

RoboCoda

Plateforme robotique modulaire

Programmation avec l'interface de développement
Arduino





Le design matériel de l'Arduino est distribué sous licence Creative Commons et est disponible sur le site d'Arduino.
Le code source de l'environnement de programmation et les bibliothèques embarquées sont disponibles sous licence GNU.
AutoProgUno est un système développé par la Sté A4, qui utilise la carte Arduino UNO.



L'ensemble des ressources numériques disponibles autour de nos projets et maquettes sont téléchargeables librement et gratuitement sur www.a4.fr

La duplication de ce dossier est autorisée sans limite de quantité au sein des établissements scolaires, aux seules fins pédagogiques, à la condition que soit cité le nom de l'éditeur : Sté A4.

La copie ou la diffusion par quelque moyen que ce soit à des fins commerciales n'est pas autorisée sans l'accord de la Sté A4.



Edité par la société A4 Technologie
 Tél. : 01 64 86 41 00 - Fax : 01 64 46 31 19
www.a4.fr

SOMMAIRE

1. Introduction.....	2
1.1. Prérequis souhaitables	2
1.2. Organisation de ce dossier	2
2. Eléments nécessaires	3
2.1. Matériels	3
2.2. Logiciels.....	3
2.3. Ressources complémentaires	3
3. Tableau d'affectation des Entrées/Sorties Arduino.....	4
4. Mise en service des applications.....	5
5. Fiche technique N°1 – F1-Avancer	7
6. Fiche technique N°2 – F2-Gauche	9
7. Fiche technique N°3 – F3-Accélération	11
8. Fiche technique N°4 – F4-Compteur.....	13
9. Fiche technique N°5 – M1-Arret	15
10. Fiche technique N°6 – M2-Changement_direction	17
11. Fiche technique N°7 – M3-Dégagement	19
12. Fiche technique N°8 – DL1-Stop.....	21
13. Fiche technique N°9 – DL2-Suivi_de_ligne	23
14. Fiche technique N°10 – DL3-Piste	25
15. Fiche technique N°11 – DL4-Perimetre.....	27
16. Fiche technique N°12 – U1-Proximite.....	29
17. Fiche technique N°13 – U2-Slalom	31
18. Fiche technique N°14 – U3-Cible	33

1. Introduction

Ce document illustre l'utilisation de la version de base du **robot CoDa** d'A4.

Les applications proposées permettent de programmer et de piloter le robot dans son environnement. Toutes les options y sont traitées ainsi que des fiches pour accompagner l'élève durant tout le projet d'initiation.

Les programmes sont développés avec les IDE* :

- **IDE Arduino** pour les applications robotique ;

* IDE : *Environnement de Développement Intégré*



Programmation
Interface



1.1. Prérequis souhaitables

La mise en œuvre des applications suppose que l'utilisateur ait des notions de base autour des logiciels et matériels utilisés. Il est utile de maîtriser les outils de programmations de bases avant de s'initier avec les programmes du robot CoDa

1.2. Organisation de ce dossier

Les applications se présentent sous forme de fiches classées par ordre de difficulté croissante.

Elles permettent de découvrir la manière d'établir une communication bidirectionnelle entre le smartphone et l'interface de pilotage du robot.

Des propositions de modifications sont suggérées à la fin de chaque fiche.

2. Éléments nécessaires

2.1. Matériels

- **Robot CoDa** monté avec la **carte Arduino Uno**.
- **Câble de programmation. USB type B** pour la programmation de la carte Arduino.

2.2. Logiciels

- **IDE Arduino** (<http://arduino.cc/en/Main/Software>) pour la programmation de l'interface **AutoProgUno**

* *IDE : Environnement de Développement Intégré*

2.3. Ressources complémentaires

Des ressources complémentaires sont disponibles sur www.a4.fr

- Dossier Technique Robot CoDa
- Dossier App Inventor 2 pour prendre en main AppInventor 2
- Guide d'utilisation PICAXE Logicator

D'autres ressources sont également disponibles sur internet. Vous pouvez entrer les mots clés suivants pour les localiser avec un moteur de recherche :

Tuto app inventor 2

Tuto picaxe

Tuto arduino uno

Carte moteur Axe 408

Robot CoDa est un robot muni de deux roues activées par deux motoréducteurs.

Il est piloté par une carte programmable Arduino auquel on peut adjoindre différents capteurs.

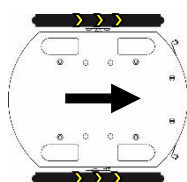
Le module de pilotage dispose de :

- 16 entrées pour des capteurs ;
- 4 sorties spéciales qui s'ajoutent à celles utilisées pour la commande des moteurs.

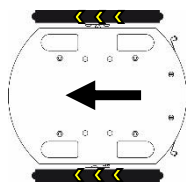
Déplacements :

Robot CoDa peut se déplacer dans toutes les directions :

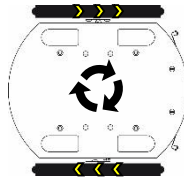
MARCHE AVANT



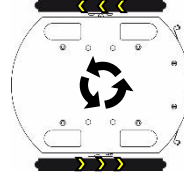
MARCHE ARRIERE



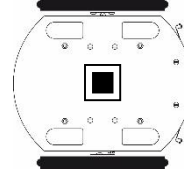
TOURNER A DROITE



TOURNER A GAUCHE



ARRÊT



Il est possible de programmer le robot pour qu'il tourne juste autour d'un seul axe (une seule roue).

Ces instructions sont modélisées par un robot avec une simple flèche en virage. La courbe et le sens de la flèche indiquera la manœuvre qu'effectuera le robot.

3. Tableau d'affectation des Entrées/Sorties Arduino

Ci-dessus le tableau résumant la connexion entre les broches de la carte AXE 408 et la carte Arduino Uno. Ce tableau sert principalement à faire le lien entre le câblage des capteurs/actionneurs et la programmation du microcontrôleur.

