

# **ROBOT ASPIRATEUR : OPTIMISATION DE L'ASPIRATION, AMELIORATION DE LA TRAJECTOIRE**

PETIT Nathan

PSI Loritz

NANCY

2016/2017

# Sommaire

---

- Présentation du système
- Cahier des charges
- Modélisation sous Python
- Création de la maquette
- Programme sous Arduino
- Vérification des résultats expérimentaux
- Conclusion

# Présentation



- Automatisation du procédé de nettoyage d'une pièce.
- Peut se repérer grâce à des capteurs intégrés dans le robot.
- Evolution exponentielle du fonctionnement de ce genre de robot : de nouvelles techniques et fonctionnalités sont créées à chaque nouveaux modèles.

# Cahier des charges

Fonction	Solution
Se déplacer	Moteur
Se diriger	Capteurs
Aspirer	Turbine
S'alimenter en électricité	Batterie et prise, secteur
Eviter les obstacles	Capteurs et programmes

## Etude des fonctions voulues

Déterminer auparavant les besoins liés à l'aspirateur et les solutions pour les remplir.

Celles-ci doivent s'effectuer de manière autonome.

Toutes ces fonctions et les solutions associées ont pour but de permettre au robot de nettoyer le sol.

## Cahier des charges

	Fonctions	Critère	Niveau d'exigence
FS1	Dimensions	Passer sous les meubles	10cm max
FS2	Poids	Doit être léger	Moins de 5kg
FS3	Résistant	Doit être solide	Coque de protection
FS4	Autonomie	Batterie ou piles	1 heure au moins
FS5	Se déplacer	Moteur ou servomoteur	1 moteur par roue motrice
FS6	Efficacité	Couvrir une surface	40m <sup>2</sup> en 1 heure
FS7	Aspiration	Efficace et puissante	Aspirer miettes
FS8	Esthétique	Plaire à l'utilisateur	Couleur et forme

Ayant maintenant les solutions techniques répondant aux principales attentes, il s'agit d'en évaluer les critères que les solutions doivent respecter.

# Modélisation Python

## Robot sans mémoire

On a alors initialement une matrice du type :

0	0	2	0	0	0	2
2	0	0	2	0	0	0
0	0	0	0	2	0	0
2	0	0	0	2	0	0
0	0	2	0	0	0	0
0	2	0	0	2	0	0
0	0	0	2	0	0	0

On modélise la pièce par une matrice, avec 1 les cases où le robot est déjà passé, 0 celles qui restent à parcourir, et tout autre chiffre que 0 ou 1 étant un obstacle.

On associe une durée à chaque mouvement du robot.

On prendra pour la modélisation des pièces carrées de 40m<sup>2</sup>.



Pseudo-code du programme :

Algorithme 1 :

(x,y) sont les coordonnées du robot dans la pièce

d=direction dans laquelle le robot avance

c=(0,1,2) #0:gauche, 1:droite, 2:arrière

si trouverObstacle(x,y,n) == **True**:

    a=entierAleat(0,2)

    si trouverObstacle(x,y,c[a]) == **False**:

        avancer(c[a]) #fonction qui avance le robot et change les cases

    sinon :

        c=(1,2) #On prive c de a et on teste les 2 autres directions de la même manière

sinon:

    avancer(d) #Cette boucle s'effectue tant que le robot rencontre des obstacles

    #On effectue ce programme jusqu'à ce que la matrice ne contienne plus de 0

Lorsque le robot passera sur une case 0, celle-ci sera changée en case 1.

On stocke le nombre de déplacement pour déterminer le temps mis pour parcourir la pièce.

## Robot avec mémoire

0	0	1	1
0	2	1	0
1	0	0	1
1	1	0	0

Même principe que précédemment pour la matrice, mais les mouvements sont désormais basés sur l'algorithme de la colonie de fourmis.

