

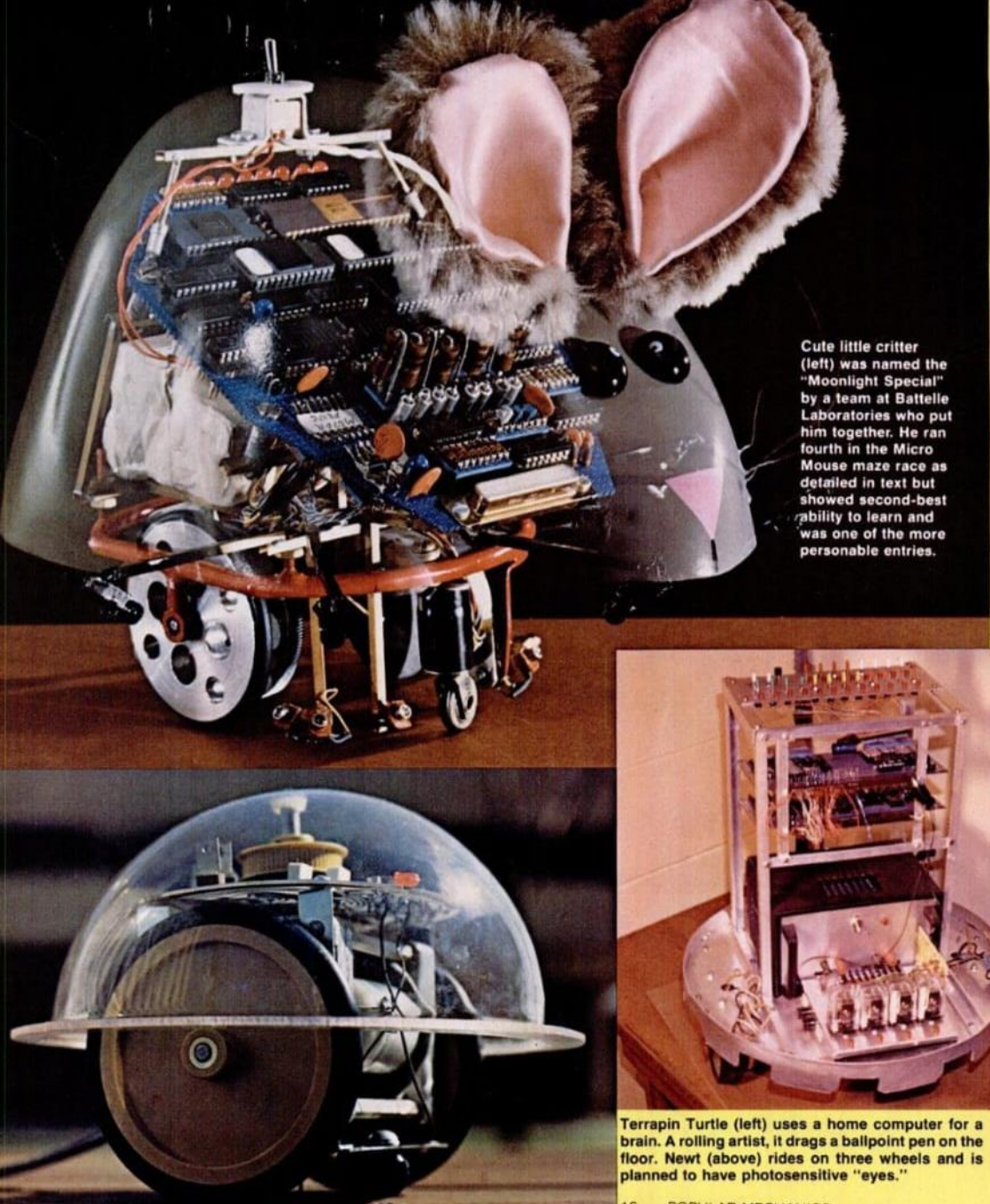
Conquering

Mountain

Un Examen des Choix Algorithmiques et Matériels pour un Temps Record

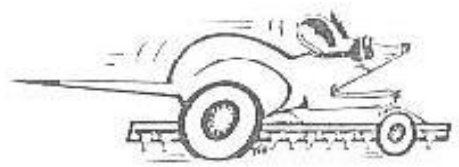
Marouane BERRAD – TA069M

général



1. C'est quoi un robot Micromouse ?
2. Histoire/règlement de la compétition.
3. Étude algorithmique:
 - La création des labyrinthes de test.
 - La recherche du meilleur algorithme de résolution.
4. Étude matérielle (les améliorations matérielles possibles).
5. Simulation de l'évitement des murs à l'aide d'un robot Arduino.

FIGURE 1 : Magazine « Popular mechanics » août 1980 p.16



MICROMOUSE À
Paris

7-10 septembre 1981
EUROMICRO 81

Le robot micromouse

Un robot Micromouse est un petit robot autonome conçu pour naviguer et trouver son chemin à travers un labyrinthe.

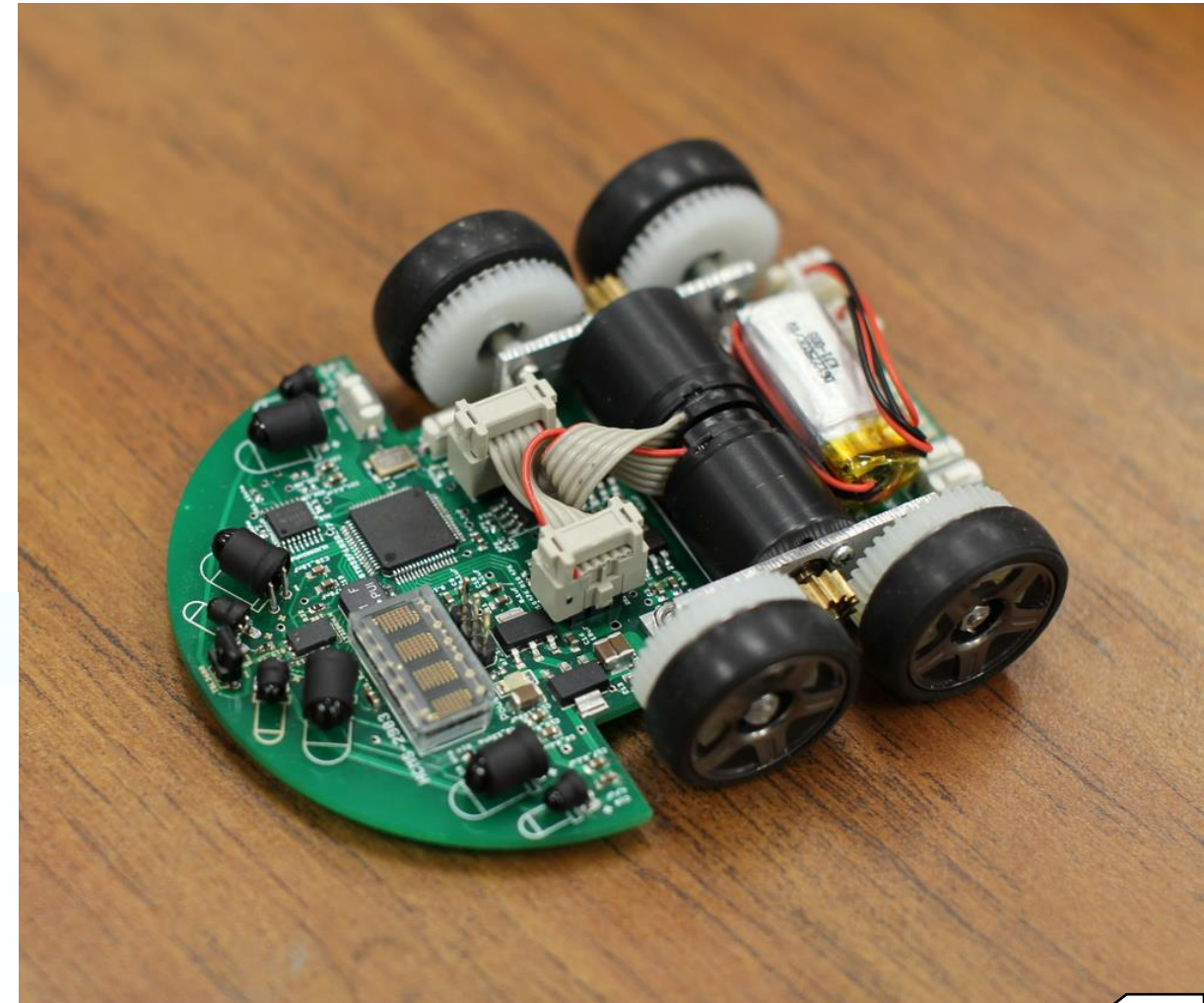
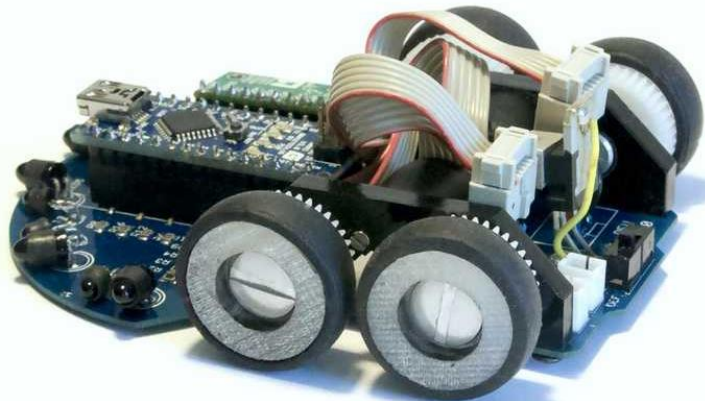


FIGURE 2 : Deux robots Micromouse

chronologie

Annonce de la compétition
« Amazing Micromouse
Competition » par IEEE Spectrum

1977

le changement du but final
au centre du labyrinthe

1980

1979

Réalisation de la première
édition de la compétition

Introduire la catégorie
“half size” 32x32

2009

1985

Première compétition
internationale

2016

Le record mondial actuel
de 3,921 secondes détenu
par “Ng Beng Kiat”

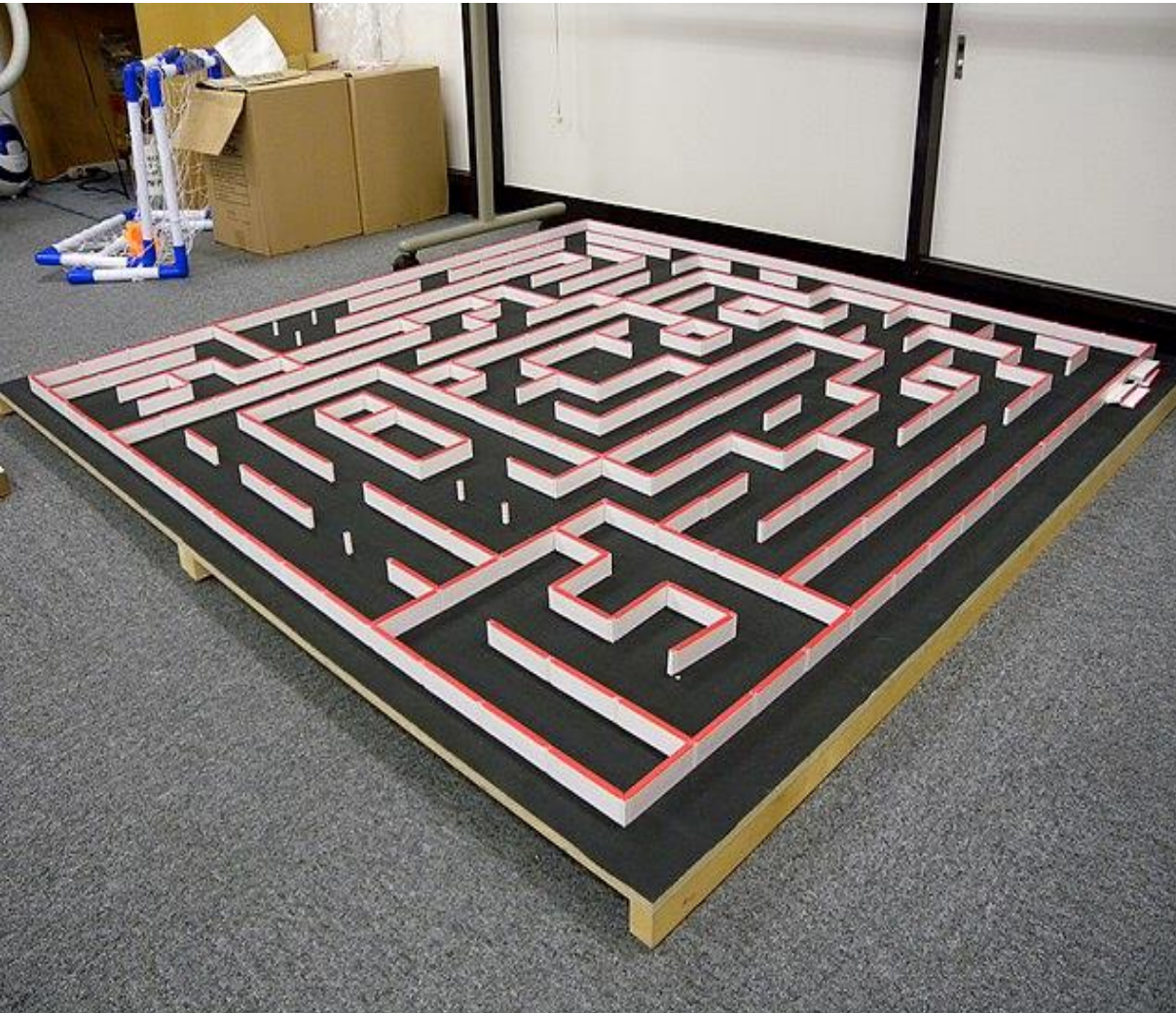
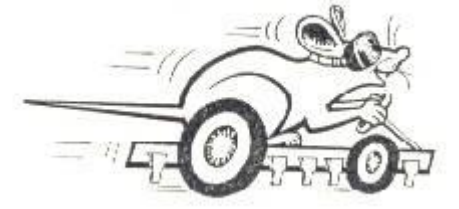
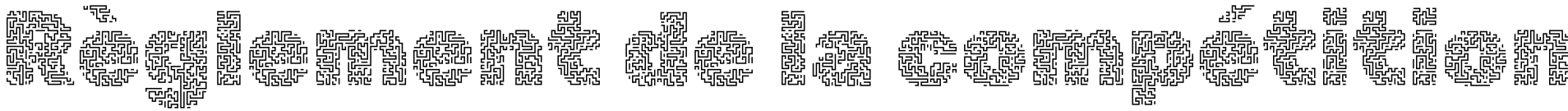


FIGURE 3 : Labyrinthe 16 x 16

1. Autonomie et Alimentation :

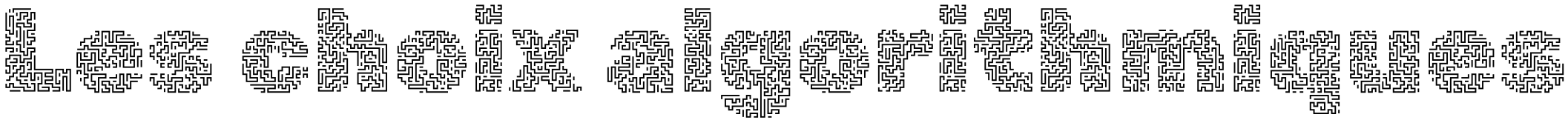
- Les Micromouses doivent être autonomes, sans utilisation de télécommande.
- Les sources d'énergie à combustion sont interdites.

