

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

<b>Introduction</b>	<b>1</b>
<b>1 Présentation du sudoku</b>	<b>3</b>
<b>2 Électronique</b>	<b>4</b>
2.1 Moteurs pas-à-pas . . . . .	4
2.2 Schéma électrique . . . . .	4
2.3 Premiers montages . . . . .	4
2.4 Circuit imprimé . . . . .	4
2.5 Écran lcd . . . . .	5
<b>3 Mécanique</b>	<b>7</b>
<b>4 Informatique</b>	<b>8</b>
4.1 Présentation globale . . . . .	8
4.2 Reconnaissance du sudoku . . . . .	8
4.3 Résolution du sudoku . . . . .	11
4.4 Interface graphique . . . . .	11
4.5 Contrôle des moteurs et écriture . . . . .	13
4.5.1 Moteurs pas-à-pas . . . . .	13
4.5.2 Coordonnées polaires . . . . .	13
4.5.3 Écriture des digits . . . . .	13
<b>A Fichier principal</b>	<b>17</b>
<b>B Script de résolution des sudokus</b>	<b>18</b>
<b>C Script de gestion de la caméra</b>	<b>19</b>
<b>D Script d’affichage du sudoku</b>	<b>20</b>
<b>E Script de gestion des moteurs pas-à-pas</b>	<b>21</b>
<b>F Script permettant l’écriture d’un sudoku</b>	<b>22</b>

# Introduction

# Chapitre 1

## Présentation du sudoku

# 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 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.

### 2.3 Premiers montages

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

### 2.4 Circuit imprimé

Pour rendre notre bras mécanique plus compact et ainsi rentrer dans le cadre de l'optimalité, nous avons souhaité remplacer la breadboard par une