Bleu: la position du robot

Rouge et magenta: cases accessibles par algorithme 1

Magenta: case privilégiée par algorithme 2

Jaune: obstacle rencontré

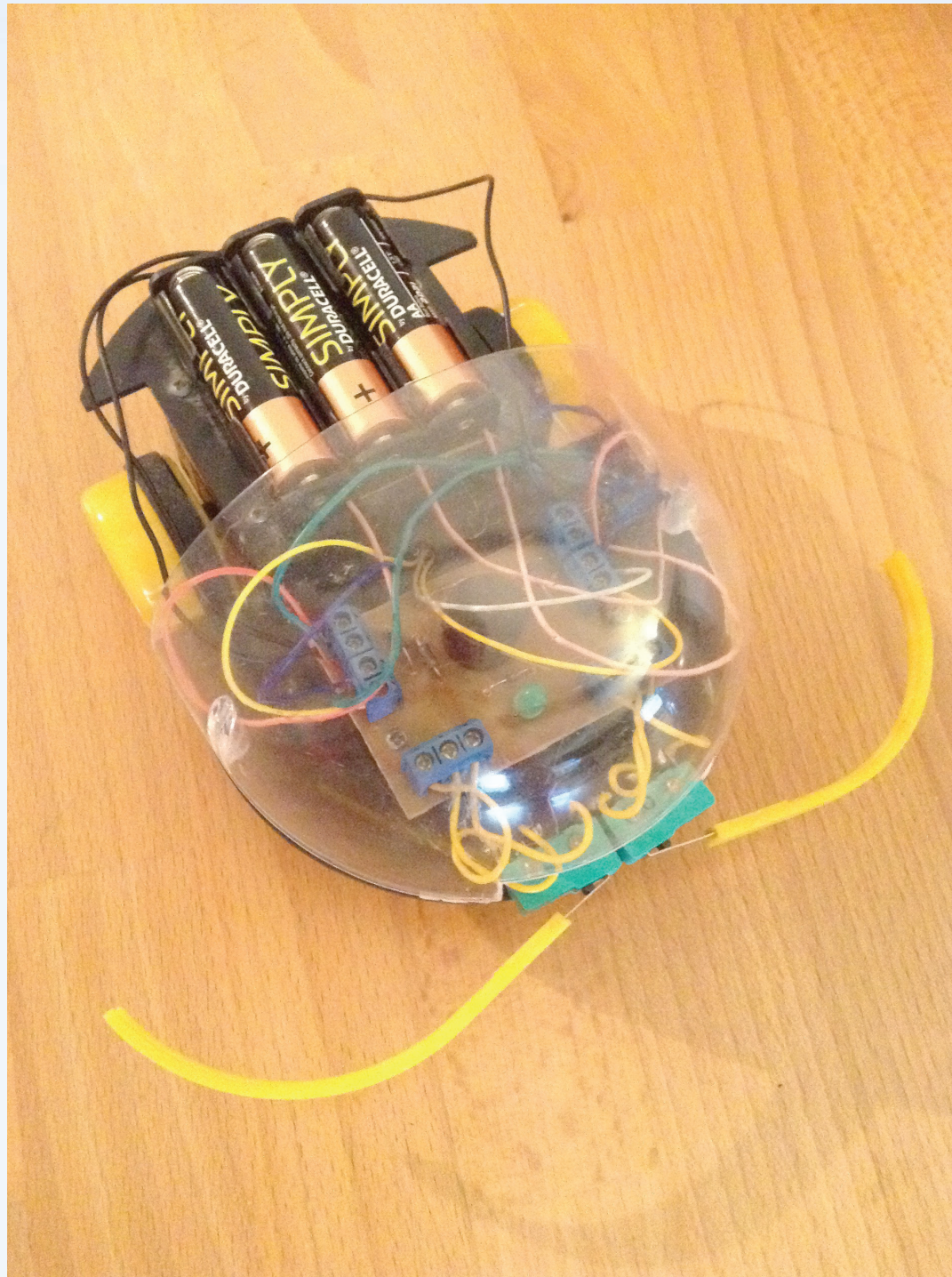


## 2 solutions possibles

Soit reprendre le premier algorithme et lui ajouter la fonction permettant de mettre en mémoire les cases déjà visitées, pour ensuite ne passer que sur celles encore non visitées.

