres qui pourraient correspondre. On part d'une case vierge quelconque qu'on remplit par un des chiffres pouvant théoriquement convenir, puis on liste les nouvelles possibilités des autres cases en fonction du chiffre précédemment ajouté puis on passe à la case suivante. S'il n'y a plus aucune possibilité, on revient sur nos pas et on change le chiffre qu'on venait d'ajouter en une autre possibilité. Si tout les chiffres possibles ont été testé, on revient de nouveau sur nos pas autant de fois que nécessaire, jusqu'à avoir compéter la totalité de la grille. Cette méthode ne peut pas être mise en oeuvre par un joueur, dans la mesure où elle lui demanderait énormément de temps, et ne présenterait pour lui aucun intérêt. Cependant, elle peut être indispensable pour les grilles diaboliques lorsque le joueur doit faire un choix, cette méthode est alors indispensable [1].

Partant de ce constat, nous avons tout d'abord implémenté la méthode de *backtracking*, ainsi, il nous était possible de résoudre toutes les grilles possibles. À l'origine, notre algorithme parcourait la grille dans un ordre prédéfini, à savoir de haut en bas et de gauche à droite. Cependant, il pouvait arriver que pour certaines grilles, ce système n'était pas tout à fait optimale. C'est pourquoi, nous avons changé ce système, de sorte que l'algorithme commence par les cases présentant le moins de possibilités, permettant dans le cas général de diminuer le temps de résolution. Cependant, dans le pire des cas, cette solution ne présentait aucune différence en terme de temps de calculs.

1. Une animation montrant le fonctionnement de cette méthode est disponible `www.github.com/alphter/Sudoku-Plotter/blob/master/pictures/backtracking.gif`

4.3 Reconnaissance du sudoku

Le script de reconnaissance du sudoku a été réalisé sous Python 2 avec le module de traitements d'image OpenCV. Il a été réalisé pour :

- Reconnaître les chiffres dans une grille du sudoku
- Déterminer la position spatiale de la grille

On photographie la grille avec une caméra Raspberry Pi (V2) comme celle-ci :

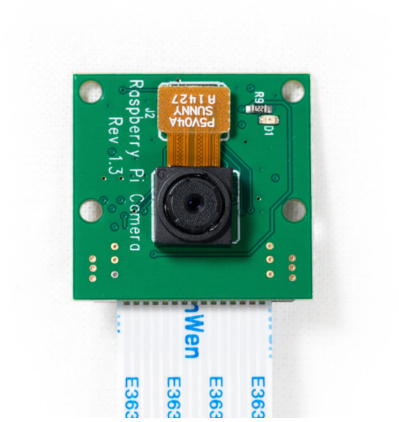

FIGURE 4.5 – Caméra Raspberry Pi V2

Voici la grille de sudoku qui nous servira d'exemple pour montrer toutes les actions du script :

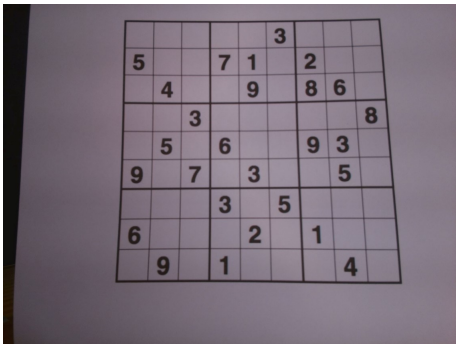

FIGURE 4.6 – Exemple de grille de sudoku

On applique sur cette photographie un filtre de type seuil (en anglais "threshold") qui va ensuite nous permettre de détecter les contours de la grille :

Par cette transformation, on peut ensuite déterminer des équations de droites des contours extérieurs de la grille. Les coordonnées des intersections des droites seront celle des coins de la grille.

On découpe alors la grille de la photographie initiale en supprimant les éventuels effets de perspective.

On découpe chaque petite case de la grille comme ci-dessous :

On utilise ensuite un module de reconnaissance de digits pour détecter les chiffres et on obtient la grille suivante, prête à être résolue :

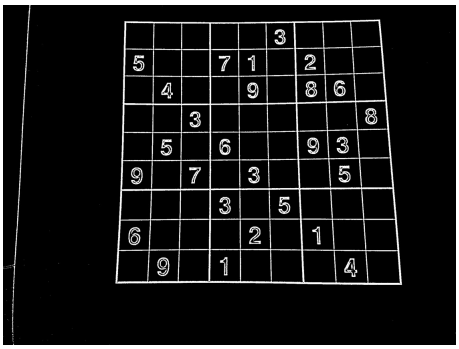

FIGURE 4.7 – Exemple de grille "seuillée"

