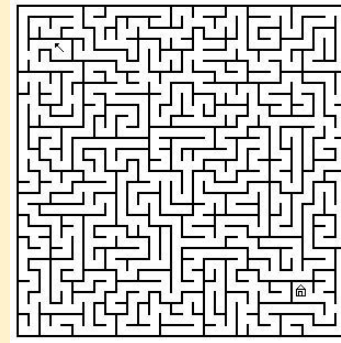


# Assembleur Projet N°5

Résolution d'un labyrinthe



# Sommaire

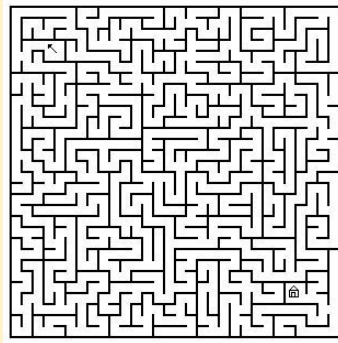
Objectif

Définition et résolution du problème

Implémentation de la solution

# Objectif

Permettre à l'evalBot de sortir d'un labyrinthe par un des chemins les plus court en connaissant de base la configuration de celui-ci

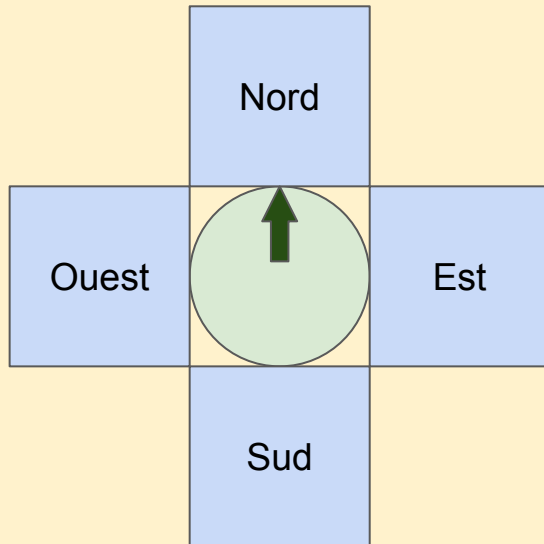


# Définition et résolution du problème

# Notion de direction

Suivant les point cardinaux.

L'EvalBot commence toujours orienté vers le Nord






Direction	Doublé de valeur
NORD	0x00
EST	0x01
SUD	0x10
OUEST	0x11

# Le labyrinthe

Tableau de case dont les valeurs indique l'état de la case.

Le robot connaît sa position grâce à une coordonnées (x,y) passée en mémoire

[illegible]

Couleur	Signification
	Mur
	Robot
	Arrivé

## Exemple de labyrinthe

# Résolution du labyrinthe

**Début**

**Initialisation du robot**

**Modification du plan permettre de trouver le chemin le plus court (1)**

**Tant que le robot n'est pas à la case arrivée (2)**

**Le robot cherche la case adjacente le rapprochant le plus de l'arrivée**

**Le robot pivote dans la direction de la case sur lequel avancer**

**Le robot avance vers la case qui le rapproche de l'arrivée**

**Fin tant que**

**Le robot avance une dernière fois pour sortir du labyrinthe**

**Fin**

# Modification du plan permettre de trouver le chemin le plus court (1)

Valeur de base de la grille

- 0xFF pour un mur
- 0xFE pour une case libre
- 0x00 pour la case d'arrivée

FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
FF	FE	FF	FE	FE	FE	FE	FE	FE	FF
FF	FE	FF	FE	FE	FE	FF	FF	FE	FF
FF	FE	FE	FE	FE	FE	FF	FF	FE	FF
FF	FF	FF	FF	FF	FE	FF	FF	FE	FF
FF	FF	FF	FF	FF	FE	FF	FE	FE	FF
FF	FF	FF	FF	FF	FF	FF	FE	FE	FF
FF	FE	FF	FE	FE	FE	FE	FE	FE	FF
FF	FE	FE	FE	FE	FE	FE	FE	FF	FF
FF	FF	FF	FF	00	FF	FF	FF	FF	FF



# Modification du plan permettre de trouver le chemin le plus court (1)

Toute case libre prend la valeur minimale entre leur valeur et celle de la case voisine de moindre valeur à laquelle on a rajouté 1.