2. Déplacement :

- Ils ne sont pas autorisés à sauter, à voler ou à endommager les murs du labyrinthe..

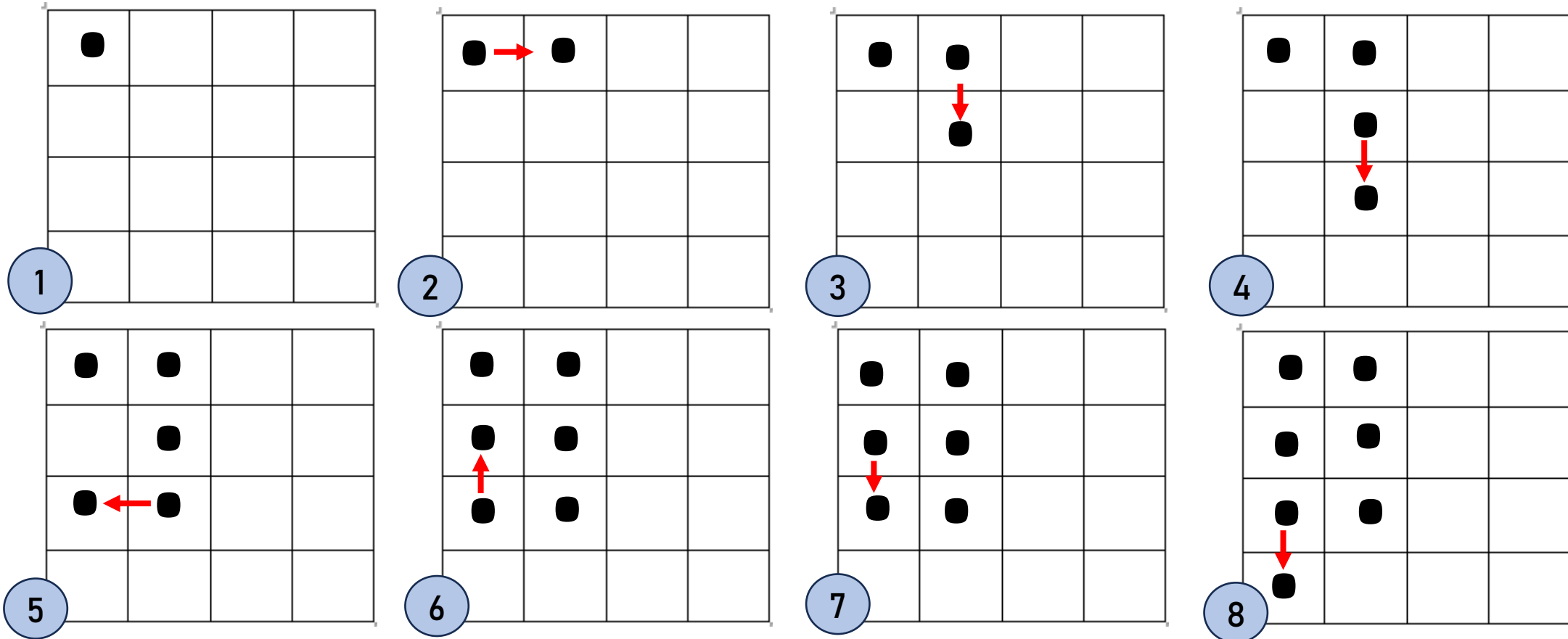
3. Temps :

- Chaque Micromouse a 10 minutes pour naviguer dans le labyrinthe.
- Le temps de chaque course est enregistré, le plus court étant retenu comme temps officiel.



Génération de labyrinthes aléatoires pour les tests

- Pour expliquer le fonctionnement de l'algorithme de génération, on va prendre l'exemple d'une labyrinthe de 4 x 4:



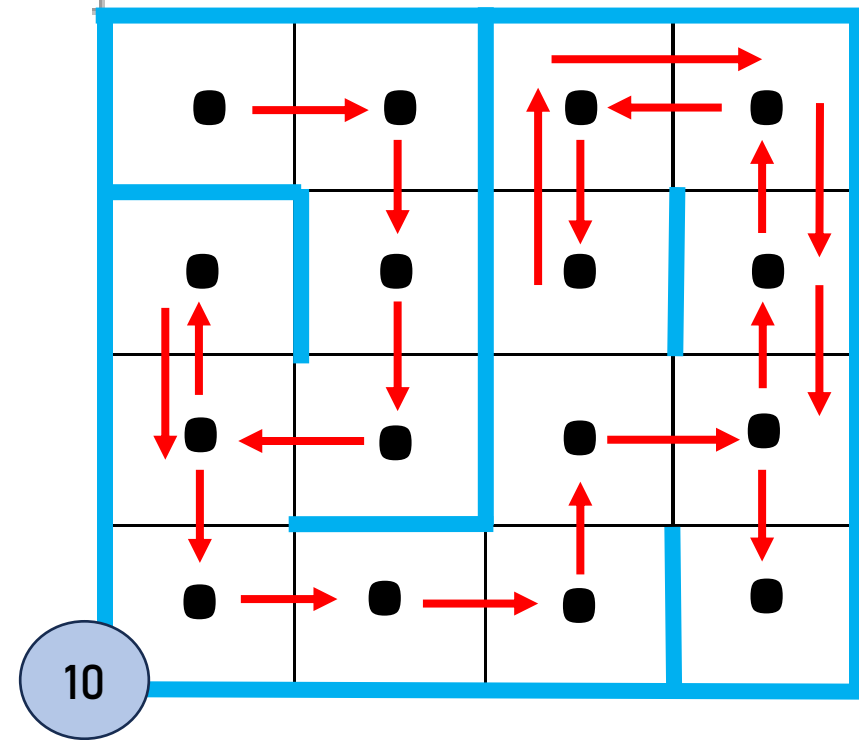
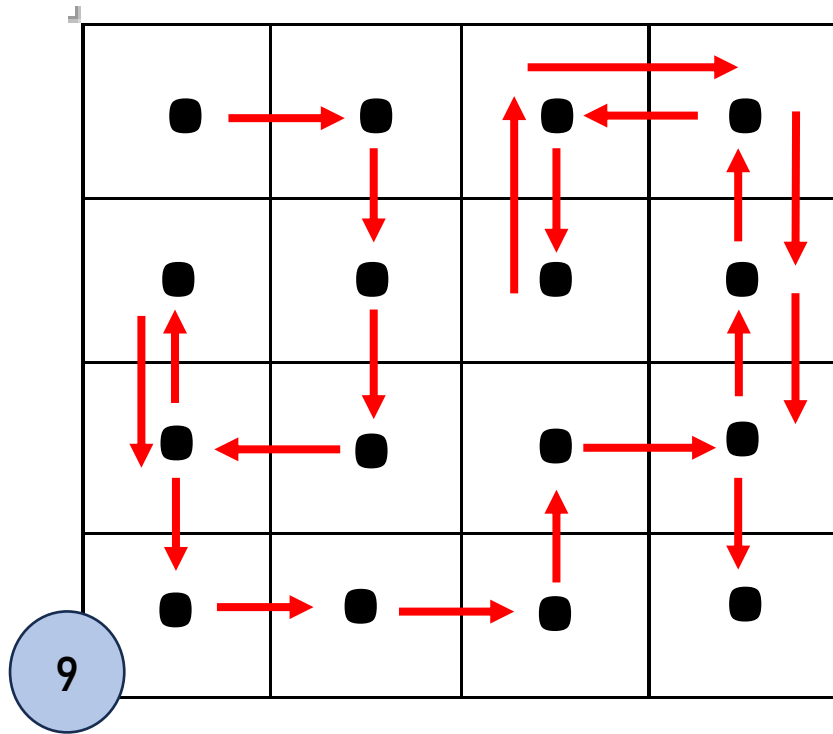


FIGURE 4 : Exemple explicite du labyrinthe 4x4.

- L'implémentation de cet algorithme en Python est basée sur les piles.

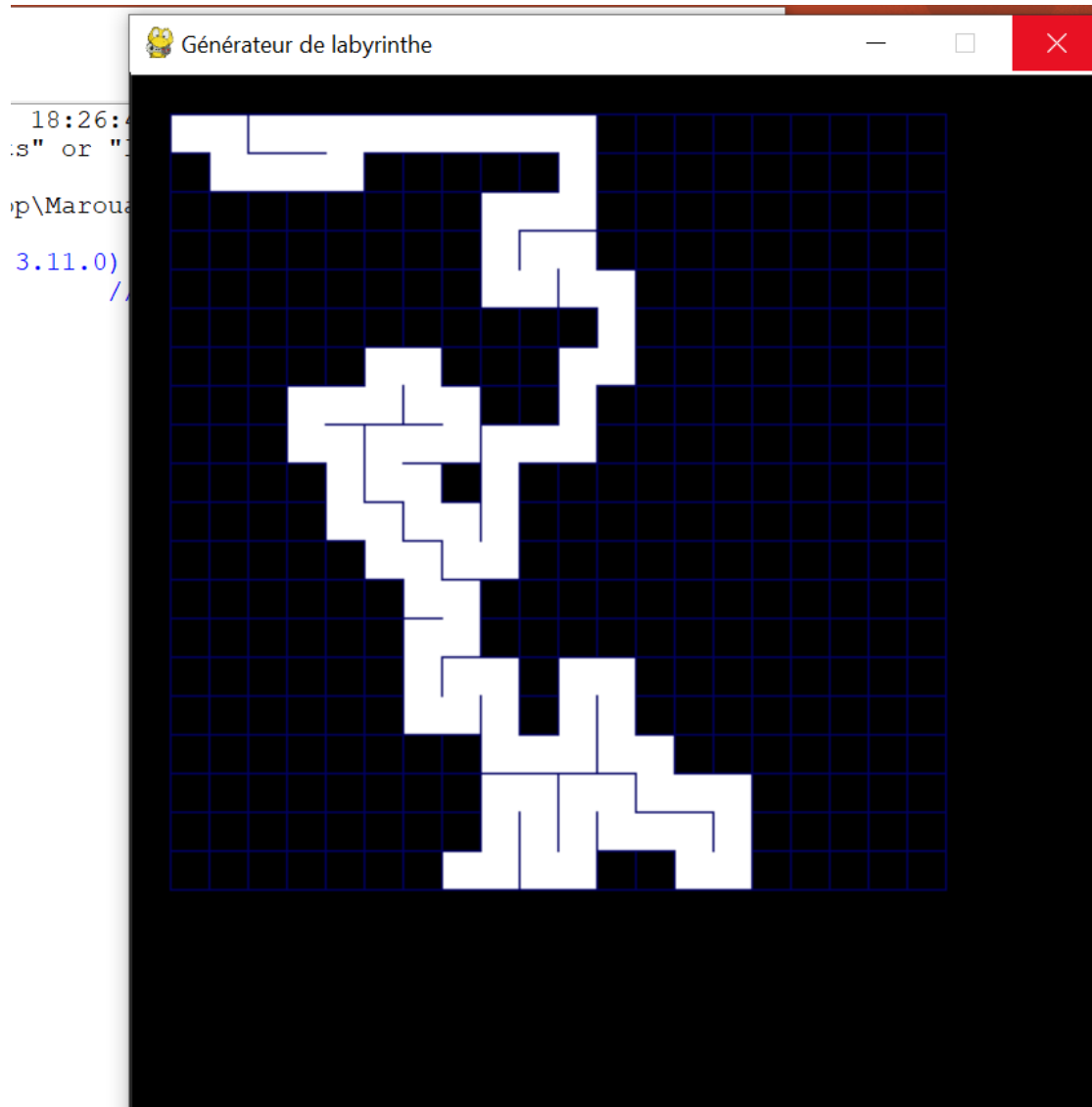


FIGURE 5 : Au cours de la création d'un labyrinthe 20x20

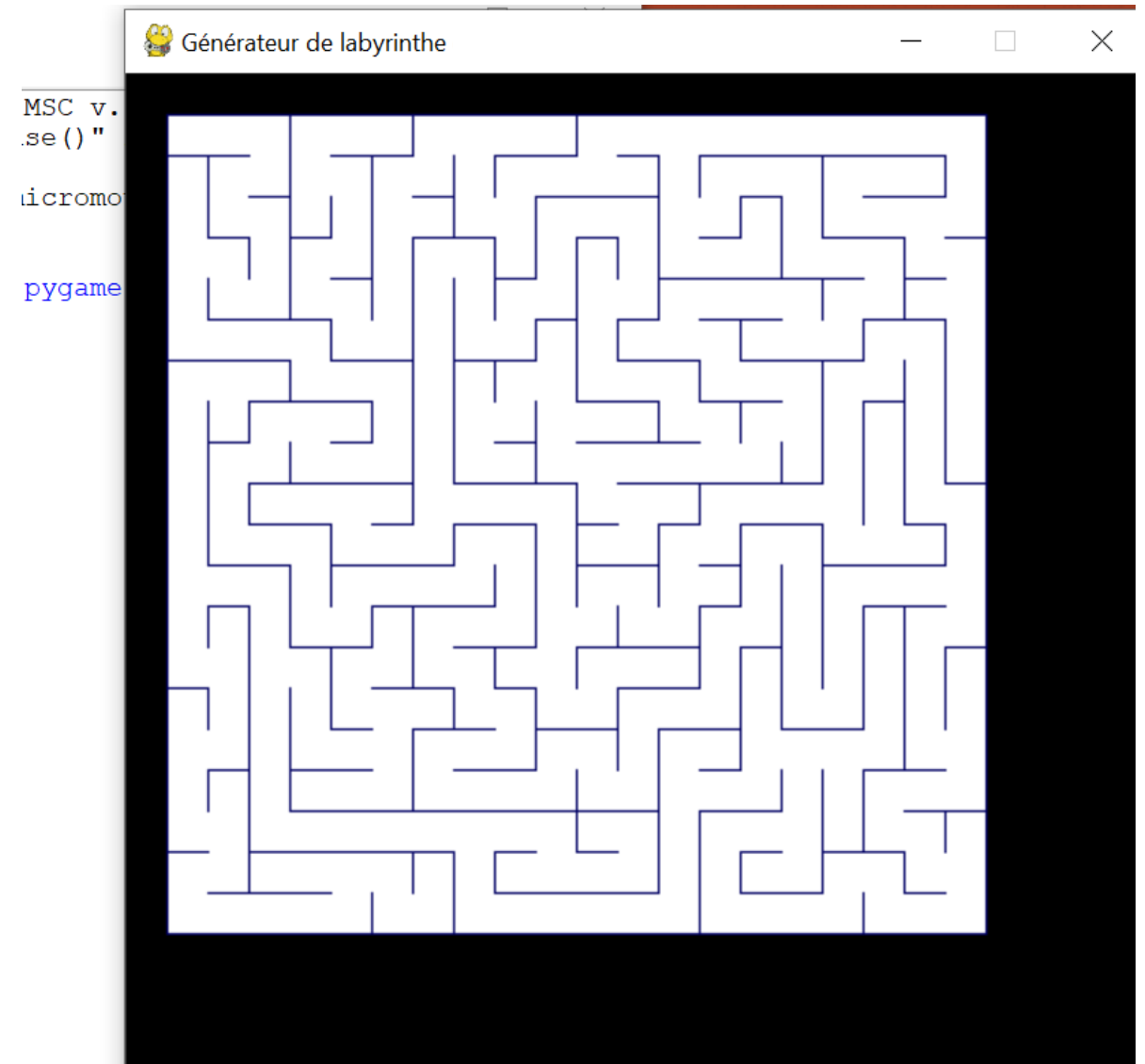


FIGURE 6 : Le résultat final généré par l'algorithme

Le labyrinthe de la compétition de Hyvinkö

Le suiveur de mur (logique de la main droite) :