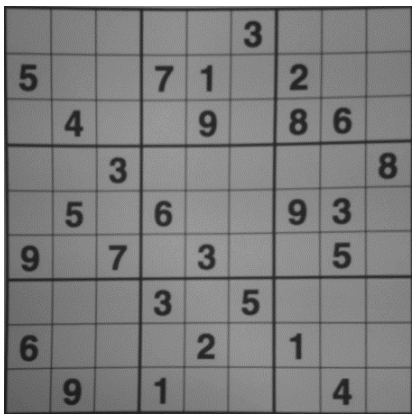

FIGURE 4.8 – Exemple de grille découpée sans perspective

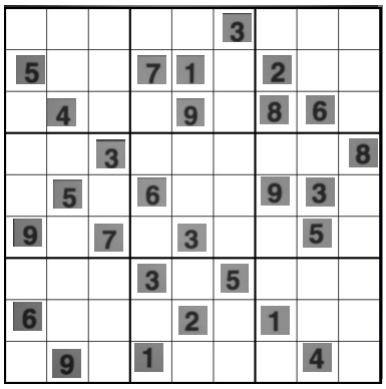

FIGURE 4.9 – Exemple de grille où chaque chiffre a été découpé

4.4 Interface graphique

Pour rendre la résolution plus simple d'utilisation, nous avons décidé d'ajouter une interface graphique. Le cahier des charges qu'elle devait vérifier était assez stricte. Elle devait pouvoir afficher, avant toute chose, une grille de sudoku qui devait dès lors être facilement modifiable. Pour cela, nous avons donc créé différents menus ; l'un permettant donc l'édition de la grille, un autre permettant de choisir la méthode de résolution, ainsi qu'un menu de résolution.

					3			
5			7	1		2		
	4			9		8	6	
		3						8
	5	2	6			9	3	
9		7		3			5	
			3		5			
6				2		1		
	9		1				4	

FIGURE 4.10 – Exemple de grille prête à être résolue

Édition Lors de l'édition de la grille, un carré rouge apparait, déplaçable avec les flèches directionnelles, il suffit alors pour modifier la case de rentrer un chiffre de 0 à 9, 0 correspondant à une case vierge. Il est également possible de sauvegarder une grille ou encore de récupérer une grille déjà enregistrée.

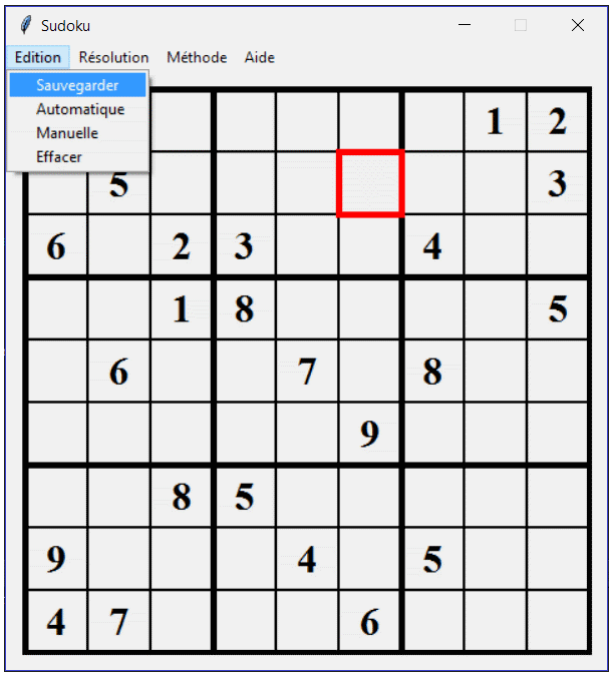

FIGURE 4.11 – Interface graphique affichant le menu Edition

Résolution Ce menu permet de lancer la résolution, et permet également de choisir entre une résolution rapide ou une résolution pas-à-pas permettant à l'utilisateur d'observer le fonctionnement de ces algorithmes.

Méthode Le choix de la méthode se fait avec le menu *Méthode*, qui permet de choisir la méthode de résolution parmi l'*inclusion*, l'*exclusion* et le *backtracking* ou de choisir une méthode dite *globale*, s'appuyant sur tous les algorithmes, permettant ainsi d'être la plus rapide possible.