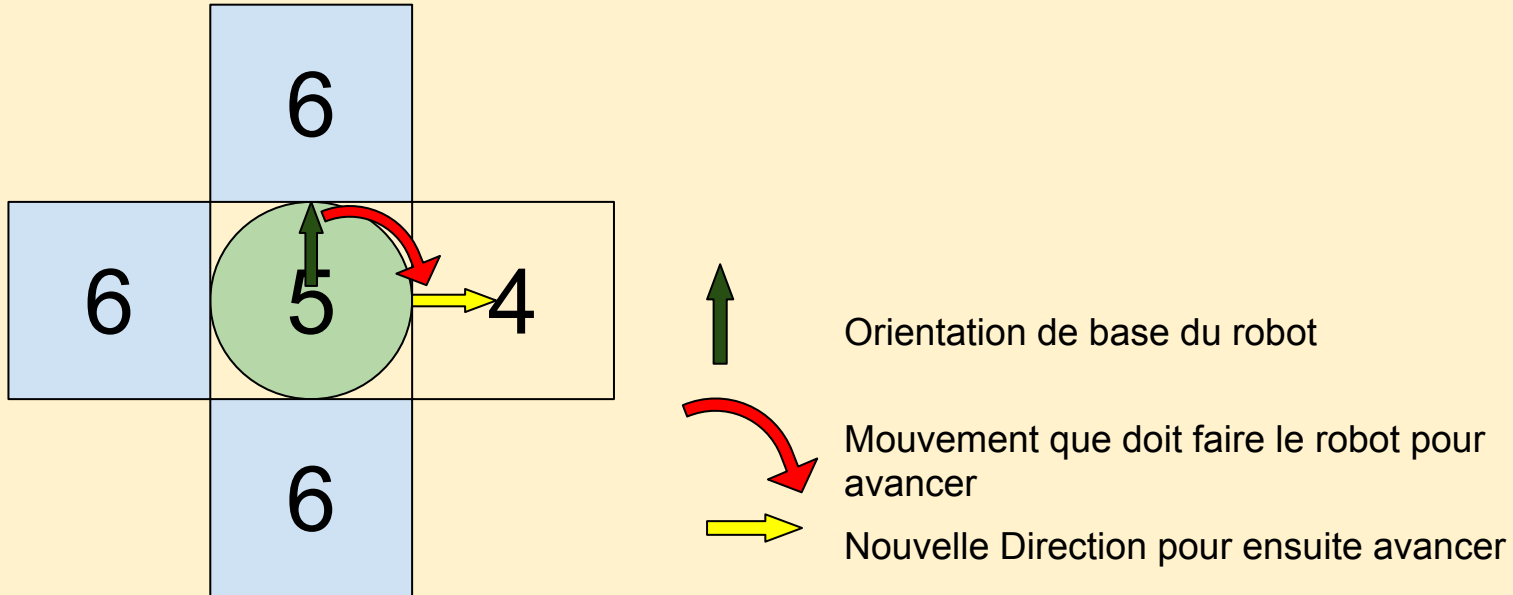
Jusqu'à ce que la grille soit à jour et ne connaisse plus de modification, on réitère le principe.

FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
FF	23	FF	17	16	15	14	13	12	FF
FF	22	FF	18	17	16	FF	FF	11	FF
FF	21	20	19	18	17	FF	FF	10	FF
FF	FF	FF	FF	FF	18	FF	FF	9	FF
FF	FF	FF	FF	FF	19	FF	7	8	FF
FF	FF	FF	FF	FF	FF	FF	6	7	FF
FF	5	FF	3	2	3	4	5	6	FF
FF	4	3	2	1	2	3	4	FF	FF
FF	FF	FF	FF	0	FF	FF	FF	FF	FF

Exemple de chemin possible pour atteindre l'arrivée

# Deplacer le robot (2)

Le robot cherche la première valeur plus petite que lui et ajoute sa direction pour avancer vers elle.