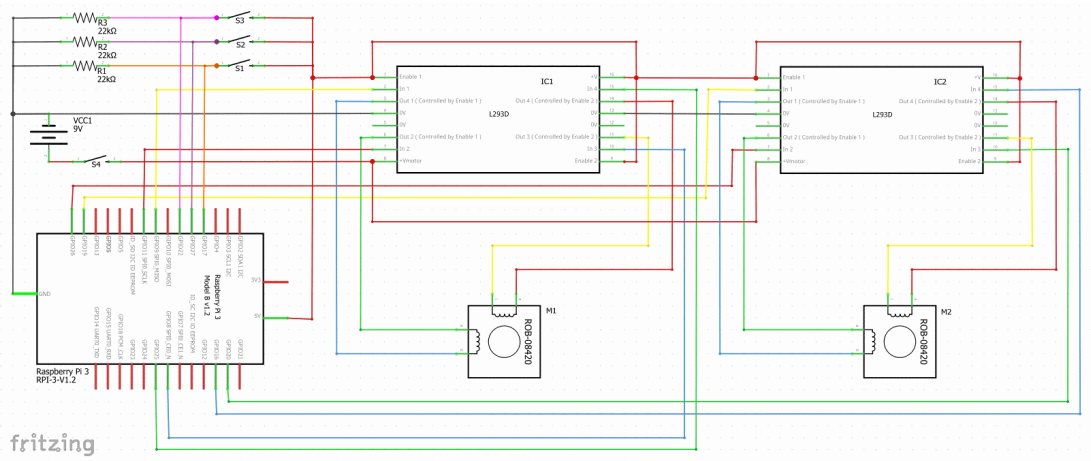


FIGURE 2.1 – Schéma électrique réalisé avec *Fritzing*

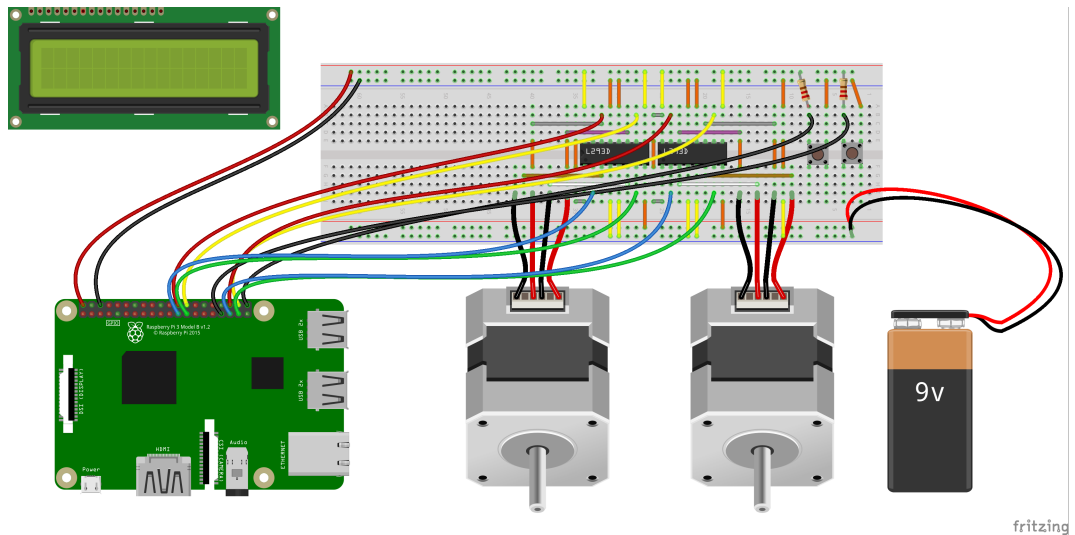


FIGURE 2.2 – Premier montage

solution bien plus compacte, à savoir un circuit imprimé. Celui-ci sera enficher directement sur les ports GPIO de la Raspberry. 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 avons hésité à réaliser entièrement ce circuit par nous même, mais du fait de la présence de deux couches, cela se serait révéler très difficile.

## 2.5 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.

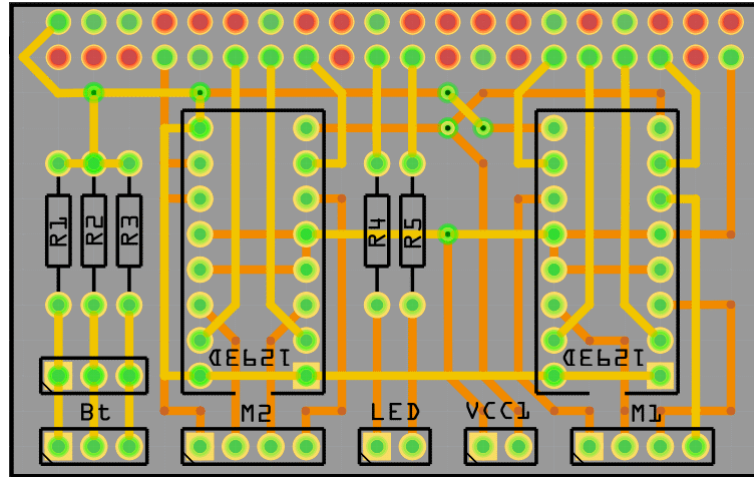


FIGURE 2.3 – Circuit imprimé réalisé avec *Fritzing*



FIGURE 2.4 – Écran lcd affichant le message de bienvenue

# Chapitre 3

## Mécanique



# 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 en Python 3, les autres se basant sur Python 2 car certaines bibliothèques dont nous avons besoin, notamment pour la reconnaissance, n'étaient disponibles 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 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, ce qui permet de tester chacun des scripts très facilement.

### 4.2 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 :

1. Reconnaître les chiffres dans une grille du sudoku
2. Déterminer la position spatiale de la grille

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

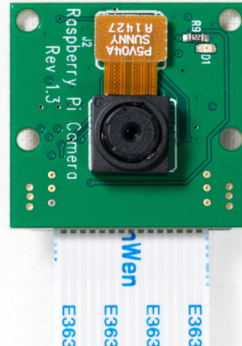


FIGURE 4.1 – Caméra Raspberry Pi V2

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

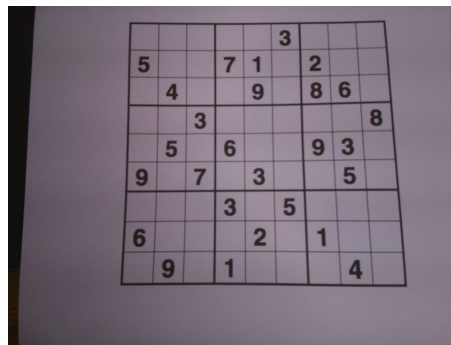


FIGURE 4.2 – Exemple de grille de sudoku

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 :

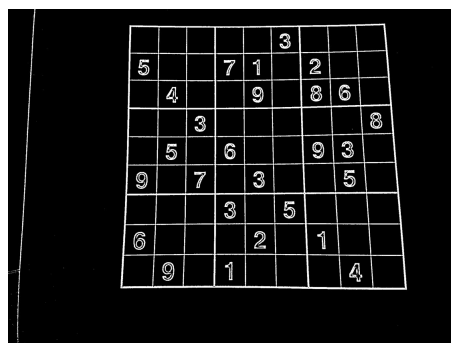


FIGURE 4.3 – Exemple de grille "seuillée"

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.