Soit l'on applique également l'algorithme de la colonie de fourmis, mais les directions empruntées sont choisies au préalable.

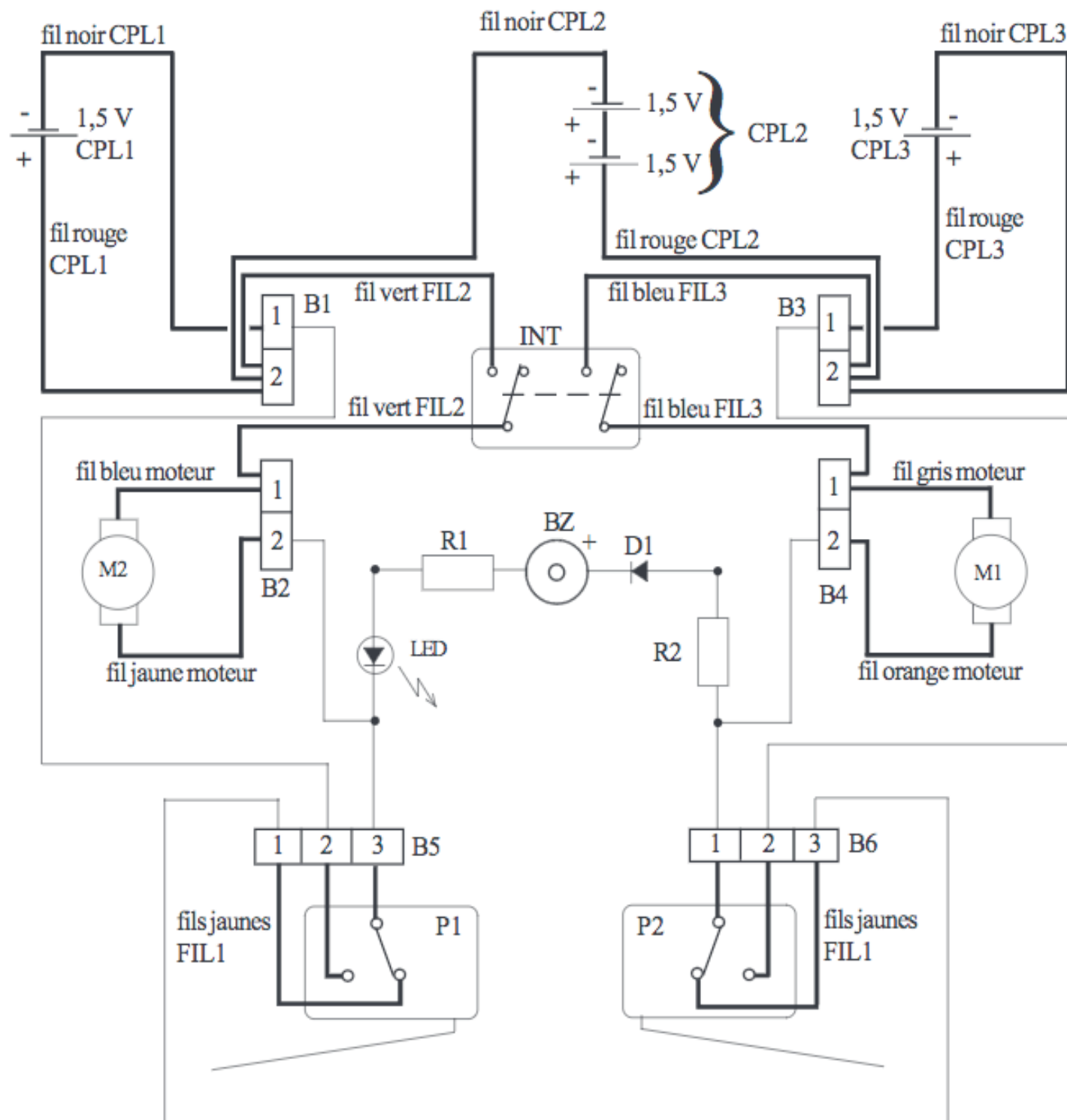
# Maquette



On se sert d'une maquette respectant certaines demandes physiques du cahier des charges comme les dimensions et la résistance.