# Implémentation de la solution

# Attribution des registres

Registre	Contenu		
R0	Valeur de la case sur lequel est le robot	R6	Position Y de la case à atteindre par le robot par rapport à la case actuelle (CC2)
R1	Position X du robot	R7	Direction vers lequel le robot doit être orienté pour atteindre la nouvelle case
R2	Position Y du robot	R8 a R11	Buffers
R3	Orientation du robot	R12	Adressage
R4	Valeur de la case que le robot doit atteindre	R13	Stack Pointer
R5	Position X de la case à atteindre par le robot par rapport à la case actuelle (CC2)	R14	Link Register
		R15	Program Counter

# Choix de configuration des GPIO

- Port D activé : PIN 3 de CLK à l'état haut
- USER\_SW1 utilisé :
  - PIN 6 du GPIODEN du Port D pour activer USER\_SW1
  - PIN 6 du GPIOPUR du Port D à l'état haut pour mettre activer résistance de Pull-Up de USER\_SW1

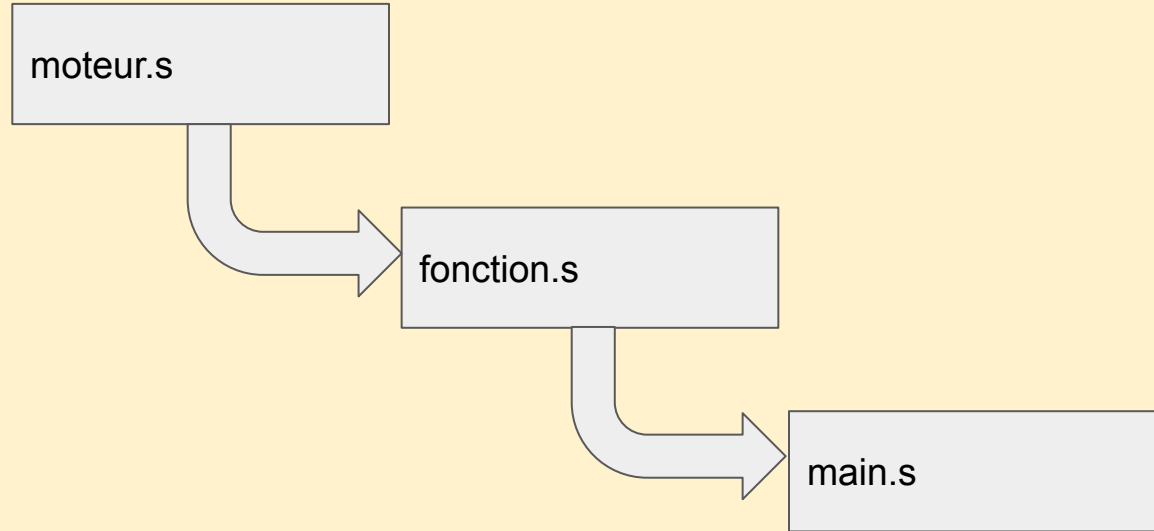
# Implémentation du labyrinthe en python

Transformation d'un tableau de caractère pour être conforme à la notation en assembleur.