N° Broche AXE 408	MODULE CAPTEUR/ACTIONNEURS	N° Broche Arduino	Nom de Variable Arduino
0 (RX)	Broche libre (réservé option Bluetooth)	0	
1 (TX)	Broche libre (réservé option Bluetooth)	1	
2	Détecteur de ligne Droit	2	detecteur_droit
3	Détecteur de ligne Central	3	detecteur_central
4	Détecteur de ligne Gauche	4	detecteur_gauche
5	Télémètre à Ultrasons	5	ultrasons
6	Broche libre (réservé servomoteur axe horizontal)	6	
7	Broche libre (réservé servomoteur axe vertical)	7	
8	Direction moteur droit	8	direction_moteur_droit
9	Puissance moteur droit	9	puissance_moteur_droit
10	Puissance moteur gauche	10	puissance_moteur_gauche
11	Direction moteur gauche	11	direction_moteur_gauche
12	Broche libre	12	
13	Broche libre	13	
A0	Broche libre	A0	
A1	Broche libre	A1	
A2	Broche libre	A2	
A3	Broche libre	A3	
A4	Microrupteur gauche	A4	contact_gauche
A5	Microrupteur droit	A5	contact_droit


4. Mise en service des applications

Les applications proposées dans les fiches suivantes sont créées avec différents IDE (Environnement de Développement Intégré).

Pour chaque application, un ensemble de fichiers à charger dans l'interface programmable Arduino est proposé.





Les éléments clés sont mis en évidence et des suggestions de modification sont proposées en vue d'adapter les programmes à un nouveau contexte d'utilisation.

Il est indispensable que la fonction Bluetooth soit mise en service dans les paramètres du Smartphone ou de la Tablette Android.








	http://arduino.cc/en/Main/Software (rubrique Arduino IDE) Cette IDE est gratuite. Elle doit être installée sur un PC.
Matériel associé	<ul style="list-style-type: none">- Interface programmable AutoProgUno ou autre carte Arduino compatible.- Cordon de liaison USB type imprimante avec le PC pour transférer les programmes dans l'interface AutoProgUno.
Fichiers	Fichiers .INO à ouvrir dans l'IDE Arduino et à téléverser dans l'interface.
Notes	Les programmes proposés dans ce document sont développés en C avec l'IDE standards Arduino. Il existe d'autres IDE compatibles Arduino (par exemple Scratch ou Ardublock).

Liste des symboles utilisés pour les capteurs:

(Voir le chapitre Montage / Assemblage pour la mise en œuvre des capteurs)

Symbole	Description
	Aucun capteur
	Module capteurs microrupteurs (détection de contact avec un obstacle).
	Module capteurs infrarouges (détection de marquage au sol).
	Module capteur à ultrasons (détection d'obstacle à distance).

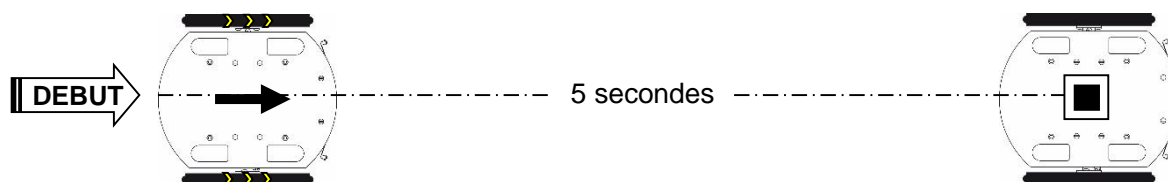
Liste des programmes

Fiche	Capteur	Description	Fichier programme
N°1		Manœuvre de base : Avancer puis s'arrêter.	F1-AVANCER.ino
N°2		Manœuvre de base : Avancer, puis tourner à gauche.	F2-GAUCHE.ino
N°3		Manœuvre de base : Avancer puis accélérer.	F3-ACCELERATION.ino
N°4		Compter un nombre de séquences : Répéter une figure 5 fois de suite puis s'arrêter.	F4-COMPTEUR.ino
N°5		Utilisation capteurs microrupteurs : S'arrêter au contact d'un obstacle.	M1-ARRET.ino
N°6		Utilisation capteurs microrupteurs : Changer de direction au contact d'un obstacle.	M2-CHANGE_DIRECTION.ino
N°7		Utilisation capteurs microrupteurs : Faire une manœuvre de dégagement au contact d'un obstacle.	M3-DEGAGEMENT.ino
N°8		Utilisation capteurs de détection de marquage au sol : S'arrêter sur un marquage au sol.	DL1-STOP.ino
N°9		Utilisation capteurs de détection de marquage au sol : Suivre une ligne marquée au sol.	DL2-SUIVI LIGNE.ino
N°10		Utilisation capteurs de détection de marquage au sol : Evoluer entre deux lignes marquées au sol.	DL3-PISTE.ino
N°11		Utilisation capteurs de détection de marquage au sol : Evoluer dans un périmètre marqué au sol.	DL4-PERIMETRE.ino
N°12		Utilisation capteur détection d'obstacle : S'arrêter à proximité d'un obstacle.	U1-PROXIMITE.ino
N°13		Utilisation capteur détection d'obstacle : Evoluer en évitant des obstacles.	U2-SLALOM.ino
N°14		Utilisation capteur détection d'obstacle : Localiser un obstacle et se diriger vers lui.	U3-CIBLE.ino

5. Fiche technique N°1 – F1-Avancer

But de l'application	Se déplacer en avant pendant 5 secondes puis s'arrêter.
Notions de programmation abordées	Pilotage des moteurs du robot. Prise en main de la conduite du robot CoDa.
Programme associé	Code source Arduino Uno F1-Avancer.ino (Fichiers téléchargeables sur www.a4.fr)
Capteur mis en jeu	Aucun

Illustration



Suggestions de modifications	Modifier le programme pour que le robot s'arrête au bout de 10 secondes.
------------------------------	--