Robot permettant, sans programme, de contourner des obstacles.

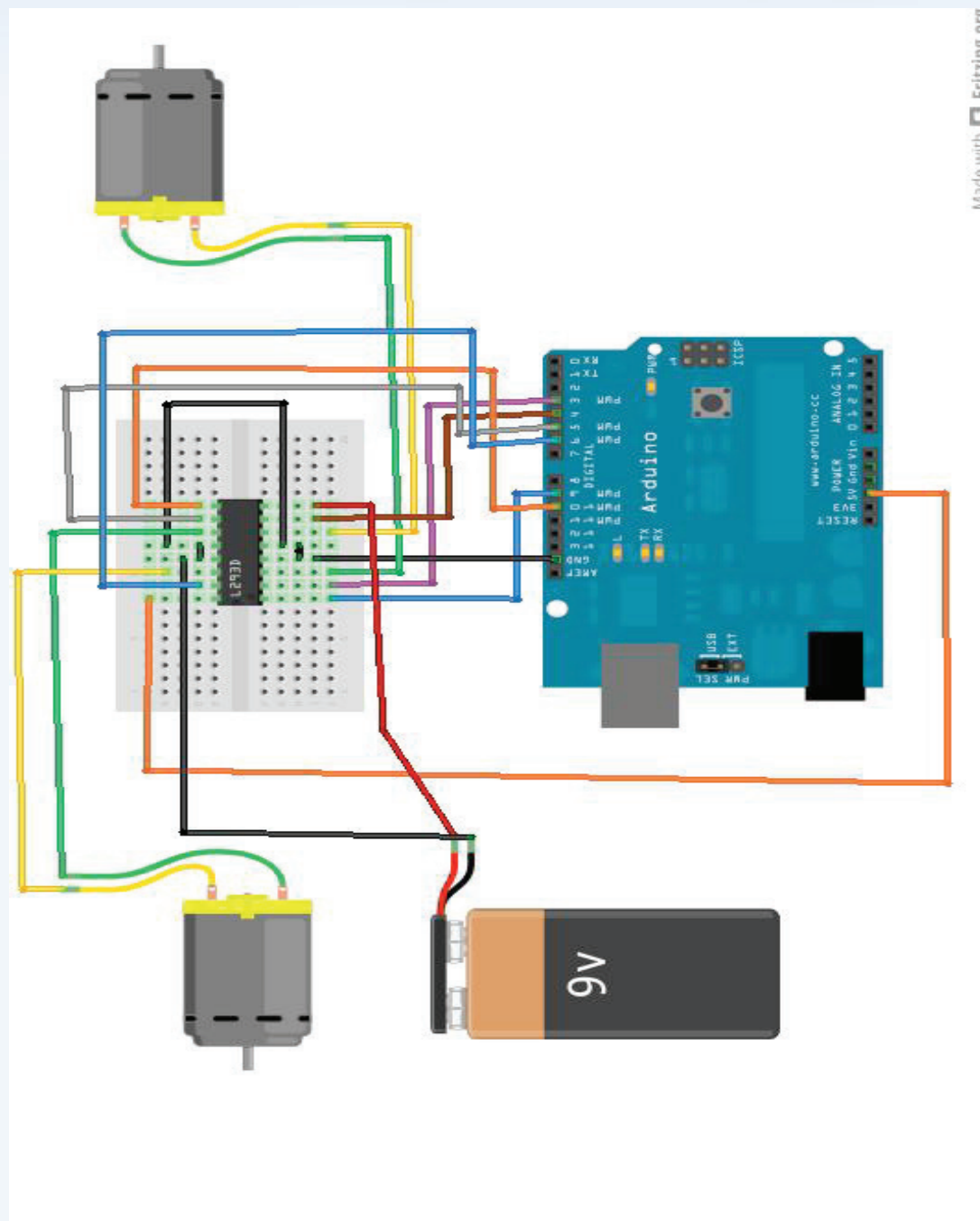




Lorsque l'interrupteur à levier d'une broche est activé, les deux moteurs vont se freiner puis celui opposé tournera alors en marche arrière pendant un temps T, pour qu'enfin les moteurs reprennent leur course.