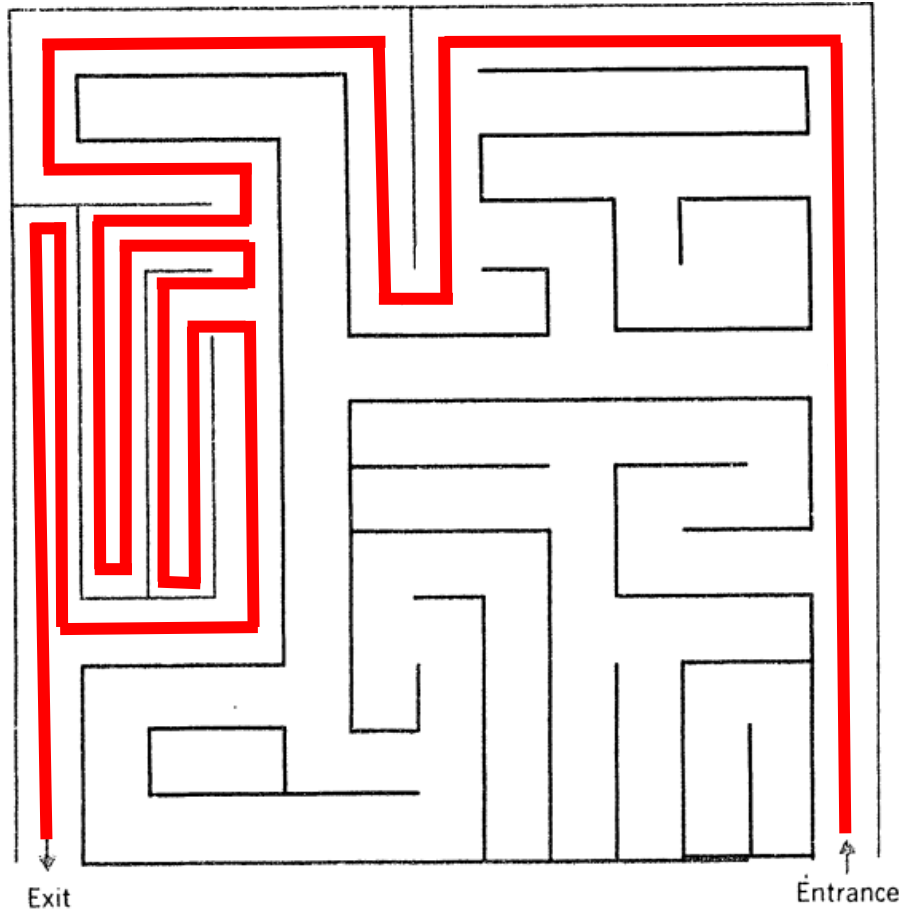


FIGURE 7 : Le labyrinthe officiel de la première édition de la compétition en 1979.

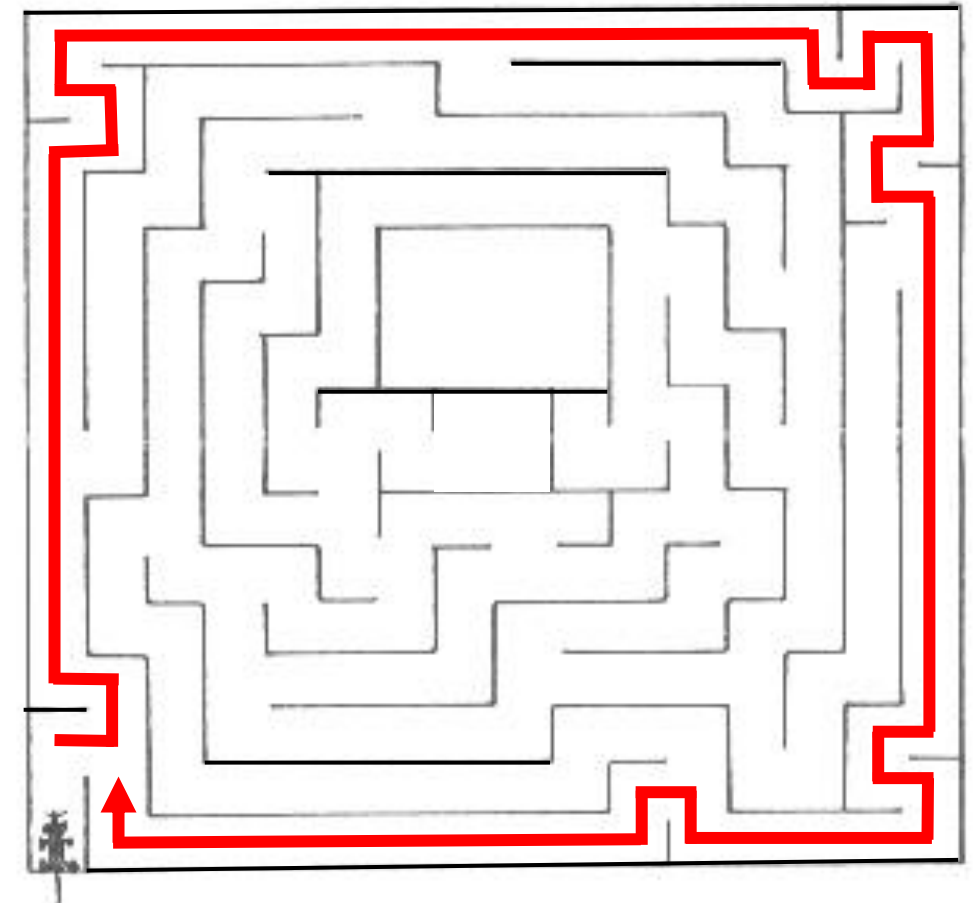
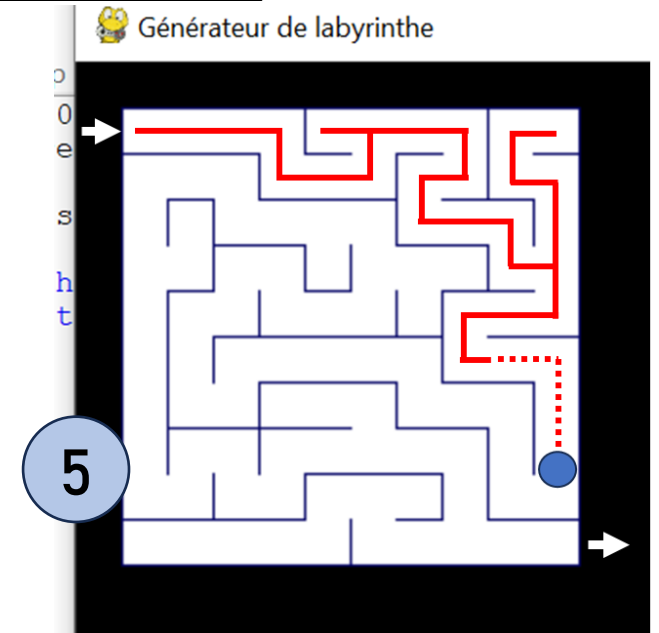
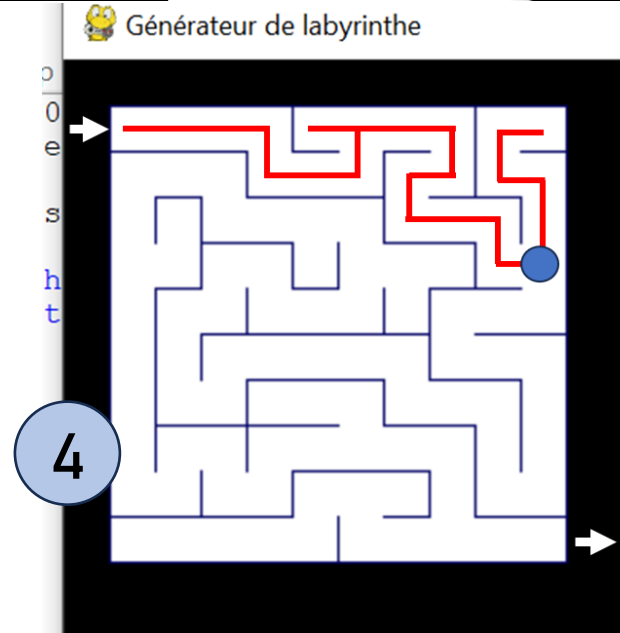
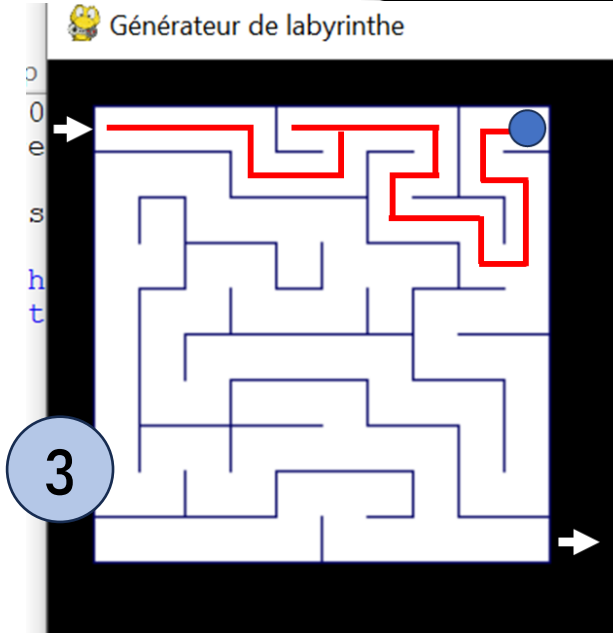
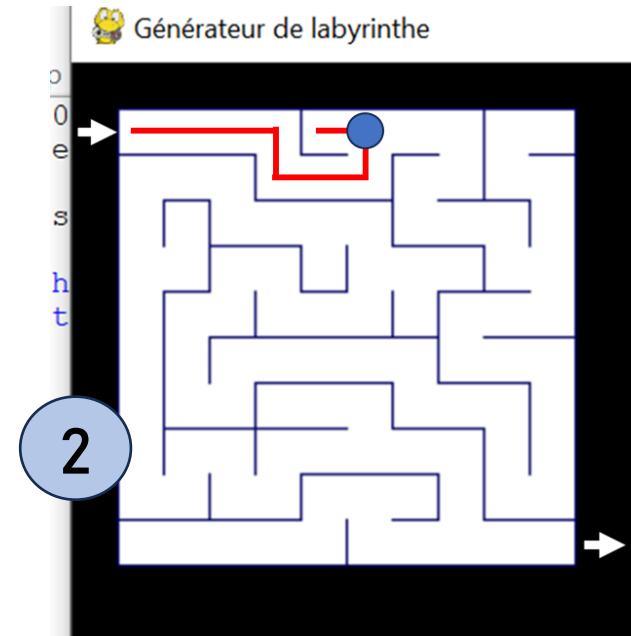
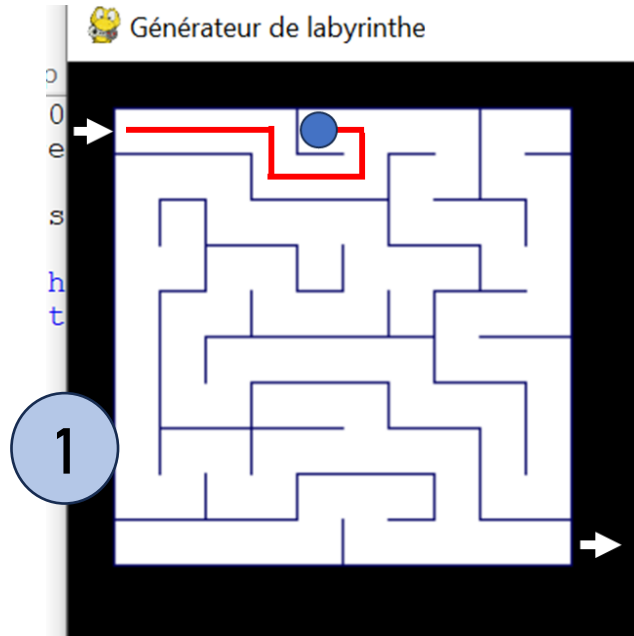


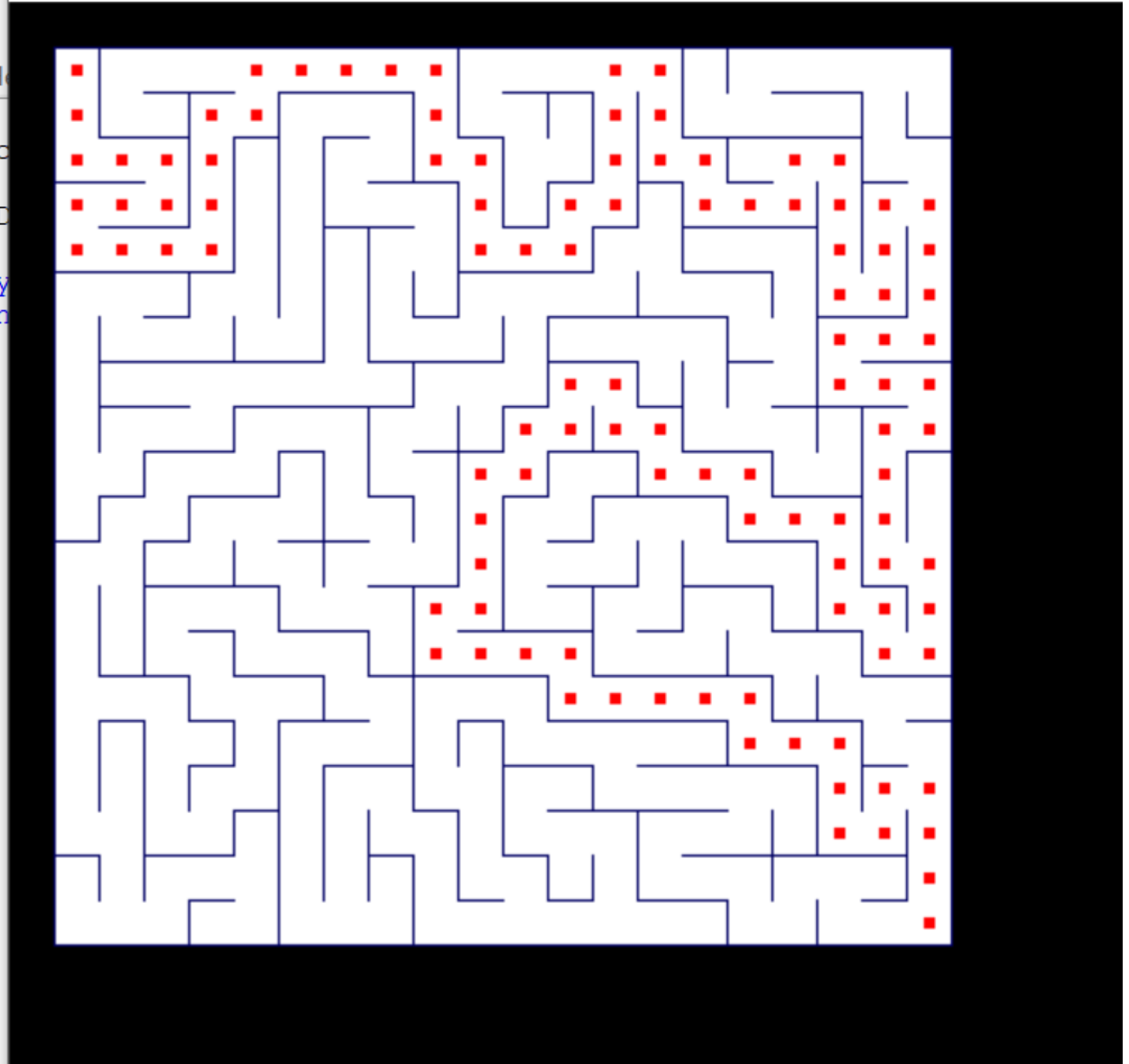
FIGURE 8 : Le labyrinthe officiel de la deuxième édition de la compétition en 1980.