Il est obligatoire de déclarer les pins et les variables dès le début d'un programme en Arduino.

```
const int direction_moteur_gauche = 11;
const int puissance_moteur_gauche = 10;
const int puissance_moteur_droit = 9;
const int direction_moteur_droit = 8;
```

Ensuite, dans la fonction setup(), il est nécessaire d'identifier les broches de l'Arduino comme des entrées ou des sorties. Ici, les moteurs sont déclarés comme des sorties.

```
pinMode (direction_moteur_droit, OUTPUT);
pinMode (direction_moteur_gauche, OUTPUT);
pinMode (puissance_moteur_gauche, OUTPUT);
pinMode (puissance_moteur_droit, OUTPUT);
```

La programmation d'un mouvement du robot CoDa se fera par l'appel de sous fonction pour éviter toute répétition des commandes dans le programme principal.

```
void arret()//Sous fonction arrêt
{
    analogWrite(puissance_moteur_droit, LOW);
    analogWrite(puissance_moteur_gauche, LOW);
    digitalWrite(direction_moteur_droit, LOW);
    digitalWrite(direction_moteur_gauche, LOW);
}
```

Pour effectuer un arrêt, la puissance des moteurs est mise à 0

```
void avancer()//Sous fonction AVANCER
{
    analogWrite(puissance_moteur_droit, puissance_moteur);
    analogWrite(puissance_moteur_gauche, puissance_moteur);
    digitalWrite(direction_moteur_droit, HIGH);
    digitalWrite(direction_moteur_gauche, HIGH);
}
```

Pour effectuer un déplacement vers l'avant, les paramètres direction moteur doivent être activés ("HIGH")

Instructions à établir dans le **Setup()** pour réaliser une seule fois les commandes. Le programme utilise deux sous-fonctions **avancer()** et **arret()** pour clarifier le code.

```
avancer(); // appel de la sous-fonction avancer

delay(5000); //attendre 5 s (la commande "avancer" reste active)

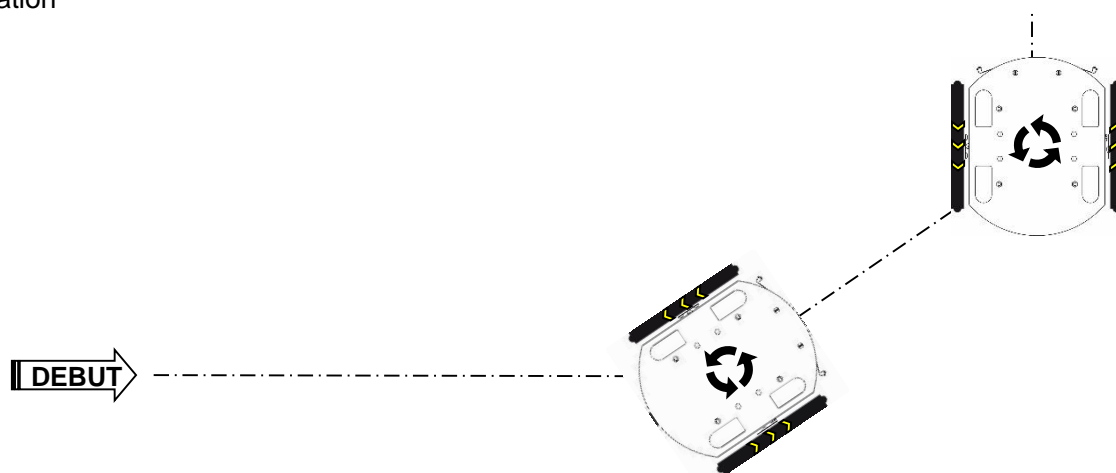
arret();// appel de la sous-fonction arret (la commande "avancer" est désactivée)
```

Note : la fonction void **loop()** n'est pas utilisée car il ne s'agit pas d'un programme qui se reboucle perpétuellement.

6. Fiche technique N°2 – F2-Gauche

But de l'application	Avancer pendant 4 secondes, tourner à gauche, avancer de nouveau
Notions de programmation abordées	Utilisation des sous-fonctions et rebouclage d'un programme
Programme associé	Code source Arduino Uno F2-Gauche.ino (Fichiers téléchargeables sur www.a4.fr)
Capteur mis en jeu	Aucun

Illustration



Suggestions de modifications	Modifier le programme pour que le robot tourne d'un angle plus faible
------------------------------	---



Dans le but du programme, il est indiqué que le robot doit effectuer une rotation vers la gauche. Il est nécessaire de déclarer une fonction **gauche()** de même type que les sous-fonctions **avancer()** et **arret()**

```
void gauche()//Sous fonction GAUHCHE
{
    analogWrite(puissance_moteur_droit, puissance_moteur);
    analogWrite(puissance_moteur_gauche, puissance_moteur);
    digitalWrite(direction_moteur_droit, HIGH);
    digitalWrite(direction_moteur_gauche, LOW);
}
```

Note : Pour que le robot tourne sur lui-même vers la gauche, il faut que le moteur droit avance (direction_moteur_droit,HIGH) et que le moteur gauche recule (direction_moteur_gauche,LOW)

Dans cette situation, les consignes doivent être inscrites dans la fonction **loop()**, car il s'agit d'un programme qui doit se reboucler à l'infini.

```
//Boucle infinie
void loop()
{
    avancer(); // appel de la sous-fonction avancer

    delay(4000); //attendre 4 s (la commande "avancer" reste active)

    gauche(); // appel de la sous-fonction "gauche"

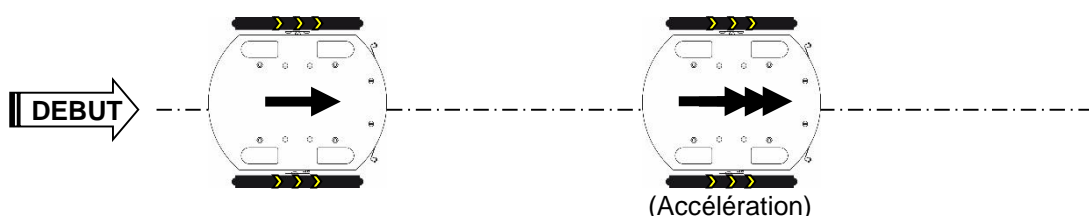
    delay (500); // attente pendant 0,5 s (la commande "gauche" reste active).

}
```