Ceci est répété jusqu'à ce que le robot se dégage de l'obstacle.

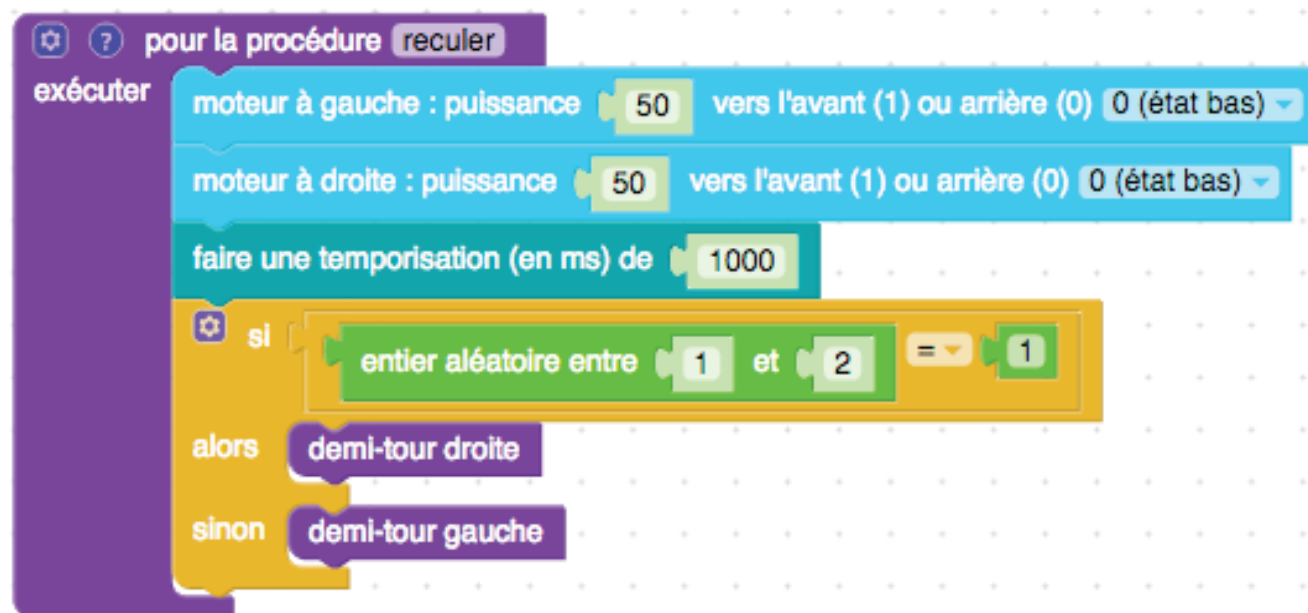
# Programme sous Arduino



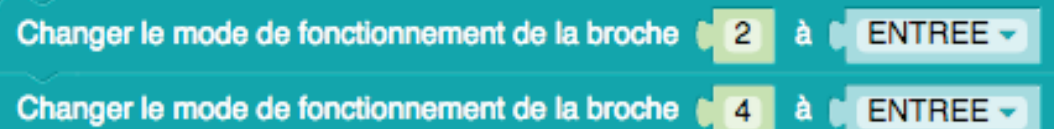
## Utilité de la carte

Permet de piloter chaque moteur indépendamment et plus précisément que la maquette seule.