DFS(Depth first search)



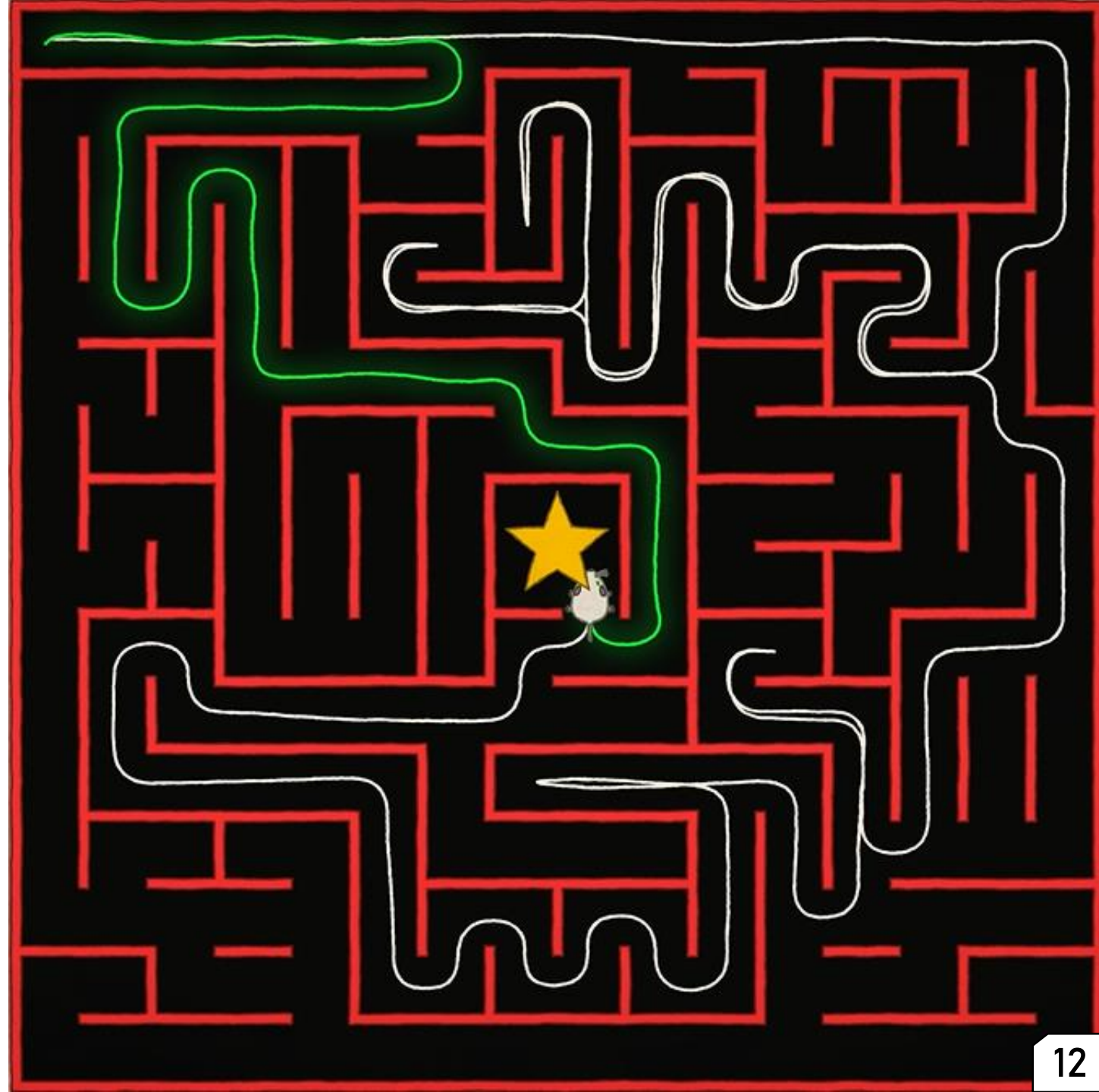
ow H
: 24
, "o
ent/D
3, Py
mmun

FIGURE 9 : Le résultat du 2^{ème} version du code python basé sur DFS et générant un labyrinthe, ainsi que sa solution.



Le chemin blanc représente la solution trouvée par l'algorithme DFS. Donc, cet algorithme garantit une solution de labyrinthe, mais il n'assure pas qu'elle soit la plus courte possible.

Figure 10 : une simulation réalisé par Derek Muller(*) de la résolution de labyrinthe par l'algorithme DFS.



L'algorithme « Flood Fill »

Expliquant le fonctionnement de cet algorithme dans le cas d'un labyrinthe de 5 x 5 :



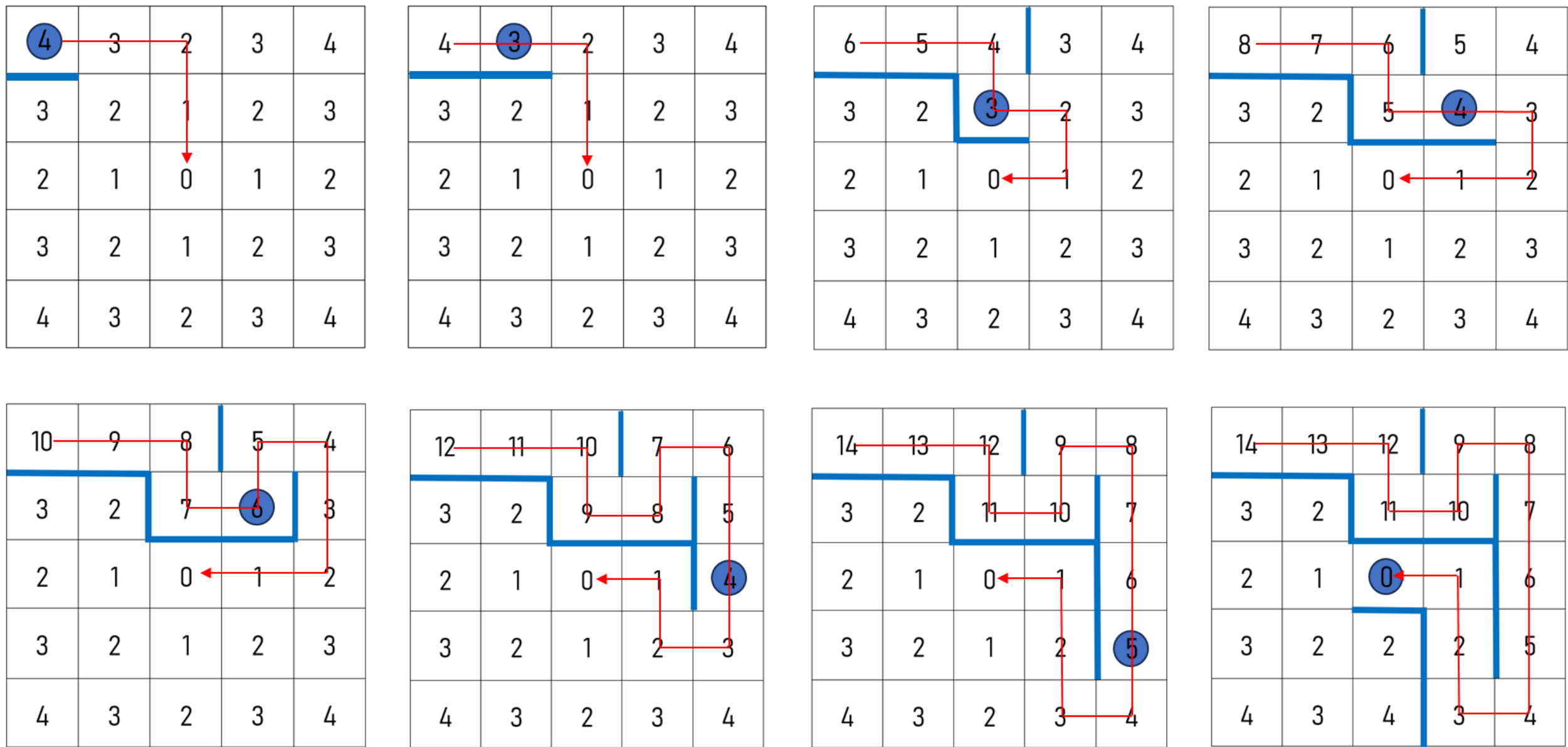
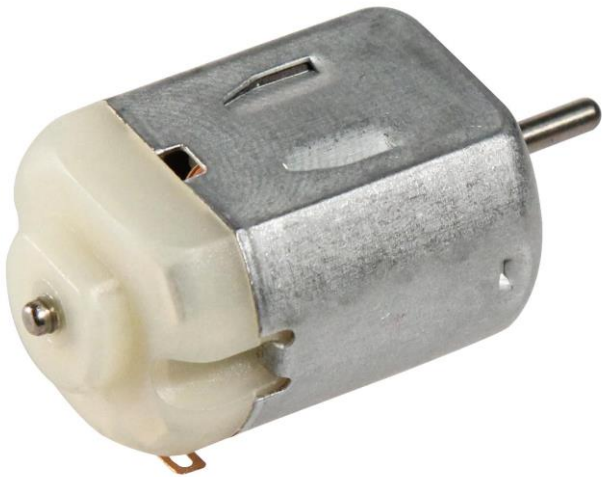


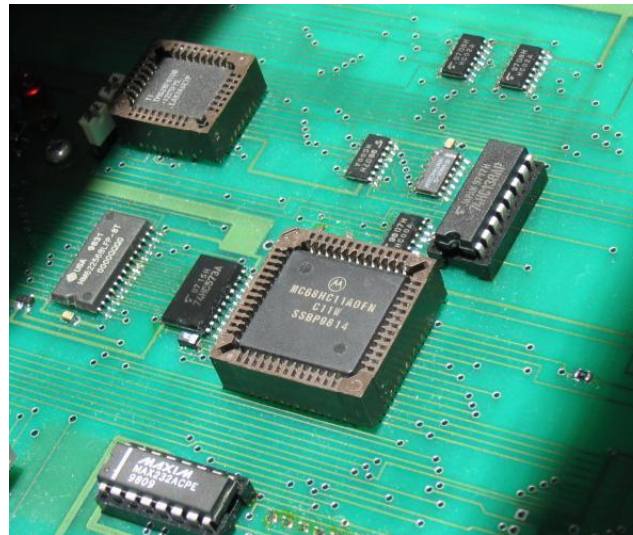
FIGURE 11 : Les étapes explicatives de l'algorithme Flood Fill appliqué à un labyrinthe de 5 x 5

Les composants matériels

Les composants matériels indispensables :



Moteurs



Microcontrôleurs



Capteurs

Limite de la pression d'aspiration

avec robot Micromouse

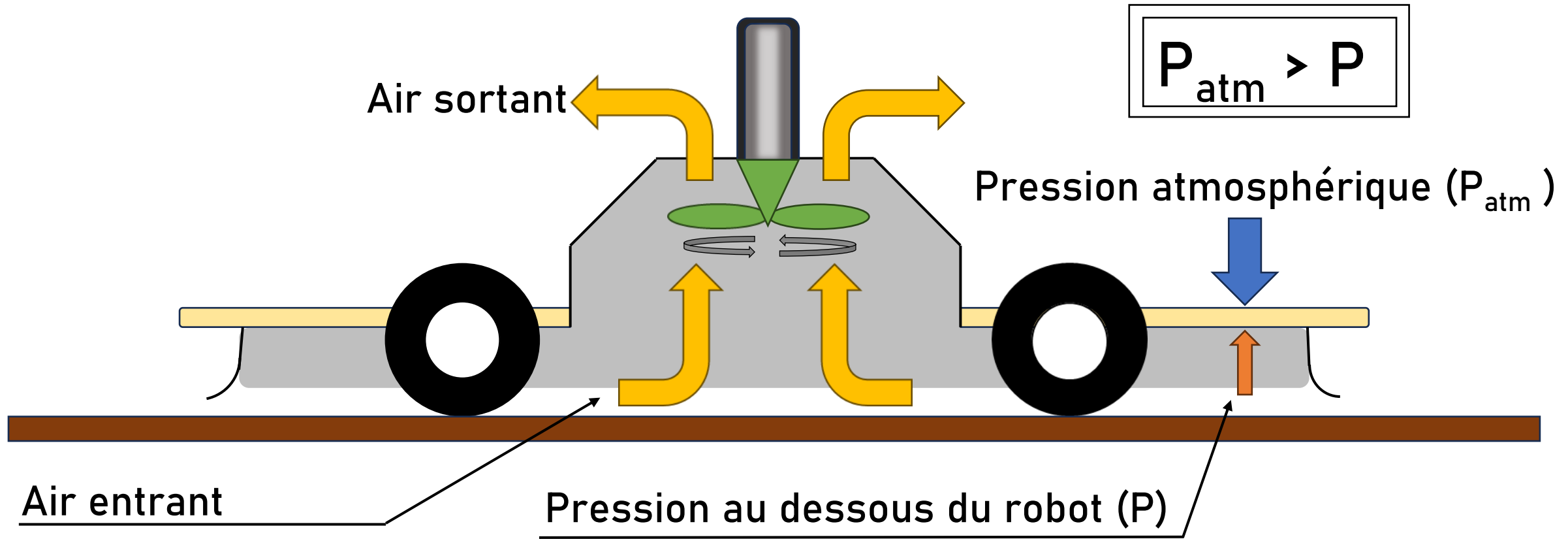


FIGURE 12 : Schéma du système d'aspiration du robot Micromouse

- Système étudié: { Le robot Micromouse }

- Bilan des forces:
 - Le poids : \vec{P}
 - La réaction du sol sur la $i^{\text{ème}}$ roue : $\vec{R}_i = \vec{T}_i + \vec{N}_i$
 - La force de pression atmosphérique : \vec{F}_{atm}
 - La force de pression au dessous du robot : \vec{F}

- On pose : $\vec{R} = \vec{R}_1 + \vec{R}_2 + \vec{R}_3 + \vec{R}_4$, G le centre de gravité du robot, S la surface supérieure et inférieure moyenne du robot et m la masse du robot.
 et $\vec{R} = \vec{T} + \vec{N}$

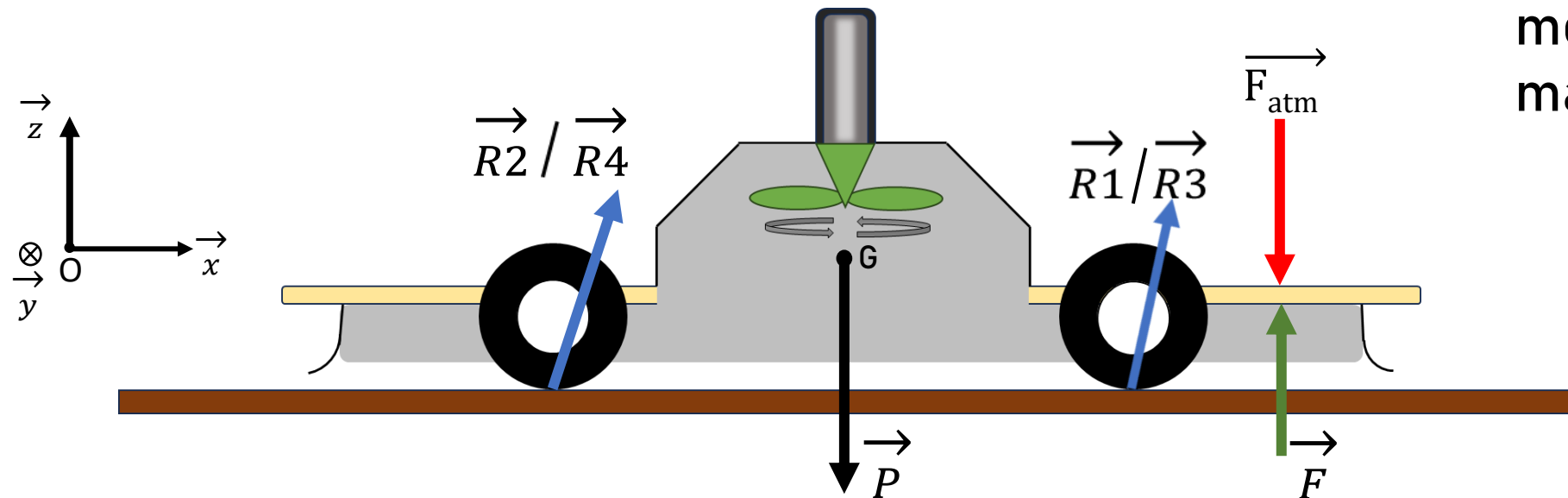


FIGURE 13 : La représentation des forces appliquées au robot sans souci d'échelle.