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

FIGURE 4.4 – Exemple de grille découpée sans perspective

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

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

FIGURE 4.5 – Exemple de grille où chaque chiffre a été découpé

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

					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 4.6 – Exemple de grille prête à être résolue

## 4.3 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 principales 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*. Enfin, il existe une méthode, très difficilement utilisable par un joueur, appelée *backtracking* (ou *retour sur trace*), qui permet de résoudre toute grille de sudoku, même si celle-ci possède plusieurs solutions.

### Inclusion

### Exclusion

**Paires** Cette méthode possède différentes variantes :

- 
- 

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

**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 qu'on pourrait théoriquement placer et on liste de nouveau les possibilités des autres cases en fonction du chiffre précédemment ajouté puis on passe à une autre case vierge. S'il n'y a plus aucune possibilités qui pourraient correspondre, on revient sur nos pas et on change le chiffre qu'on venait d'insérer en une autre possibilité. S'il n'y a plus de possibilités, on revient de nouveau sur nos pas autant de fois que nécessaire jusqu'à avoir compéter la totalité de la grille.

## 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.

**Edition** Lors de l'édition de la grille, un carré rouge apparaît, 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.

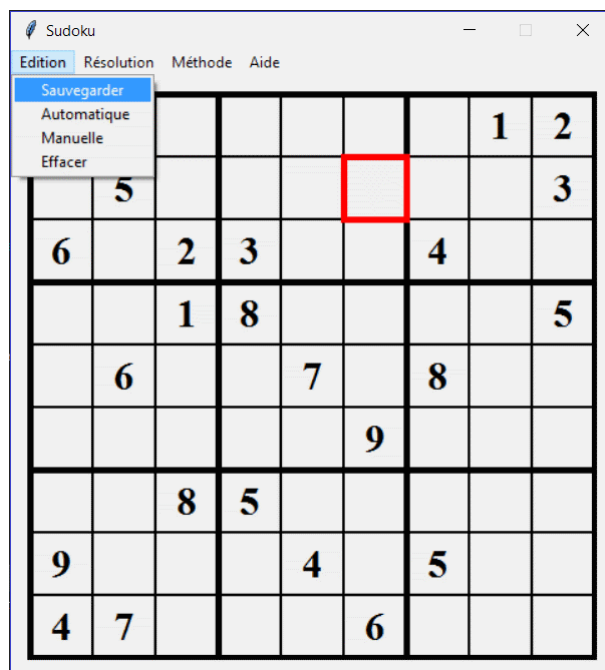


FIGURE 4.7 – Interface graphique affichant le menu Edition

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



FIGURE 4.8 – Interface graphique affichant le menu Edition

**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*<sup>1</sup> ou de choisir une méthode dite *globale*, s'appuyant sur tous les algorithmes, permettant ainsi d'être la plus rapide possible.

## 4.5 Contrôle des moteurs et écriture

### 4.5.1 Moteurs pas-à-pas

### 4.5.2 Coordonnées polaires

Par soucis de compacité de nouveau, nous avons décidé de développer une structure se basant sur les coordonnées cylindriques, permettant des dimensions maximum de 40 par 10 cm au lieu de 40 par 40 en coordonnées cartésiennes.