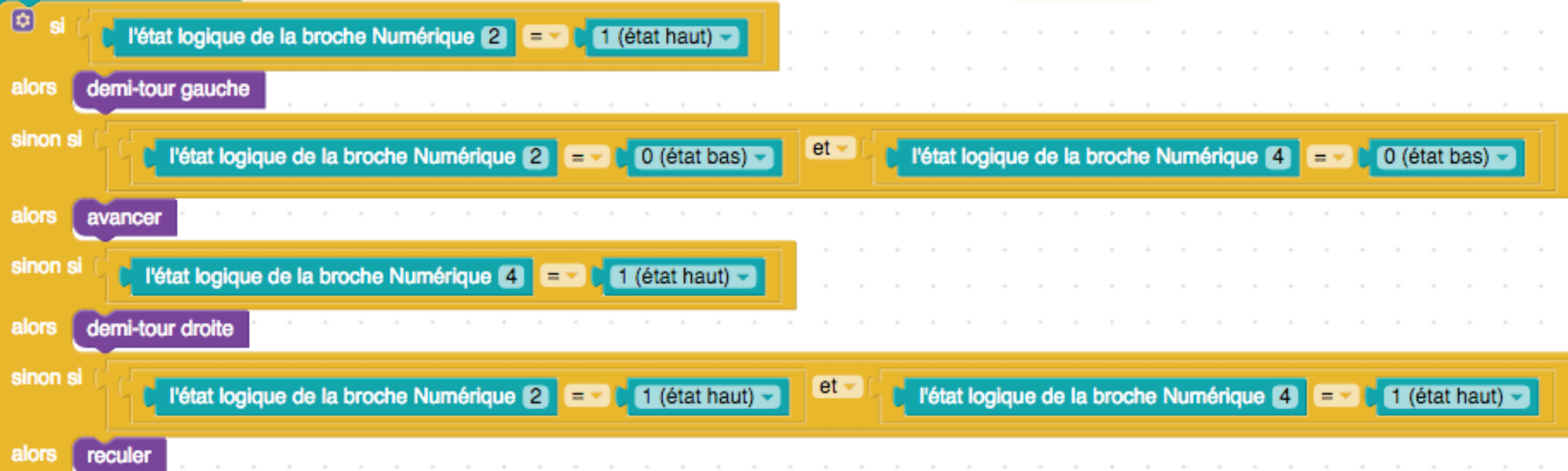
Permet également de choisir la période pendant laquelle le robot tourne sur lui-même.



#### initialisation (setup)



#### répéter indéfiniment (loop)



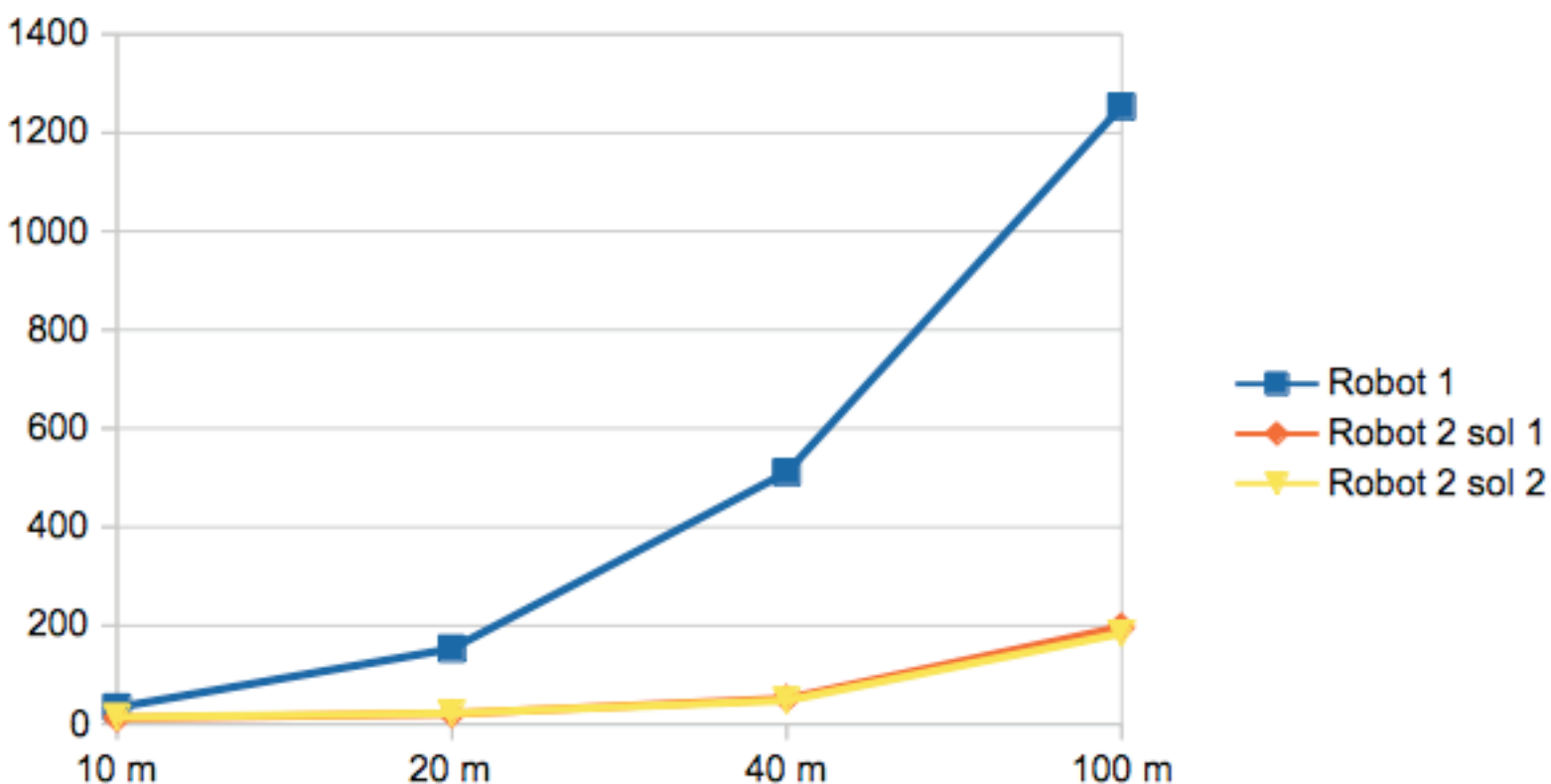
On se sert alors de Blockly@rduino pour coder sans avoir à apprendre le langage de programmation propre à Arduino.