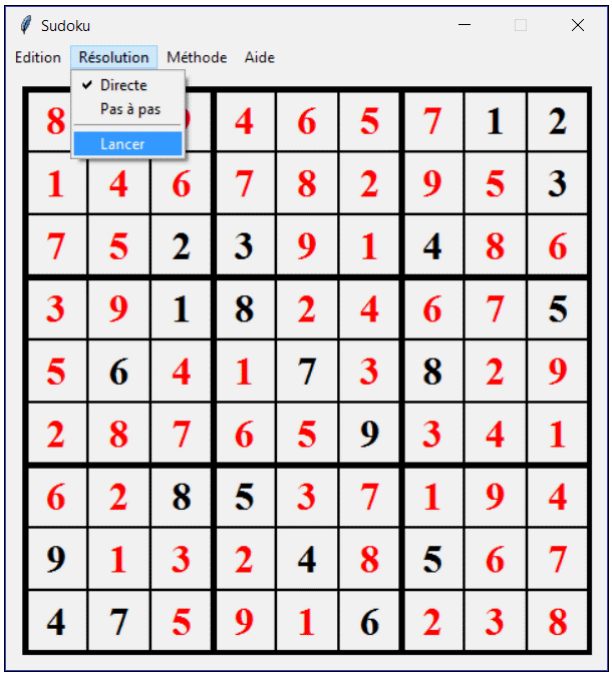

FIGURE 4.12 – Interface graphique affichant le menu Édition

4.5 Contrôle des moteurs et écriture

Moteurs pas-à-pas

Les moteurs se devaient de fonctionner simultanément pour des questions de précision évidentes. Comme nous l'avons dit précédemment [2], il suffit d'envoyer des impulsions dans les bobines à la bonne fréquence pour faire tourner les moteurs à la bonne vitesse et dans le bon sens. Ces impulsions sont envoyés au deux moteurs de manière cycliques, un cycle étant constitué de quatre séquences.

Coordonnées polaires

Par soucis de compacité et ainsi s'inscrire dans l'optimalité du programme, nous avons décidé de développer une structure se basant sur les coordonnées polaires, permettant des dimensions maximum de 40 par 10 cm. L'utilisation de telles coordonnées nécessitait une plus grande précision et relevait donc d'un plus grand défi technique. Les coordonnées polaires ne sont de plus, que très peu utilisés pour ce type de système, ce qui nous à incité à développer une structure se basant sur ces coordonnées.

Écriture des digits

Nous souhaitions dès le début pouvoir écrire des grilles de différentes tailles, et non des grilles de tailles fixes. C'est pourquoi nous avons basé le script d'écriture sur du point par point, permettant de choisir la précision théorique des chiffres tracés. Cependant les moteurs pas-à-pas ne pouvant se déplacer que

2. Cf 2.1 Moteur pas-à-pas

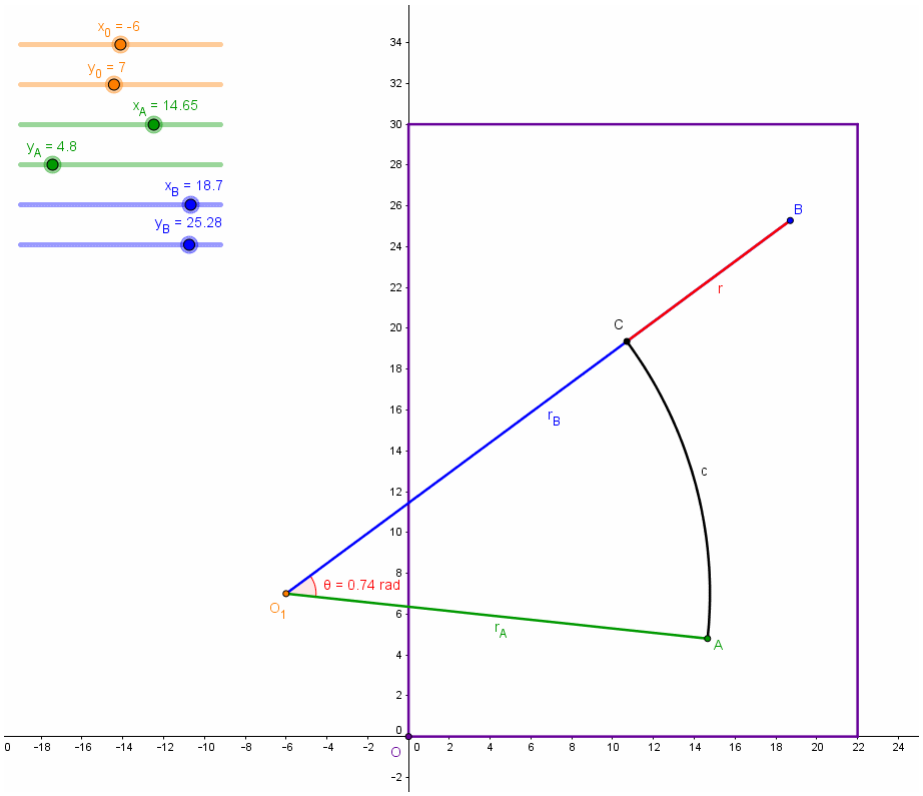

FIGURE 4.13 – Changement de repères réalisé avec *Geogebra*

d'un pas fixé de 1.8 ° , une trop grande précision ne pourrait entraînerait que des déplacements nuls des moteurs, c'est pourquoi cette manière de programmation permet un compromis entre précision théorique et précision pratique.