7. Fiche technique N°3 – F3-Accélération

But de l'application	Avancer 1 secondes puis accélérer 1 seconde
Notions de programmation abordées	Modification de la vitesse de déplacement
Programme associé	Code source Arduino Uno F3-Acceleration.ino (Fichiers téléchargeables sur www.a4.fr)
Capteur mis en jeu	Aucun

Illustration



Suggestions de modifications	Ajouter une variable de vitesse pour chaque moteur. Cette manipulation peut être utile pour faire avancer les deux moteurs à la même vitesse.
------------------------------	---

Commentaire du programme d'automatisme :

La vitesse du Robot Coda est réglable pour chaque moteur (gauche et droit) sur une échelle qui s'étend de 0 à 255.

Il est possible d'écrire directement la valeur de la vitesse à la place de la variable ou encore définir une vitesse pour chaque moteur dès le début pour faire avancer le robot en ligne droite s'il diverge dans une direction pour la même consigne de vitesse sur chaque moteur.



La vitesse des moteurs se gèrent par la variable **puissance_moteur**. Elle peut varier entre 0 et 255 où 255 est la vitesse maximum des moteurs. Dans un premier temps, le robot avance à une vitesse de **150** puis après 1 secondes à une vitesse de **250**.

```
void loop()
{
    puissance_moteur = 150; // vitesse des moteurs à 150

    avancer(); // appel de la sous-fonction avancer

    delay(1000); //attendre 2 s (la commande "avancer" reste active)

    puissance_moteur = 250; // vitesse des moteurs à 250

    avancer(); // appel de la sous-fonction avancer

    delay(1000); // Attente pendant 0,5 s (la commande «gauche» reste active).
}
```

La variable **puissance_moteur** consigne la même vitesse pour les deux moteurs. Il est également possible de déclarer une variable pour déterminer la consigne des deux moteurs séparément.

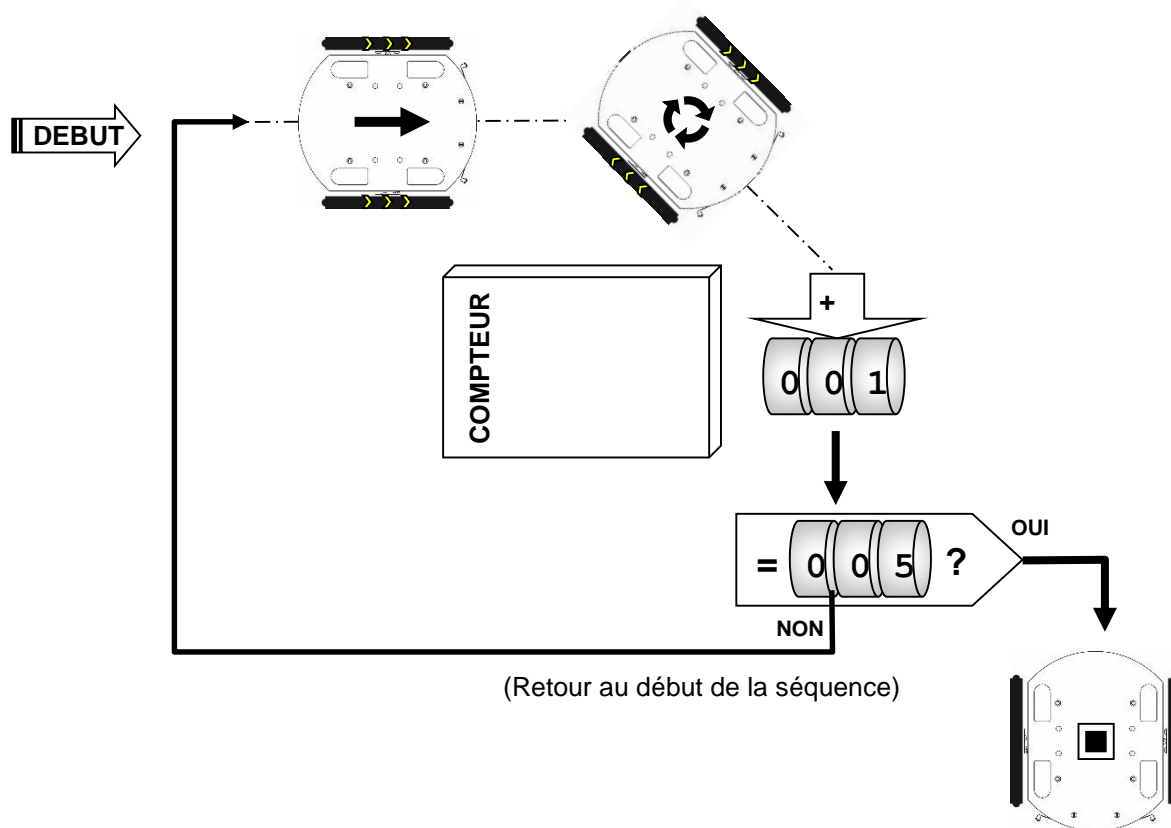
```
void avancer()//Sous fontcion AVANCER
{

    analogWrite(puissance_moteur_droit, puissance_moteur);
    analogWrite(puissance_moteur_gauche, puissance_moteur);
    digitalWrite(direction_moteur_droit, HIGH);
    digitalWrite(direction_moteur_gauche, HIGH);
}
```