- En réalité, les deux roues de chaque côté sont très rapprochées. Parfois, il y a seulement deux roues motrices avec une roue libre (roue de castor). De plus, lors de la prise d'un virage, il n'y a pas de mouvement de pivot de la roue avant ; la rotation du véhicule est assurée par la différence de vitesse entre la roue gauche et la roue droite.

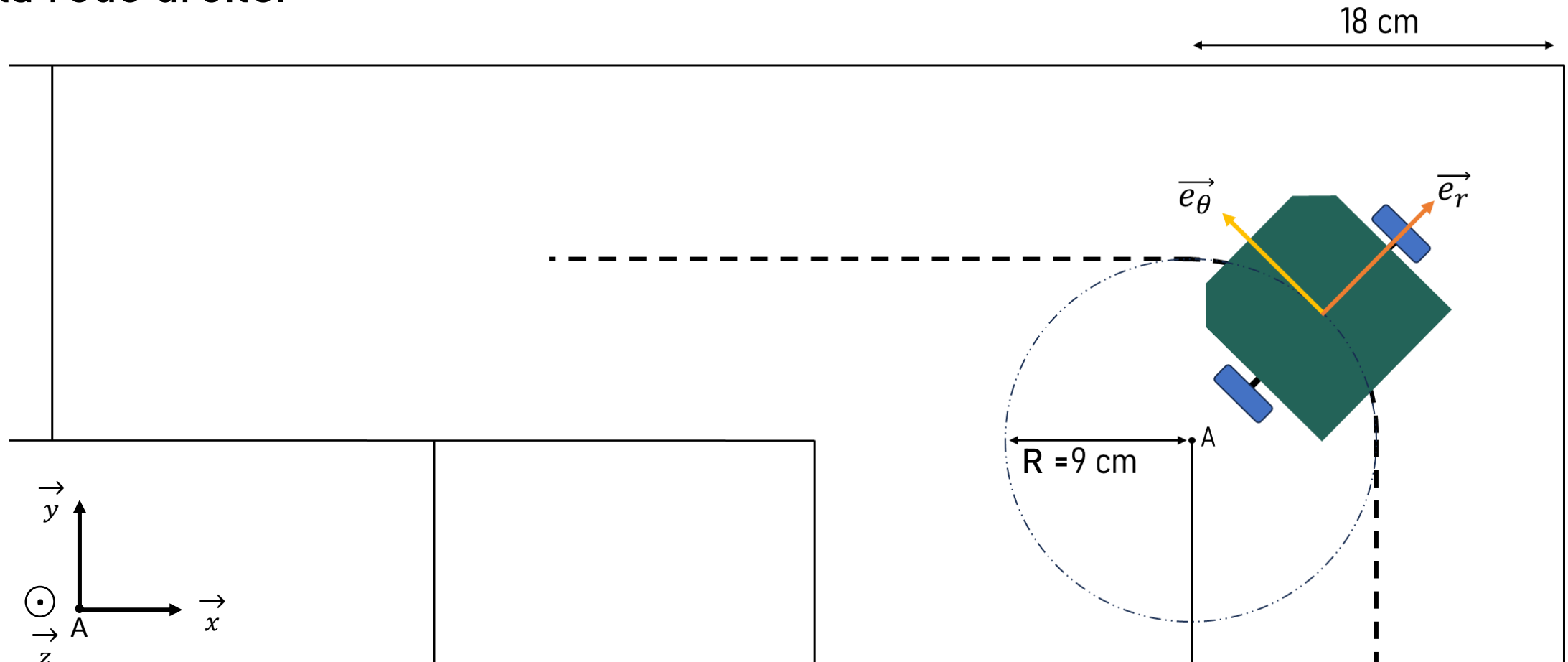


FIGURE 14 : Schéma du robot prenant un virage

- On applique le principe fondamental de la dynamique sur le robot Micromouse dans le référentiel $(0, \vec{x}, \vec{y}, \vec{z})$ supposé galiléen :

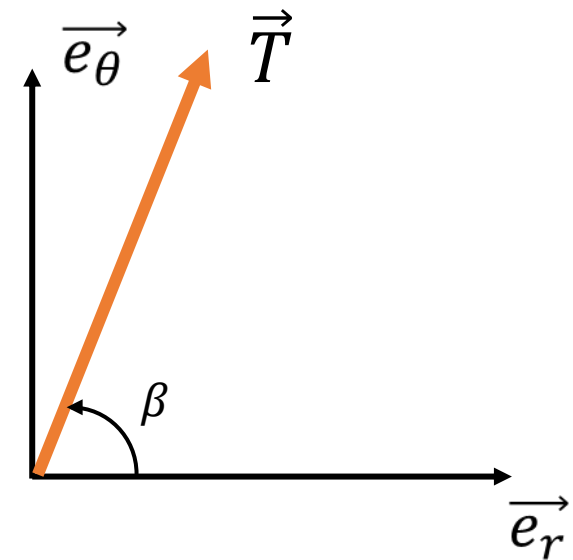
$$m\vec{a} = \vec{P} + \vec{R} + \vec{F} + \vec{F}_{atm}$$

- En projetant suivant les axes de la base cylindrique $(A, \vec{e}_\theta, \vec{e}_r, \vec{z})$ on obtient :

$$\begin{cases} m \frac{V^2}{R} = -T \cos \beta \\ \dot{V} = T \sin \beta \\ 0 = -mg + N + S \times P - S \times P_{atm} \end{cases}$$

- On conserve les deux équations :

$$(1) \quad \begin{cases} V^2 = -\frac{TR \cos \beta}{m} \\ N = mg + S(P_{atm} - P) \end{cases}$$



- En plus, d'après la loi de coulomb (dans le cas de roulement sans glissement) :

$$\boxed{\|\vec{T}\| < \mu_s \|\vec{N}\|} \quad (2) \quad \text{donc on peut prendre } \mu_s = 0,9$$

De (1) et (2), on déduit que :

$$V^2 < \mu_s (mg + S(P_{atm} - P)) \frac{R}{m} |\cos \beta|$$

Donc,

$$V < \sqrt[2]{\mu_s (mg + S(P_{atm} - P)) \frac{R}{m}}$$

D'où,

$$\boxed{V_{max(asp\acute{e}ration)} = \sqrt[2]{\mu_s (mg + S(P_{atm} - P)) \frac{R}{m}}}$$

- Or, dans le cas normal, sans existence du système d'aspiration, la vitesse maximale pour prendre un virage sera :

$$V_{max} = \sqrt[2]{\mu_s g R}$$

Donc ,

$$V_{max} < V_{max(asp\acute{e}ration)}$$

- Donc, l'ajout du système d'aspiration au robot augmente la vitesse maximale de rotation, ce qui permet de gagner du temps et de minimiser le temps final de la résolution du labyrinthe.

Synthèse de l'écriture de code pour le robot Arduino

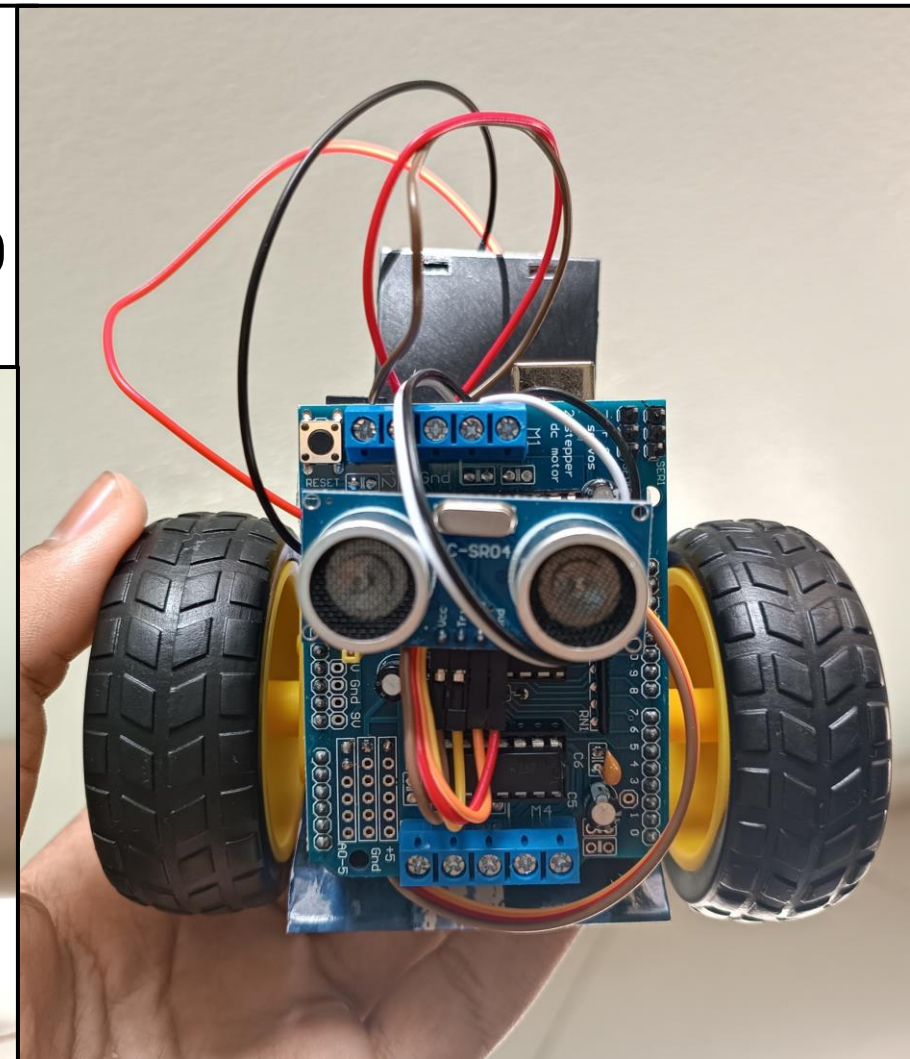
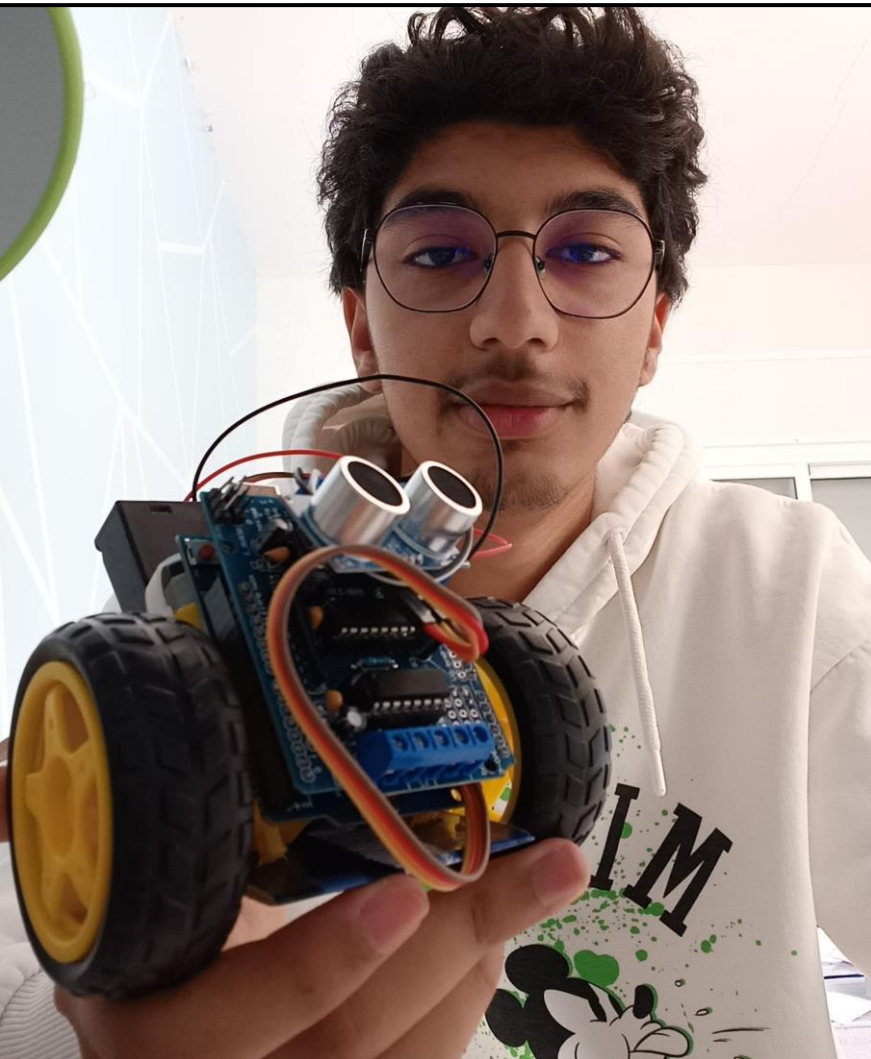


FIGURE 15 : Images du robot Arduino prise sous différents angles

- Les différentes conceptions de ce robot :