# Vérification

Dimension	10m <sup>2</sup>	20m <sup>2</sup>	40m <sup>2</sup>	100m <sup>2</sup>
Robot 1	24	88	296	1253
Robot 2 1 <sup>o</sup> solution	12	21	52	197
Robot2 2 <sup>o</sup> solution	14	22	48	184

## Simulation

En ayant pris comme dimensions de pièces 10m<sup>2</sup>, 20m<sup>2</sup> et 40m<sup>2</sup>, avec 15% d'obstacles, on peut tracer des courbes montrant l'efficacité des différents programmes et leurs correspondances au cahier des charges.





## Expérimentalement

Dimension	10m2	20m2	40m2	Autonomie
Robot 1	34	< 100	< 100	83
Robot 2 1° solution	17	41	79	71
Robot 2 2° solution	18	36	61	75

On se sert de la maquette et du programme Arduino pour tester le robot dans des conditions réelles d'utilisation.

On vérifie ainsi le temps obtenu par simulation et l'autonomie du robot.

- Autonomie respectée pour tous les robots par piles Ni-MH 1,2V
- Efficacité demandée atteinte seulement sur le robot 2; solution 2
- On observe que plus la pièce grandit, plus le robot 2; solution 2 est efficace comparé aux autres
- Dans tous les cas, le robot 1 n'est pas assez performant



# Conclusion

La simulation et l'expérimentation donnent des résultats dans les mêmes gammes de valeurs.

Le résultat dépend fortement du robot.

Les critères du cahier des charges sont respectés.

Limite de la simulation. N'ont pas été pris en compte :

- *La forme des salles, ni la taille des obstacles.*
- Le rajout d'un algorithme de recherche du plus court chemin.
- *La fonction d'aspiration, qui a été ignorée.*

Pour améliorer la précision de l'étude et les résultats, il faudrait :

- Rajouter un algorithme de recherche du plus court chemin
- Utiliser des capteurs plus évolués, de préférences optiques.
- Créer un algorithme qui fonctionne selon la pièce, au préalable cartographiée.

**Merci pour votre attention.**