8. Fiche technique N°4 – F4-Compteur

But de l'application	Répéter une figure 5 fois de suite puis s'arrêter.
Notions de programmation abordées	Répéter une séquence plusieurs fois. Définir une variable de comptage interne au programme.
Programme associé	Code source Arduino Uno F4-Compteur.ino (Fichiers téléchargeables sur www.a4.fr)
Capteur mis en jeu	Aucun

Illustration



Suggestions de modifications	Utilisation d'un autre type de boucle, modification du nombre d'itérations.
------------------------------	---



Initialisation d'une variable compteur :

```
int compteur=0; // initialisation de la variable compteur
```

Le comptage se fera à l'aide de la boucle **While**. La boucle s'exécute et incrémente la variable **compteur** à chaque passage. Dès lors que compteur = 5 le programme sort de la boucle et le robot s'arrête avec la fonction arrêt.

```
void loop()
{
  while (compteur <5)
  {
    avancer(); // appel de la sous-fonction avancer

    delay(1000); //attendre 1 s (la commande "avancer" reste active)

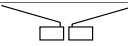
    droite();// appel de la sous-fonction droite (la commande « avancer » est désactivée).

    delay(250); // Attente pendant 0,5 s (la commande «droite» reste active).

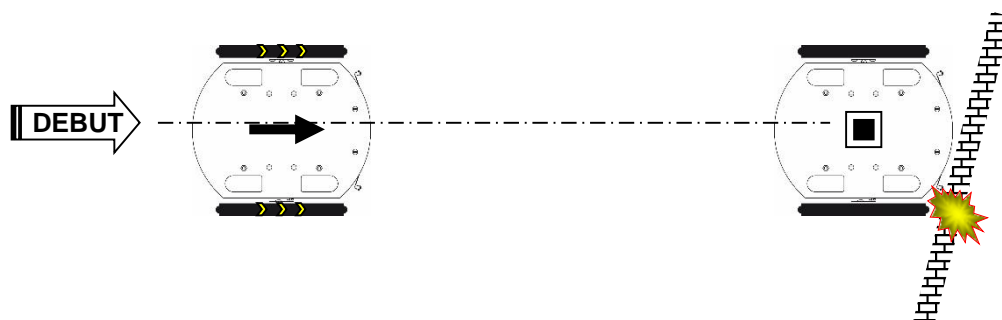
    compteur=compteur+1;
  }

  arret();
}
```


9. Fiche technique N°5 – M1-Arret

But de l'application	S'arrêter au contact d'un obstacle.
Notions de programmation abordées	Test de l'état de chaque capteur microrupteurs droit et gauche.
Programme associé	Code source Arduino Uno M1-Arret.ino (Fichiers téléchargeables sur www.a4.fr)
Capteur mis en jeu	Microrupteurs gauche et droit 

Illustration



Suggestions de modifications	
------------------------------	--



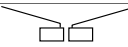
Initialisation du robot en marche avant dans le Setup()

```
// Etat Initial Moteurs  
  
|avancer(); // aller dans la sous-fonction avancer
```

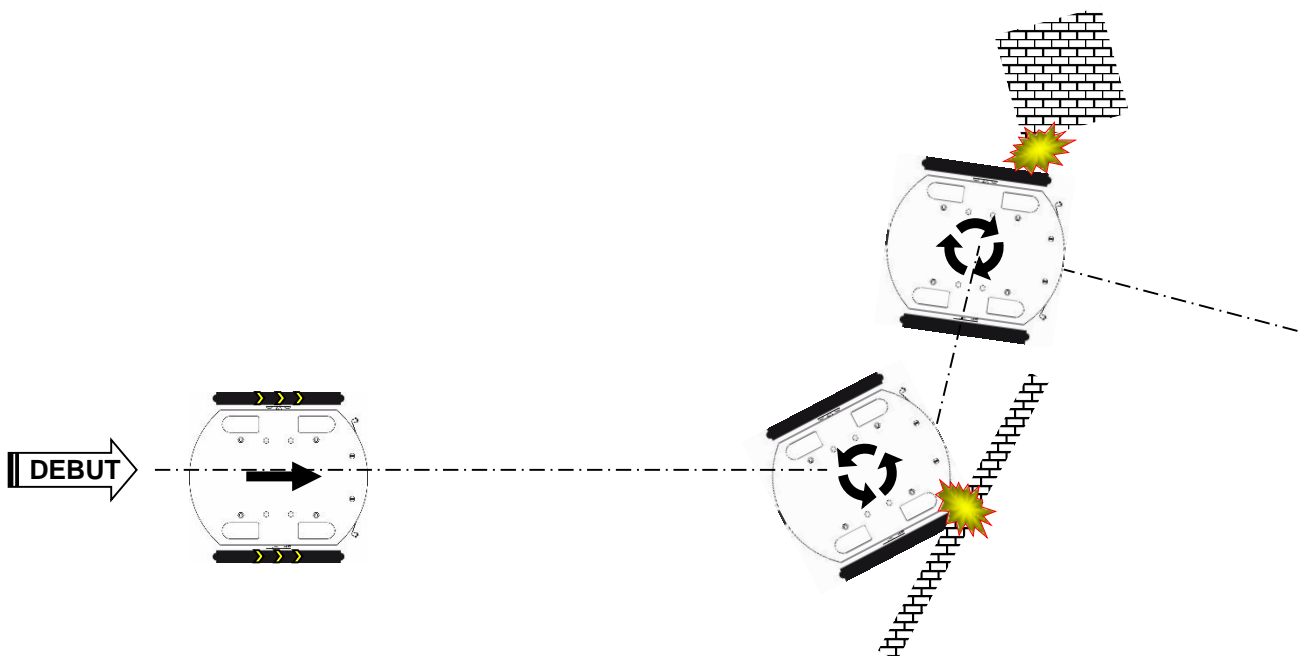
Si un des microrupteurs est activé, le programme rentre dans la sous-fonction arret(), qui arrête complètement le robot.

```
void loop()  
{  
  
  if (digitalRead (contact_droit) == HIGH) // Si interrupteur droit activé  
  {  
    arret();  
  } // Fin de Si  
  
  else if (digitalRead (contact_gauche) == HIGH) // Si interrupteur gauche activé  
  {  
    arret();  
  } // Fin de Si  
  
} // Fin du programme principal
```