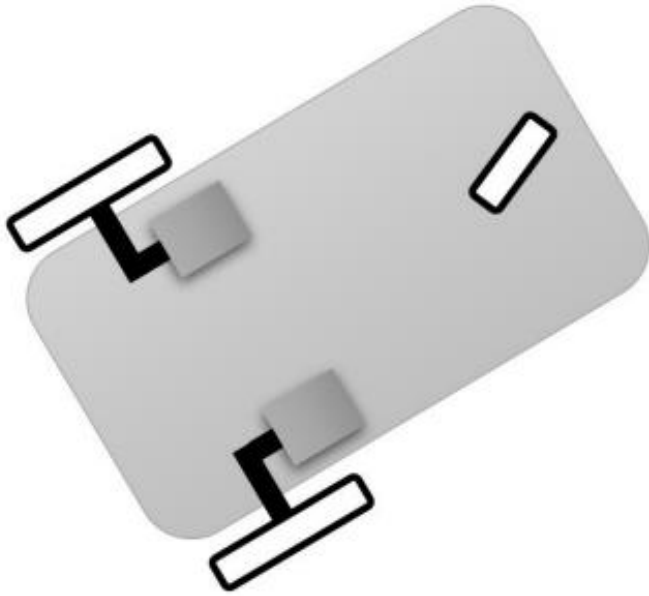


FIGURE 16 : Robot à 2 roues motrices et une roue libre

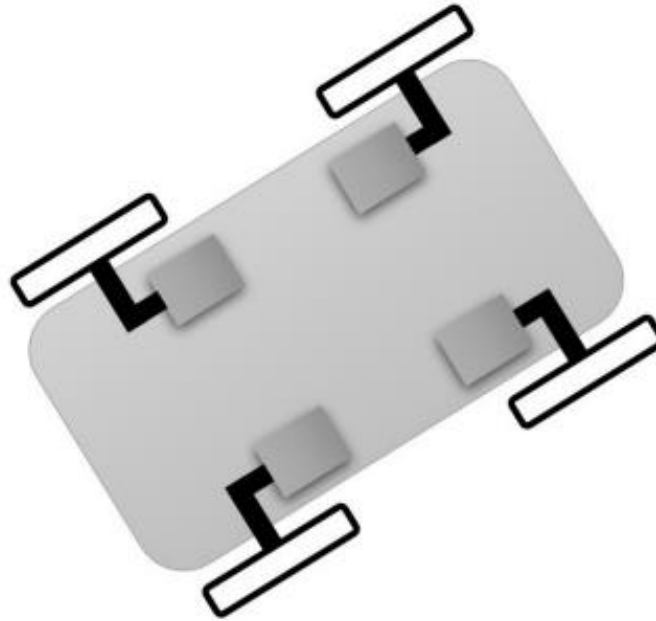


FIGURE 17 : Robot à 4 roues motrices

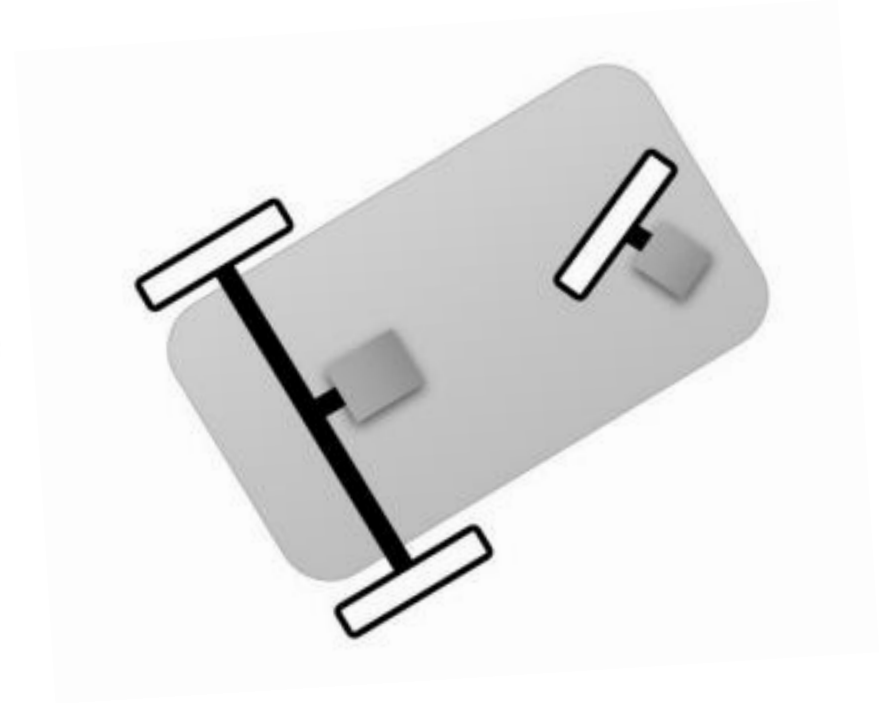


FIGURE 18 : Robot à 2 roues motrices et une roue directionnelle

Le robot Arduino Uno R3

La carte Arduino
Uno R3

Batterie lithium
3.7V

L293D Motor
Driver

Fils de
prototypage

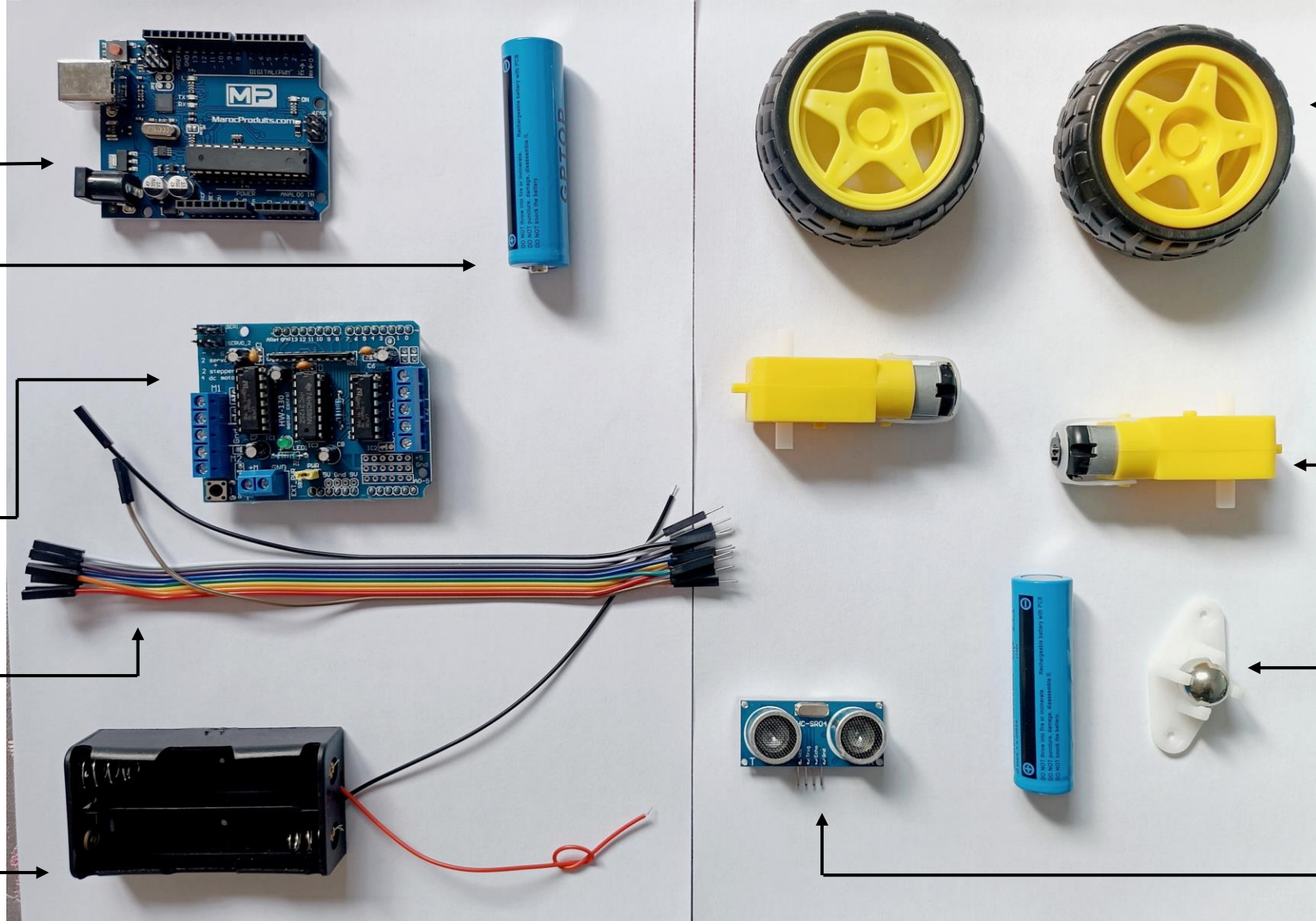
Support de 2
batteries

Roue

Moteur à
courant continu

Roue castor

Capteur
ultrason
HC-SR04



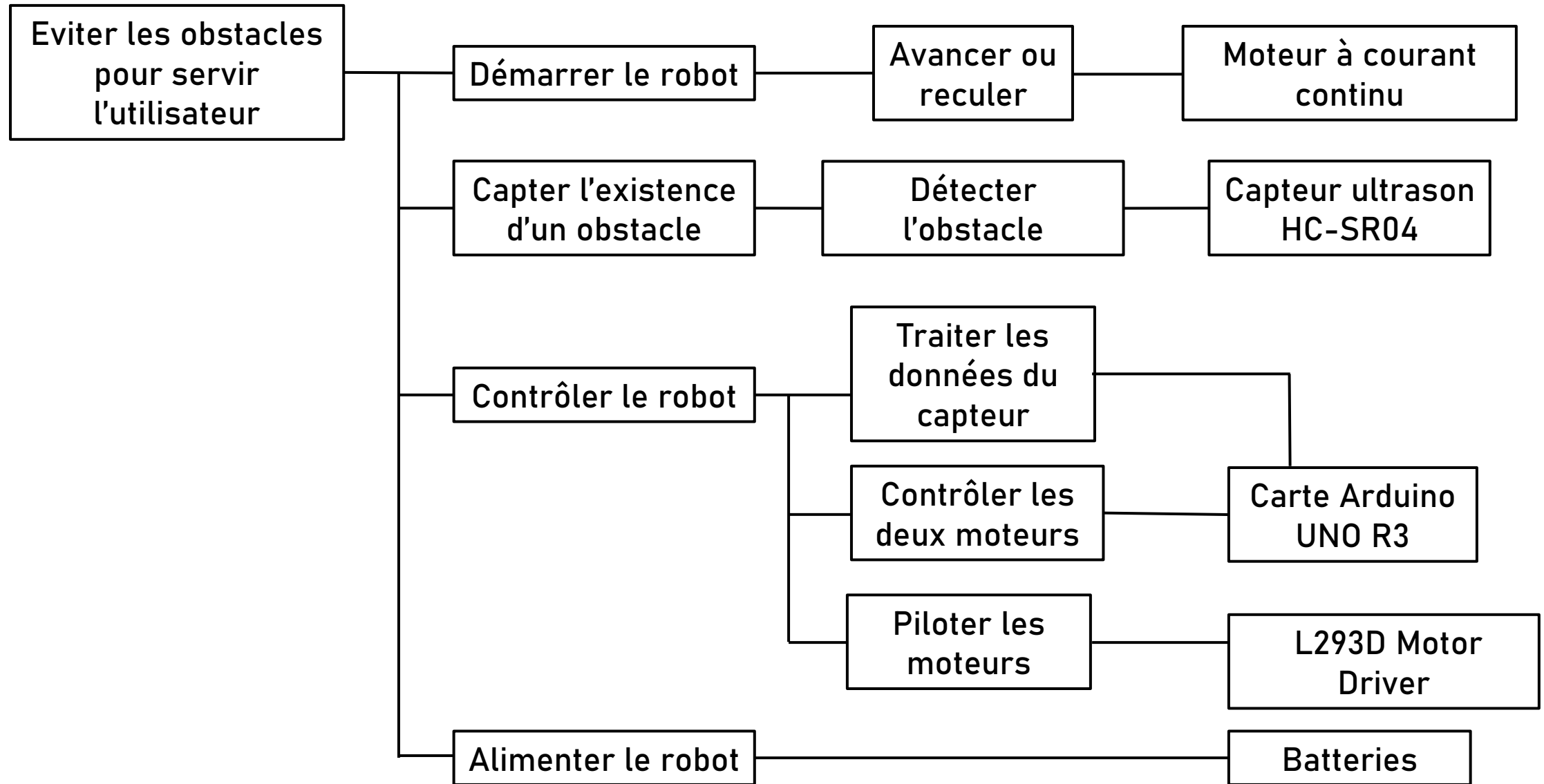


FIGURE 19 : Diagramme FAST du robot

Navigation du robot

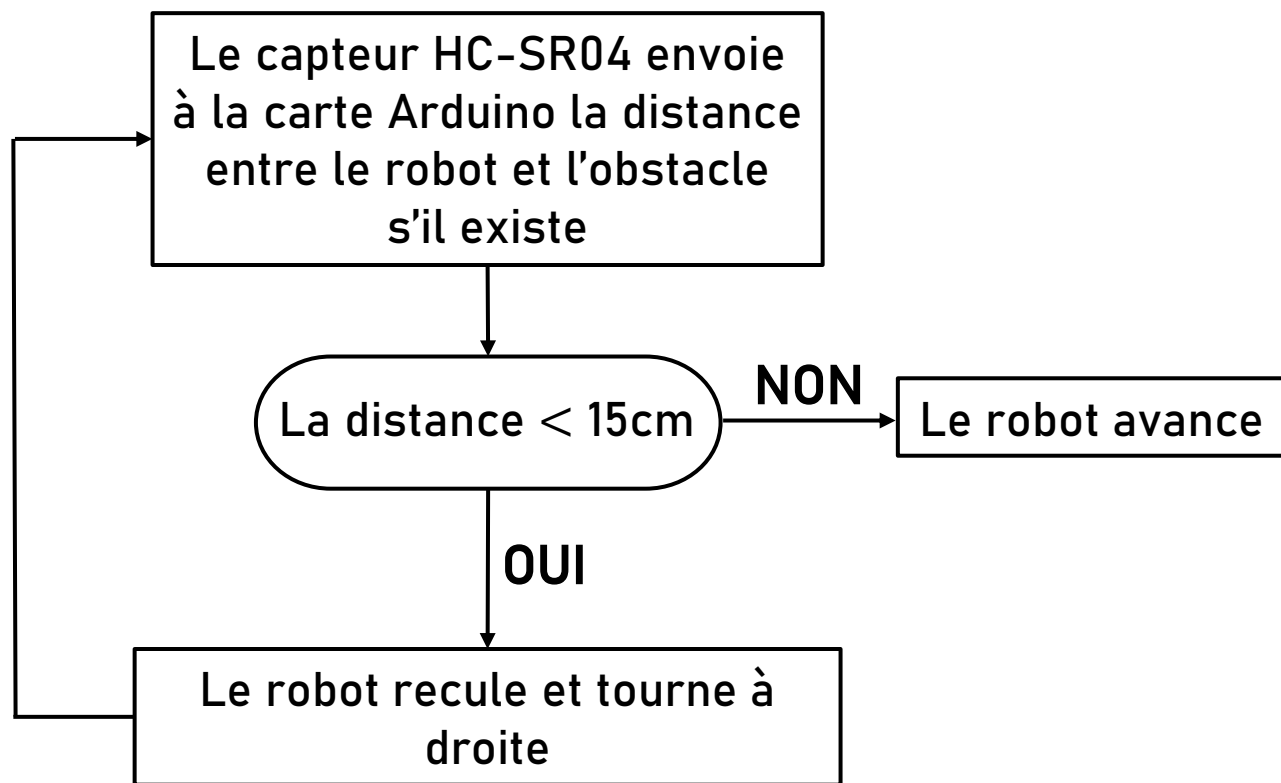


FIGURE 20 : Le schéma de fonctionnement du robot Arduino

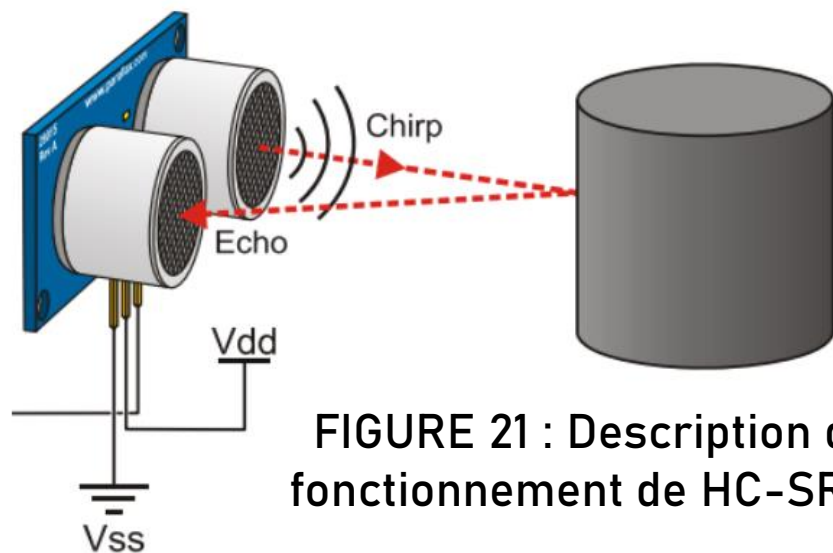


FIGURE 21 : Description du fonctionnement de HC-SR04

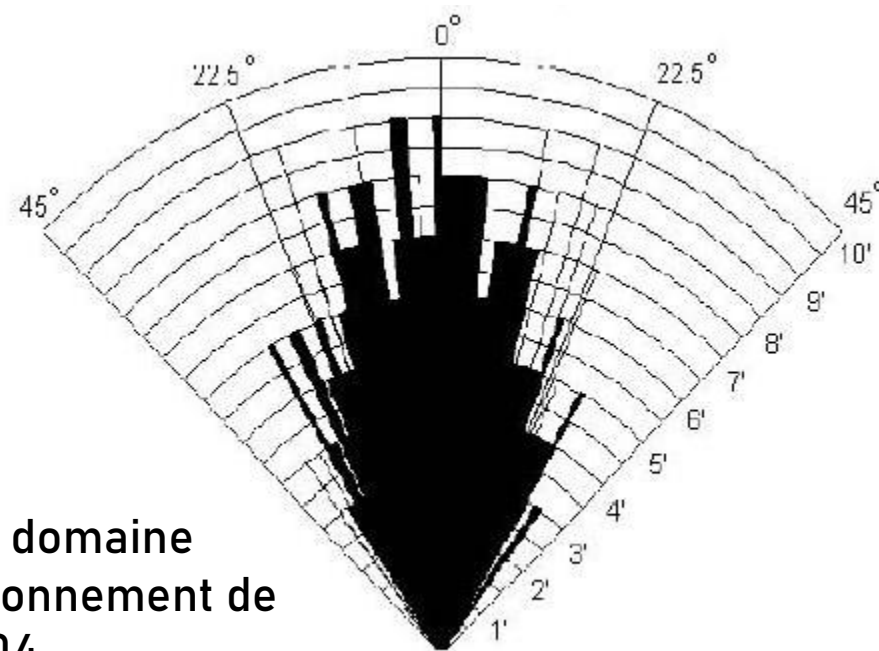
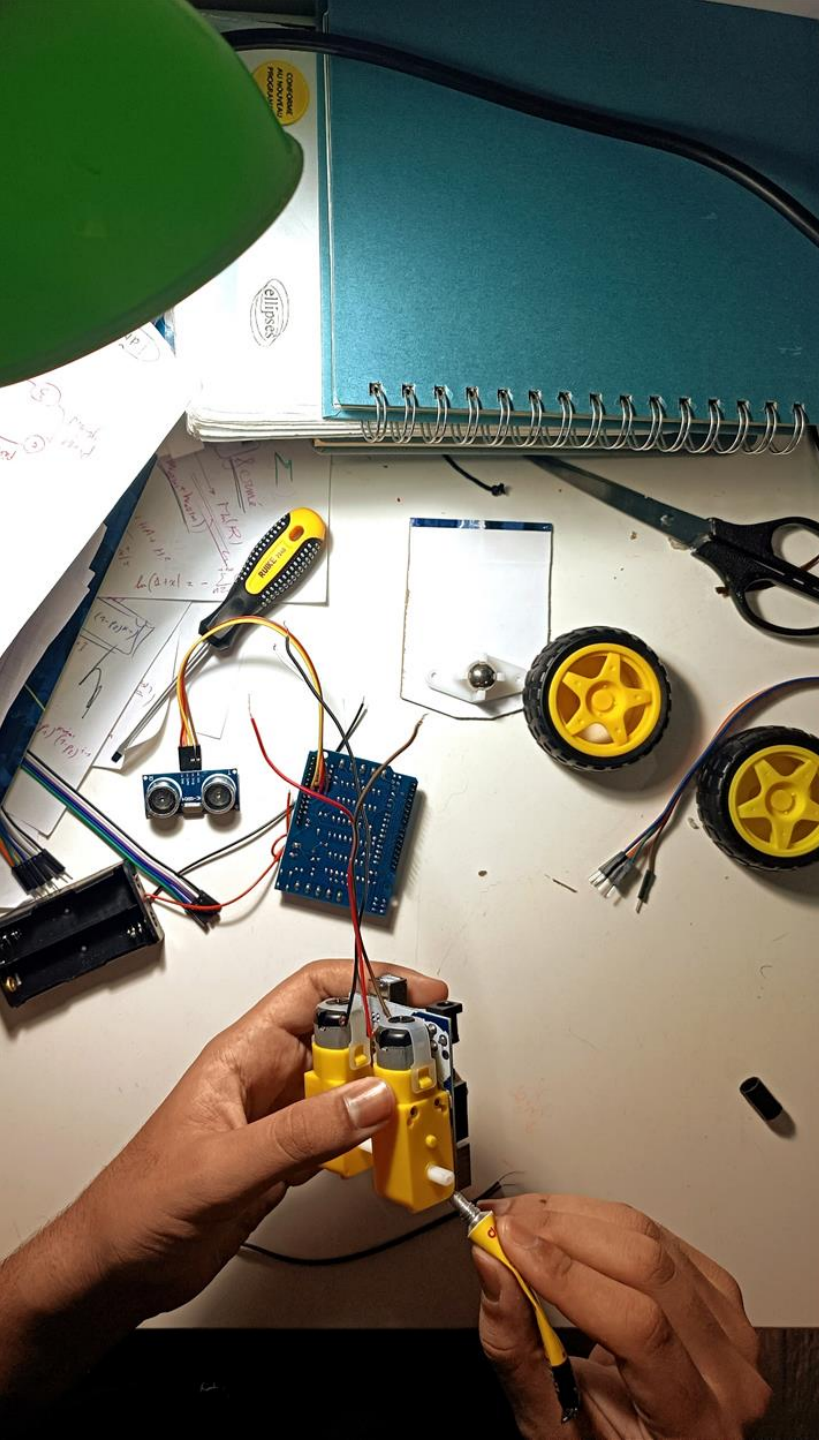