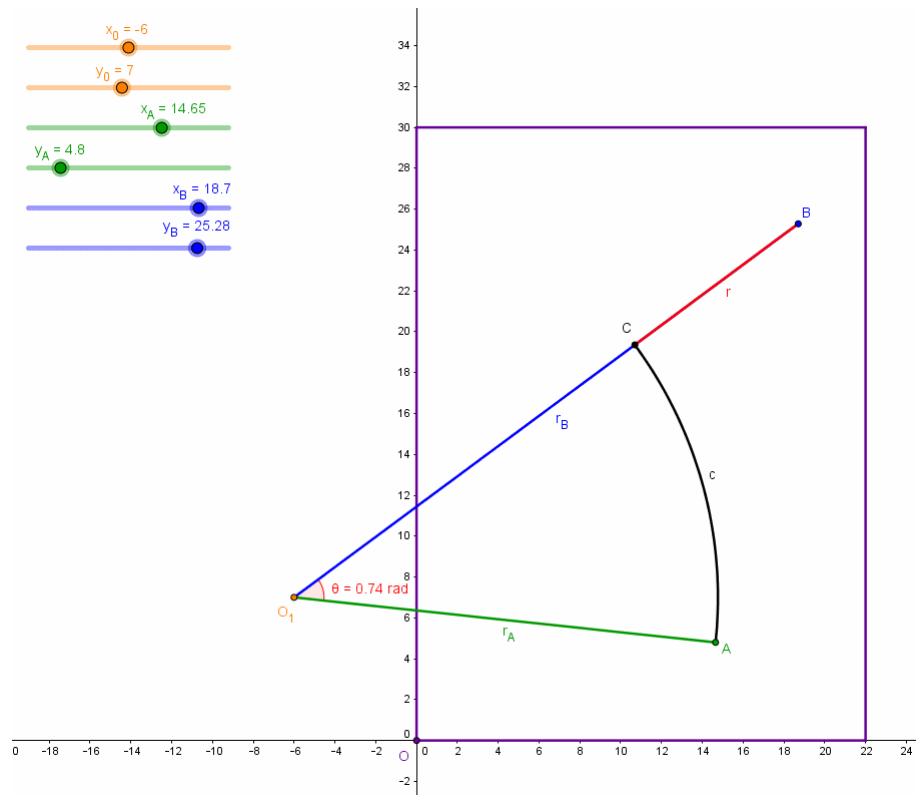


FIGURE 4.9 – Changement de repères réalisé avec *Geogebra*

### 4.5.3 Écriture des digits

1. Cf 4.3 Méthode de Résolution

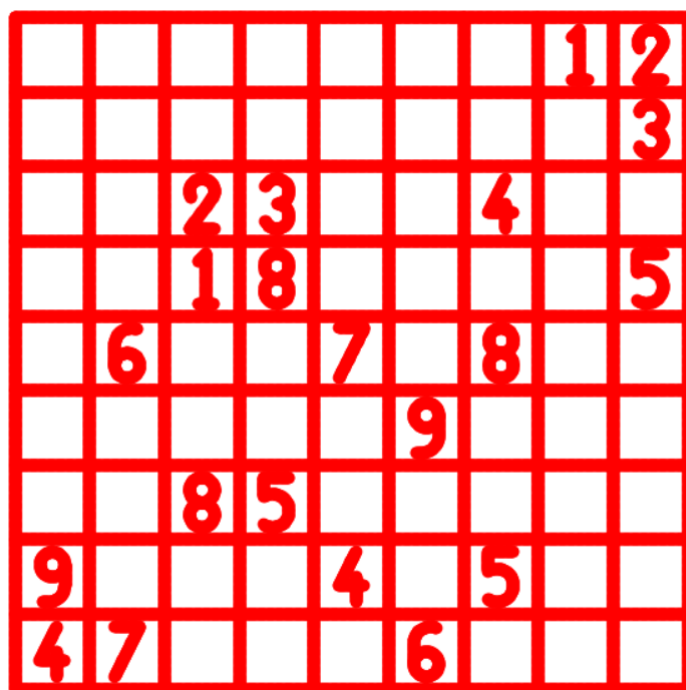


FIGURE 4.10 – Grille de sudoku en point par point tracée avec *Matplotlib*

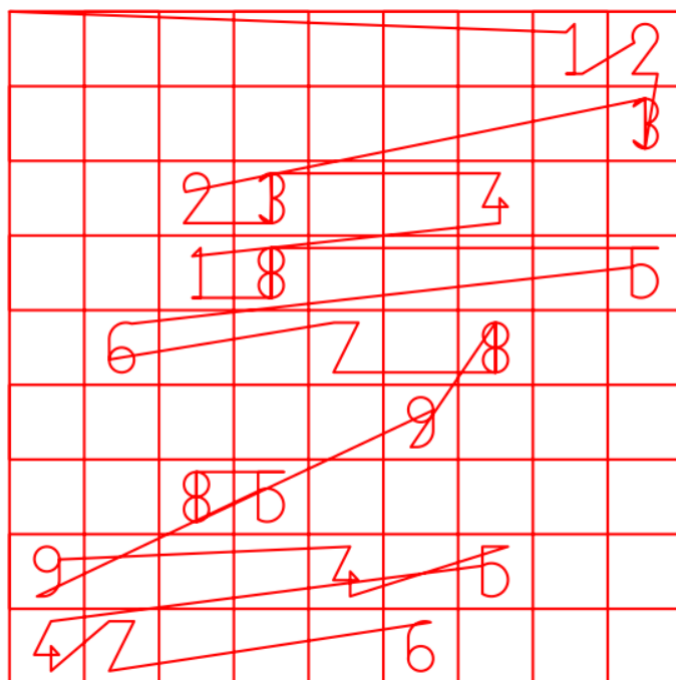


FIGURE 4.11 – Grille de sudoku tracée avec *Matplotlib*

# Conclusion



# Remerciements

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

## Annexe A

### Fichier principal

## Annexe B

### Script de résolution des sudokus

## Annexe C

### Script de gestion de la caméra

## Annexe D

### Script d’affichage du sudoku

## **Annexe E**

### **Script de gestion des moteurs pas-à-pas**

## **Annexe F**

### **Script permettant l'écriture d'un sudoku**