10. Fiche technique N°6 – M2-Changement_direction

But de l'application	Changer de direction dès la rencontre d'un obstacle
Notions de programmation abordées	Test de l'état de chaque capteur microrupteurs droit et gauche puis exécute une instruction en fonction du microrupteur activé.
Programme associé	Code source Arduino Uno M2-Changement_direction.ino (Fichiers téléchargeables sur www.a4.fr)
Capteur mis en jeu	Microrupteurs gauche et droit 

Illustration



Suggestions de modifications



Le robot avance tout droit jusqu'à ce qu'il rencontre un obstacle sur ses microrupteurs. Il prend alors la direction opposé de l'obstacle pendant 500 ms avant de continuer tout droit.

```
void loop()
{


  avancer(); // aller dans la sous-fonction avancer

  if (digitalRead (contact_droit) == HIGH) // Si interrupteur droit activé
  {
    gauche(); // aller dans la sous-fonction gauche
    delay(500); // tourner à gauche pendant 500 ms
  } // Fin de Si

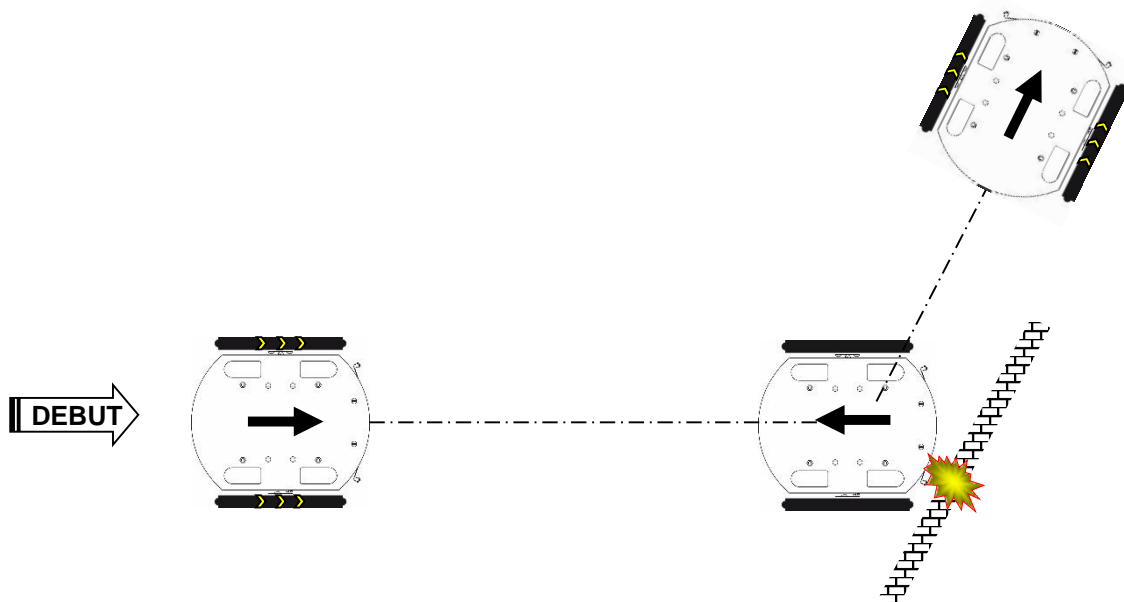
  if (digitalRead (contact_gauche) == HIGH) // Si interrupteur gauche activé
  {
    droite(); // aller dans la sous-fonction droite
    delay(500); // tourner à droite pendant 500 ms
  } // Fin de Si

} // Fin du programme principal
```

11. Fiche technique N°7 – M3-Dégagement

But de l'application	Eviter un obstacle détecté par un des capteurs microrupteurs en reculant dans un premier temps pour se dégager de l'obstacle puis en changeant de direction.
Notions de programmation abordées	Test de l'état de chaque capteur microrupteurs droit et gauche puis exécute une instruction en fonction du microrupteur activé.
Programme associé	Code source Arduino Uno M3-Degagement.ino (Fichiers téléchargeables sur www.a4.fr)
Capteur mis en jeu	Microrupteurs gauche et droit 

Illustration



Suggestions de modifications



Dans un premier temps, la sous-fonction reculer() doit être déclarée.

```
void reculer()//Sous fonction RECULER
{

    analogWrite(puissance_moteur_droit, puissance_moteur);
    analogWrite(puissance_moteur_gauche, puissance_moteur);
    digitalWrite(direction_moteur_droit, LOW);
    digitalWrite(direction_moteur_gauche, LOW);
}
```

Le robot avance tout droit jusqu'à ce qu'il rencontre un obstacle sur ses microrupteurs. Il recule pendant 1 seconde puis prend alors la direction opposé de l'obstacle pendant 500 ms avant de continuer tout droit.

```
void loop()
{


    avancer(); // aller dans la sous-fonction avancer

    if (digitalRead (contact_droit) == HIGH) // Si interrupteur droit activé
    {
        reculer();
        delay(1000); // reculer pendant 1000 ms
        gauche(); // aller dans la sous-fonction gauche
        delay(500); // tourner à gauche pendant 500 ms
    } // Fin de Si

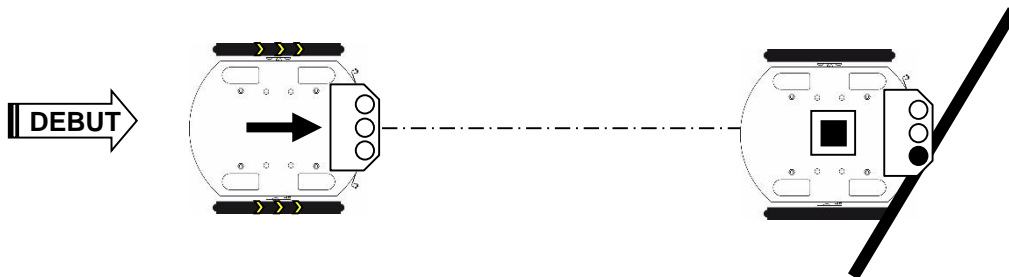
    if (digitalRead (contact_gauche) == HIGH) // Si interrupteur gauche activé
    {
        reculer();
        delay(1000); // reculer pendant 1000 ms
        droite(); // aller dans la sous-fonction droite
        delay(500); // tourner à droite pendant 500 ms
    } // Fin de Si

} // Fin du programme principal
```