FIGURE 22 : Le domaine angulaire de fonctionnement de HC-SR04



Robotique

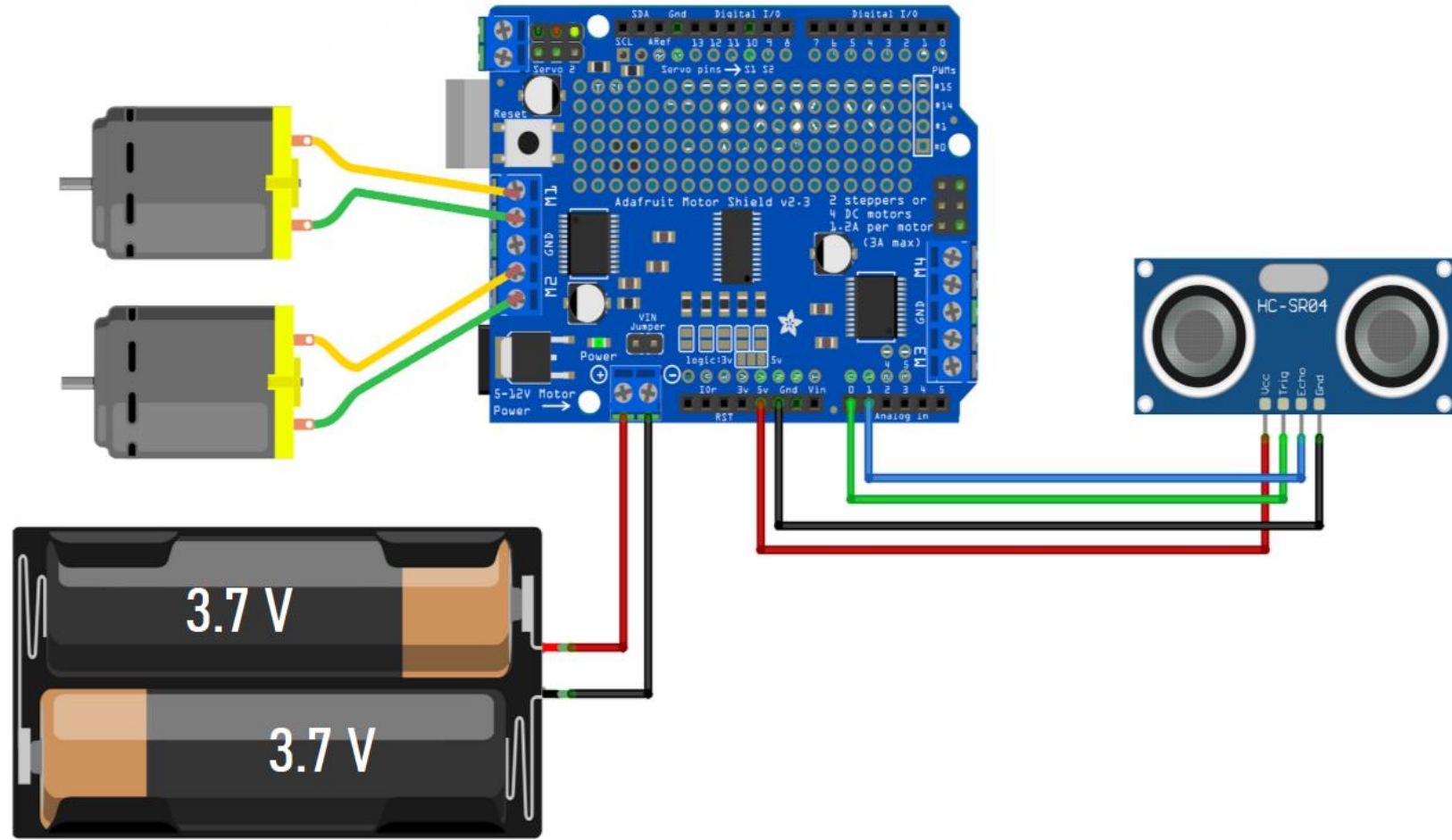


FIGURE 23 : Le montage du robot

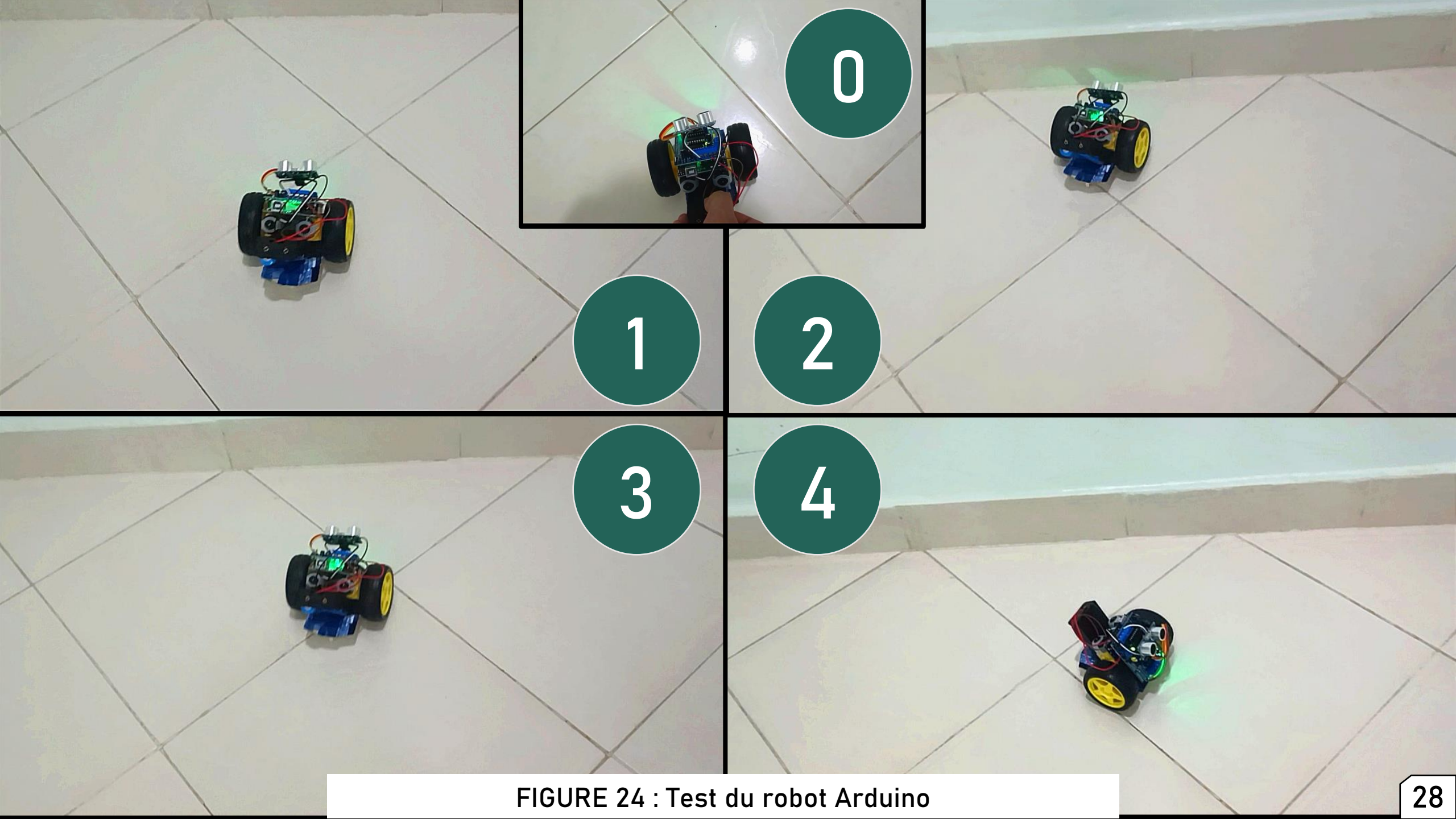


FIGURE 24 : Test du robot Arduino

CONCLUSION

WORLDWIDE ATTENTION

Amnax

Code python de génération aléatoire des labyrinthes 16x16 :

```
generateur de labyrinthe.py - C:\Users\Client\Desktop\Marouane\micromouse\maze\ge...
File Edit Format Run Options Window Help
1 import pygame
2 import time
3 import random
4
5 # Configuration de la fenêtre pygame
6 LARGEUR = 500
7 HAUTEUR = 600
8 FPS = 30
9
10 BLANC = (255, 255, 255)
11 VERT = (0, 255, 0)
12 BLEU = (0, 0, 100)
13
14 # Initialiser Pygame
15 pygame.init()
16 ecran = pygame.display.set_mode((LARGEUR, HAUTEUR))
17 pygame.display.set_caption("Générateur de labyrinthe")
18 horloge = pygame.time.Clock()
19
20 # Initialisation des variables du labyrinthe
21 x = 0
22 y = 0
23 w = 20
24 grille = []
25 visites = []
26 pile = []
27
28 def construire_grille(x, y, w):
29     for i in range(1, 17):
30         x = 20
31         y = y + 20
32         for j in range(1, 17):
33             pygame.draw.line(ecran, BLEU, [x, y], [x + w, y])
34             pygame.draw.line(ecran, BLEU, [x + w, y], [x + w, y + w])
35             pygame.draw.line(ecran, BLEU, [x + w, y + w], [x, y + w])
36             pygame.draw.line(ecran, BLEU, [x, y + w], [x, y])
37             grille.append((x, y))
38             x = x + 20
39
```

```

40 def pousser_haut(x, y):
41     pygame.draw.rect(ecran, BLANC, (x + 1, y - w + 1, 19, 39), 0)
42     pygame.display.update()
43
44 def pousser_bas(x, y):
45     pygame.draw.rect(ecran, BLANC, (x + 1, y + 1, 19, 39), 0)
46     pygame.display.update()
47
48 def pousser_gauche(x, y):
49     pygame.draw.rect(ecran, BLANC, (x - w + 1, y + 1, 39, 19), 0)
50     pygame.display.update()
51
52 def pousser_droite(x, y):
53     pygame.draw.rect(ecran, BLANC, (x + 1, y + 1, 39, 19), 0)
54     pygame.display.update()
55
56 def cellule_unique(x, y):
57     pygame.draw.rect(ecran, VERT, (x + 1, y + 1, 18, 18), 0)
58     pygame.display.update()
59
60 def cellule_retour(x, y):
61     pygame.draw.rect(ecran, BLANC, (x + 1, y + 1, 18, 18), 0)
62     pygame.display.update()
63
64 def creer_labyrinthe(x, y):
65     cellule_unique(x, y)
66     pile.append((x, y))
67     visites.append((x, y))
68     while len(pile) > 0:
69         time.sleep(.05) # bien visualiser la méthode de construction du labyrin
70         cellule = []
71         if (x + w, y) not in visites and (x + w, y) in grille:
72             cellule.append("droite")
73         if (x - w, y) not in visites and (x - w, y) in grille:
74             cellule.append("gauche")
75         if (x, y + w) not in visites and (x, y + w) in grille:
76             cellule.append("bas")
77         if (x, y - w) not in visites and (x, y - w) in grille:
78             cellule.append("haut")
79

```

```

80         if len(cellule) > 0:
81             cellule_choisie = random.choice(cellule)
82             if cellule_choisie == "droite":
83                 pousser_droite(x, y)
84                 x = x + w
85                 visites.append((x, y))
86                 pile.append((x, y))
87             elif cellule_choisie == "gauche":
88                 pousser_gauche(x, y)
89                 x = x - w
90                 visites.append((x, y))
91                 pile.append((x, y))
92             elif cellule_choisie == "bas":
93                 pousser_bas(x, y)
94                 y = y + w
95                 visites.append((x, y))
96                 pile.append((x, y))
97             elif cellule_choisie == "haut":
98                 pousser_haut(x, y)
99                 y = y - w
100                 visites.append((x, y))
101                 pile.append((x, y))
102         else:
103             x, y = pile.pop()
104             cellule_unique(x, y)
105             time.sleep(0.03)
106             cellule_retour(x, y)
107
108 x, y = 20, 20
109 construire_grille(40, 0, 20)
110 creer_labyrinthe(x, y)
111
112 # Boucle pygame
113 en_cours = True
114 while en_cours:
115     horloge.tick(FPS)
116     for evenement in pygame.event.get():
117         if evenement.type == pygame.QUIT:
118             en_cours = False
119
120 pygame.quit()

```

La 2^{ème} version
du code python
basé sur DFS
et générant un
labyrinthe
16x16, ainsi que
sa solution :

```
generateur de labyrinthe 2.0.py - C:/Users/Client/Desktop/Marouane/micromouse/maze...
File Edit Format Run Options Window Help
1 import pygame
2 import time
3 import random
4
5 # Configurer la fenêtre pygame
6 LARGEUR = 500
7 HAUTEUR = 600
8 FPS = 30
9
10 BLANC = (255, 255, 255)
11 VERT = (0, 255, 0)
12 BLEU = (0, 0, 100)
13 ROUGE = (255, 0, 0)
14
15 # Initialiser Pygame
16 pygame.init()
17 pygame.mixer.init()
18 ecran = pygame.display.set_mode((LARGEUR, HAUTEUR))
19 pygame.display.set_caption("Générateur de Labyrinthe 2.0")
20 horloge = pygame.time.Clock()
21
22 # Configurer les variables du labyrinthe
23 x = 0
24 y = 0
25 w = 20
26 grille = []
27 visité = []
28 pile = []
29 solution = {}
30
```



```

31 # Construire la grille
32 def construire_grille(x, y, w):
33     for i in range(1, 17):
34         x = 20
35         y = y + 20
36         for j in range(1, 17):
37             pygame.draw.line(ecran, BLEU, [x, y], [x + w, y])
38             pygame.draw.line(ecran, BLEU, [x + w, y], [x + w, y + w])
39             pygame.draw.line(ecran, BLEU, [x + w, y + w], [x, y + w])
40             pygame.draw.line(ecran, BLEU, [x, y + w], [x, y])
41             grille.append((x, y))
42             x = x + 20
43
44 def pousser_haut(x, y):
45     pygame.draw.rect(ecran, BLANC, (x + 1, y - w + 1, 19, 39), 0)
46     pygame.display.update()
47
48 def pousser_bas(x, y):