FIGURE 4.14 – Chiffres d'une précision théorique assez faible

En augmentant la précision théorique, on obtient des chiffres quasiment continus permettant d'obtenir la figure suivante.

Il faut prévoir dans le script la possibilité d'envoyé une impulsion au servomoteur afin qu'il lève le stylo lorsqu'un chiffre a été tracée afin d'éviter que les chiffres ne soient reliés entre eux, comme le suggère la figure suivante.

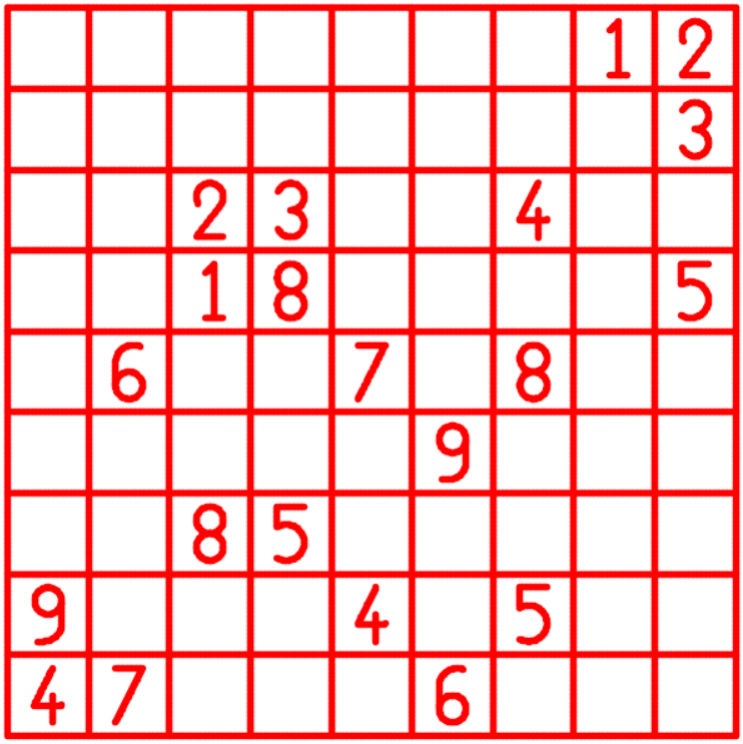

FIGURE 4.15 – Grille de sudoku en point par point tracée avec *MatplotLib*

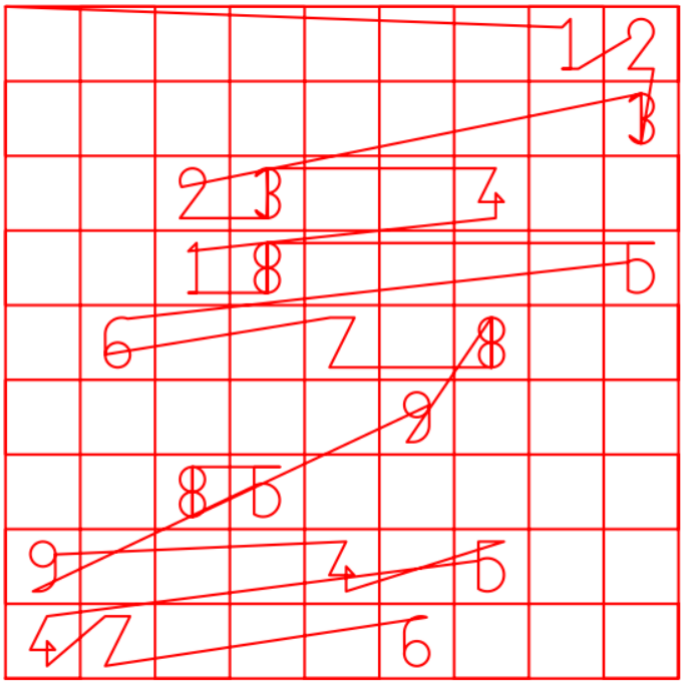

FIGURE 4.16 – Grille de sudoku tracée avec *MatplotLib*

Conclusion

Tous les scripts et ce dossier sont disponibles sur GitHub [3]

3. Sudoku Plotter - `www.github.com/alphter/Sudoku-Plotter/`

Remerciements

Ce TIPE a été très bénéfique et nous a tout d'abord permis d'améliorer les compétences acquises l'an dernier en Python.

De plus, de nombreuses nouvelles compétences ont du être mises en œuvre :

– le contrôle de moteurs à l'aide d'une Raspberry Pi

– la conception de pièces en 3D

Nous remercions M. Couvez pour ses précieux conseils et Tristan Vajente pour les impressions de pièces en 3D.