12. Fiche technique N°8 – DL1-Stop

But de l'application	Avancer en ligne droite et s'arrêter au croisement d'un marquage au sol.
Notions de programmation abordées	Test de l'état de chaque capteur détection de ligne.
Programme associé	Code source Arduino Uno DL1-Stop.ino (Fichiers téléchargeables sur www.a4.fr)
Capteur mis en jeu	Microrupteurs gauche et droit 

Illustration



Suggestions de modifications	Arrêter le robot lorsque deux capteurs ou plus sont activés.
------------------------------	--



Dans un premier temps, il est obligatoire de déclarer les 3 broches du détecteur de ligne

```
const int detecteur_droit = 2;
const int detecteur_central = 3;
const int detecteur_gauche = 4;
```

Les broches doivent ensuite être déclarées comme des entrées dans la fonction setup().

```
// Définition des pins comme des entrées
pinMode (detecteur_droit, INPUT);
pinMode (detecteur_gauche, INPUT);
pinMode (detecteur_central, INPUT);
```

Le robot est initialisé pour avancer dans la fonction setup().

```
// Etat Initial Moteurs

avancer(); // aller dans la sous-fonction avancer
```

Le robot avance et teste si un détecteur de ligne rencontre une ligne noire. Si la condition est vraie le programme rentre dans la fonction arret() qui arrête les moteurs du robot.


```
void loop()
{

    if (digitalRead(detecteur_central) == HIGH)
    {
        arret();
    }
    else if (digitalRead(detecteur_droit) == HIGH)
    {
        arret();
    }

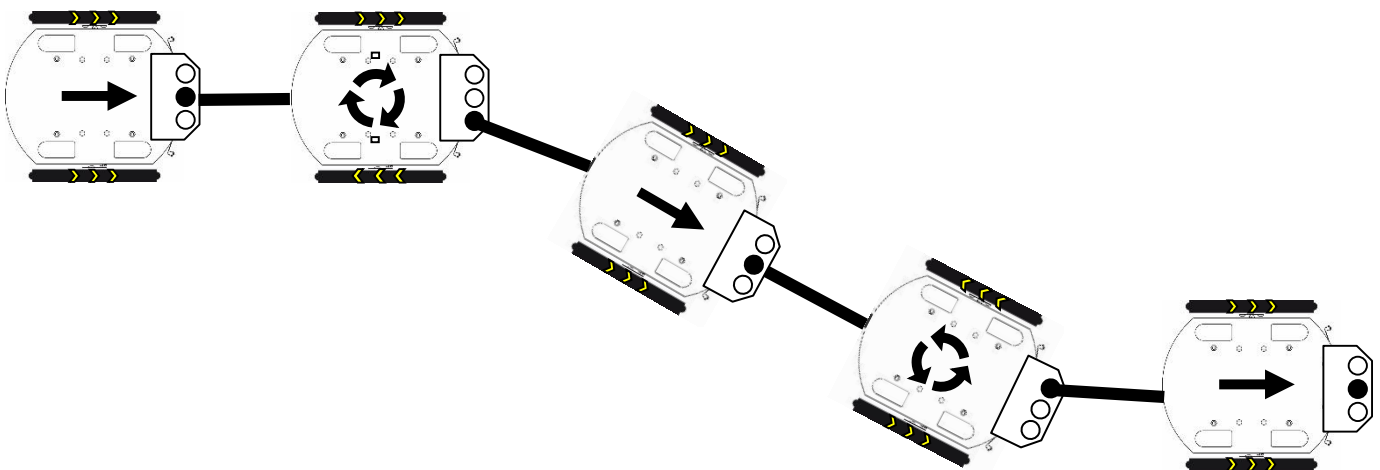
    else if (digitalRead(detecteur_gauche) == HIGH)
    {
        arret();
    }

} // Fin du programme principal
```


13. Fiche technique N°9 – DL2-Suivi_de_ligne

But de l'application	Suivre une ligne marquée au sol
Notions de programmation abordées	Condition sur chaque capteur détection de ligne.
Programme associé	Code source Arduino Uno DL2-Suivi_de_ligne.ino (Fichiers téléchargeables sur www.a4.fr)
Capteur mis en jeu	 Module capteurs infrarouges (détection de marquage au sol).

Illustration



Suggestions de modifications	Créer de nouvelle sous-fonction qui remplacera les sous-fonctions droite() et gauche() et qui permettent de rendre le robot plus fluide dans ses déplacements (n'activer qu'un seul moteur lors d'un virage à droite ou à gauche)
------------------------------	---



Pour suivre une ligne, le robot doit se remettre en permanence sur sa ligne. Dès lors que le détecteur de ligne gauche est actif, le robot est en train de dévier vers la droite. Le robot se redresse alors vers la gauche pour se remettre sur la ligne et vice versa.