49     pygame.draw.rect(ecran, BLANC, (x + 1, y + 1, 19, 39), 0)
50     pygame.display.update()
51
52 def pousser_gauche(x, y):
53     pygame.draw.rect(ecran, BLANC, (x - w + 1, y + 1, 39, 19), 0)
54     pygame.display.update()
55
56 def pousser_droite(x, y):
57     pygame.draw.rect(ecran, BLANC, (x + 1, y + 1, 39, 19), 0)
58     pygame.display.update()
59
60 def cellule_unique(x, y):
61     pygame.draw.rect(ecran, VERT, (x + 1, y + 1, 18, 18), 0)
62     pygame.display.update()
63
64 def cellule_retour(x, y):
65     pygame.draw.rect(ecran, BLANC, (x + 1, y + 1, 18, 18), 0)
66     pygame.display.update()
67
68 def cellule_solution(x, y):
69     pygame.draw.rect(ecran, ROUGE, (x + 8, y + 8, 5, 5), 0)
70     pygame.display.update()

```

```

72 def creuser_labyrinthe(x, y):
73     cellule_unique(x, y)
74     pile.append((x, y))
75     visité.append((x, y))
76     while len(pile) > 0:
77         time.sleep(.01)
78         cellule = []
79         if (x + w, y) not in visité and (x + w, y) in grille:
80             cellule.append("droite")
81         if (x - w, y) not in visité and (x - w, y) in grille:
82             cellule.append("gauche")
83         if (x, y + w) not in visité and (x, y + w) in grille:
84             cellule.append("bas")
85         if (x, y - w) not in visité and (x, y - w) in grille:
86             cellule.append("haut")
87         if len(cellule) > 0:
88             cellule_choisie = random.choice(cellule)
89             if cellule_choisie == "droite":
90                 pousser_droite(x, y)
91                 solution[(x + w, y)] = x, y
92                 x = x + w
93                 visité.append((x, y))
94                 pile.append((x, y))
95             elif cellule_choisie == "gauche":
96                 pousser_gauche(x, y)
97                 solution[(x - w, y)] = x, y
98                 x = x - w
99                 visité.append((x, y))
100                 pile.append((x, y))
101             elif cellule_choisie == "bas":
102                 pousser_bas(x, y)
103                 solution[(x, y + w)] = x, y
104                 y = y + w
105                 visité.append((x, y))
106                 pile.append((x, y))

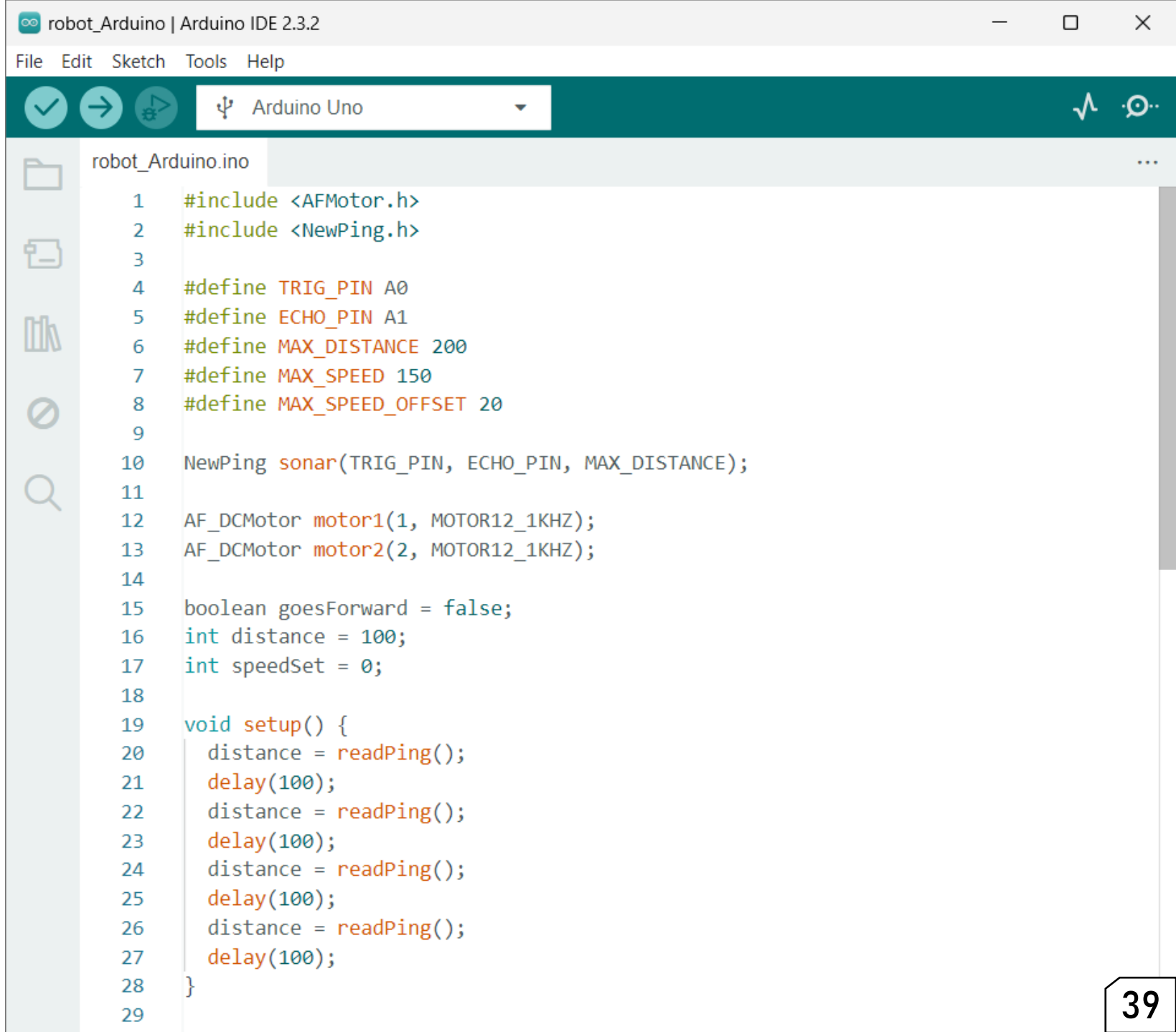
```

```

107         elif cellule_choisie == "haut":
108             pousser_haut(x, y)
109             solution[(x, y - w)] = x, y
110             y = y - w
111             visité.append((x, y))
112             pile.append((x, y))
113         else:
114             x, y = pile.pop()
115             cellule_unique(x, y)
116             time.sleep(.05)
117             cellule_retour(x, y)
118
119     def tracer_route_retour(x, y):
120         cellule_solution(x, y)
121         while (x, y) != (20, 20):
122             x, y = solution[x, y]
123             cellule_solution(x, y)
124             time.sleep(.01)
125
126     x, y = 20, 20
127     construire_grille(40, 0, 20)
128     creuser_labyrinthe(x, y)
129     tracer_route_retour(320, 320)
130
131     # Boucle pygame
132     en_cours = True
133     while en_cours:
134         horloge.tick(FPS)
135         for event in pygame.event.get():
136             if event.type == pygame.QUIT:
137                 en_cours = False
138

```

Code Arduino du robot éviteur d'obstacles :



The screenshot shows the Arduino IDE 2.3.2 interface. The top bar includes the menu (File, Edit, Sketch, Tools, Help) and a toolbar with icons for checking, running, and uploading code, along with a dropdown menu for the board (Arduino Uno). The left sidebar contains icons for file explorer, serial monitor, and search. The main editor window displays the code for 'robot_Arduino.ino'.

```
robot_Arduino.ino
1  #include <AFMotor.h>
2  #include <NewPing.h>
3
4  #define TRIG_PIN A0
5  #define ECHO_PIN A1
6  #define MAX_DISTANCE 200
7  #define MAX_SPEED 150
8  #define MAX_SPEED_OFFSET 20
9
10 NewPing sonar(TRIG_PIN, ECHO_PIN, MAX_DISTANCE);
11
12 AF_DCMotor motor1(1, MOTOR12_1KHZ);
13 AF_DCMotor motor2(2, MOTOR12_1KHZ);
14
15 boolean goesForward = false;
16 int distance = 100;
17 int speedSet = 0;
18
19 void setup() {
20     distance = readPing();
21     delay(100);
22     distance = readPing();
23     delay(100);
24     distance = readPing();
25     delay(100);
26     distance = readPing();
27     delay(100);
28 }
29
```

```
30 void loop() {
31     delay(40);
32
33     if (distance <= 15) {
34         moveStop();
35         delay(100);
36         moveBackward();
37         delay(300);
38         moveStop();
39         delay(200);
40
41         // Turn right by default
42         turnRight();
43         delay(300);
44         moveStop();
45     } else {
46         moveForward();
47     }
48     distance = readPing();
49 }
50
51 int readPing() {
52     delay(70);
53     int cm = sonar.ping_cm();
54     if (cm == 0) {
55         cm = 250;
56     }
57     return cm;
58 }
59
60 void moveStop() {
61     motor1.run(RELEASE);
62     motor2.run(RELEASE);
63 }
64
```



```
65 void moveForward() {
66     if (!goesForward) {
67         goesForward = true;
68         motor1.run(FORWARD);
69         motor2.run(FORWARD);
70         for (speedSet = 0; speedSet < MAX_SPEED; speedSet += 2) {
71             motor1.setSpeed(speedSet);
72             motor2.setSpeed(speedSet);
73             delay(5);
74         }
75     }
76 }
77
78 void moveBackward() {
79     goesForward = false;
80     motor1.run(BACKWARD);
81     motor2.run(BACKWARD);
82     for (speedSet = 0; speedSet < MAX_SPEED; speedSet += 2) {
83         motor1.setSpeed(speedSet);
84         motor2.setSpeed(speedSet);
85         delay(5);
86     }
87 }
88
89 void turnRight() {
90     motor1.run(FORWARD);
91     motor2.run(BACKWARD);
92     delay(160);
93     motor1.run(FORWARD);
94     motor2.run(FORWARD);
95 }
```