Contenu de la case	Valeur initiale	Nouvelle valeur
arrivée	0	0x00
vide	1	0xFE
emplacement du robot	2	0xFE
mur	3	0xFF

X	Y	Largeur	Hauteur
Grille contenant l'ensemble des cases			

# Les fichiers utilisés



# fonction.s

## Fonctions d'initialisation des composants de l'EvalBot

- SW\_INIT  
Permet d'initialiser et d'activer les switchs de l'EvalBot
- INIT  
Permet d'initialiser l'ensemble des fonctionnalités utiles à l'EvalBot pour le programme (regroupant les trois fonctions précédentes et MOTEUR\_INIT du fichier moteur.s)



# fonction.s

## Fonctions de déplacement de l'EvalBot grâce aux moteurs

- MOTEUR\_PIVOTE\_DROITE  
Permet de faire pivoter l'EvalBot de  $90^\circ$  dans le sens horaire.
- MOTEUR\_PIVOTE\_GAUCHE  
Permet de faire pivoter l'EvalBot de  $90^\circ$  dans le sens anti-horaire.
- MOTEUR\_DEMI\_TOUR  
Permet de faire pivoter l'EvalBot de  $180^\circ$  dans le sens horaire.
- MOTEUR\_AVANCE  
Permet de faire avancer l'EvalBot d'une case en ligne droite

# fonction.s

## Fonctions utilisé dans le main pour résoudre le problème du labyrinthe

- CHOIX\_DIRECTION  
Permet de déterminer dans quelle direction doit se diriger l'EvalBot pour se rapprocher de l'arrivée.
- SOLVE\_MAZE  
A partir du plan du labyrinthe en mémoire, la fonction le modifie pour déterminer le chemin le plus court (voir "Chercher le plus court chemin (1)")

# Fonctionnement du main

Début

INIT

SOLVE\_MAZE

Tant que le robot n'est pas à la case arrivée

CHOIX\_DIRECTION

MOTEUR\_PIVOTE\_DROITE ou MOTEUR\_PIVOTE\_GAUCHE ou MOTEUR\_DEMI\_TOUR ou RIEN

MOTEUR\_AVANCE

Fin tant que

MOTEUR\_AVANCE

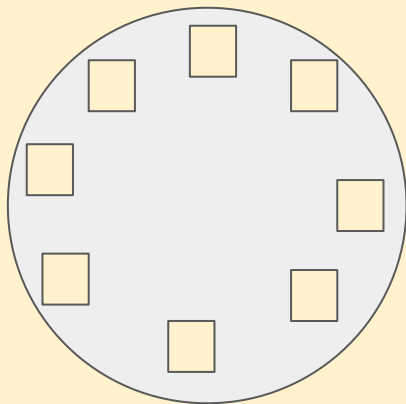
Fin

# Proposition d'amélioration

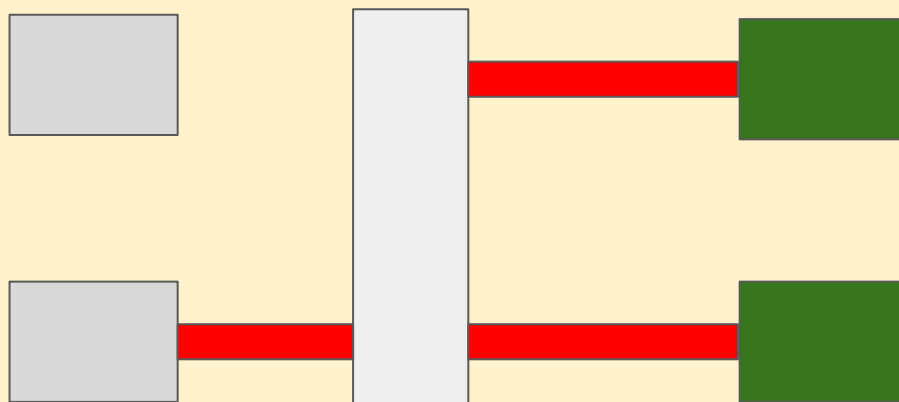
Solution possible:

On compte le nombre de passage de l'infrarouge à travers la roue des 2 côtés. Pour avancer en ligne droite, les 2 roues doivent avoir effectué 11 captations.

Pour tourner, 7 captations sont nécessaires



Roue vue du côté



Roue vue du dessus