Il est important d'écrire la condition du détecteur central en premier car cette condition est prioritaire sur les deux autres.

```
void loop()
{
    if (digitalRead(detecteur_central) == HIGH)
    {
        avancer();
    }
    else if (digitalRead(detecteur_gauche) == HIGH)
    {
        gauche();
    }

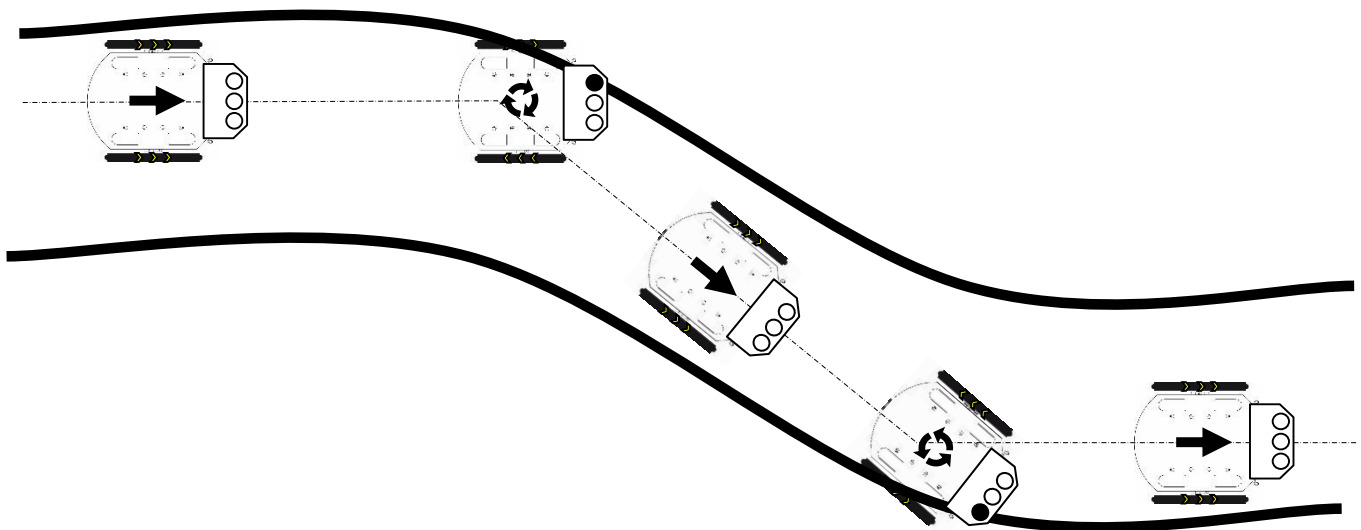
    else if (digitalRead(detecteur_droit) == HIGH)
    {
        droite();
    }

} // Fin du programme principal
```

14. Fiche technique N°10 – DL3-Piste

But de l'application	Evoluer sur une piste délimitée par des marquages au sol
Notions de programmation abordées	Condition sur chaque capteur détection de ligne.
Programme associé	Code source Arduino Uno DL3-Piste.ino (Fichiers téléchargeables sur www.a4.fr)
Capteur mis en jeu	 Module capteurs infrarouges (détection de marquage au sol).

Illustration



Suggestions de modifications	Modifier la structure de la condition du détecteur central dans le cas où le robot détecte une ligne noire sur le détecteur central droit devant lui.
------------------------------	---



Pour rester sur la piste, le robot doit se maintenir en permanence sur sa ligne. Dès lors que le détecteur de ligne gauche est actif, le robot dévie vers la gauche. Le robot se redresse vers la droite pour se remettre sur la piste et vice versa

Il est important d'écrire la condition du détecteur central en premier car cette condition est prioritaire sur les deux autres.

```
void loop()
{
    if (digitalRead(detecteur_central) == HIGH)
    {
        reculer();
        delay(250);
    }

    else if (digitalRead(detecteur_droit) == HIGH)
    {
        gauche();
    }

    else if (digitalRead(detecteur_gauche) == HIGH)
    {
        droite();
    }

    else
    {
        avancer(); // aller dans la sous-fonction avancer
    }
}
```




Dès lors qu'un des capteurs de ligne détecte une ligne noire, le robot recule puis vas dans la direction opposée à ce capteur.

Exemple : Si une ligne noire est détectée sur le capteur de ligne gauche, le robot recule puis tourne à droite avant de continuer son chemin.

```
void loop()
{


    if (digitalRead (detecteur_droit) == HIGH)
    {
        reculer();
        delay(200);
        gauche();
        delay(200);
    }

    else if (digitalRead (detecteur_gauche) == HIGH)
    {
        reculer();
        delay(200);
        droite();
        delay(200);
    }

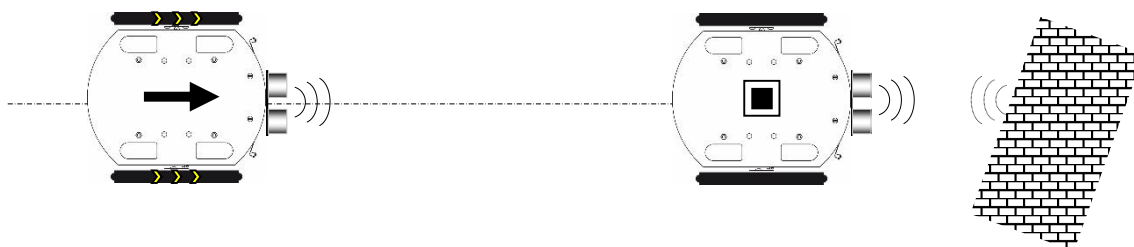
    else
    {
        avancer(); // aller dans la sous-fonction avancer
    }

} // Fin du programme principal
```

16. Fiche technique N°12 – U1-Proximate

But de l'application	Evoluer en ligne droite et s'arrêter à l'approche d'un obstacle
Notions de programmation abordées	Utilisation du module à ultrasons pour détecter un obstacle à distance
Programme associé	Code source Arduino Uno <i>U1-Proximate .ino</i> (Fichiers téléchargeables sur www.a4.fr)
Capteur mis en jeu	 Module capteur à ultrasons (détection d'obstacle à distance).

Illustration



Suggestions de modifications	Modifier la distance maximum d'arrêt du robot lors de la détection d'un obstacle
------------------------------	--



Dans un premier temps, il est obligatoire de déclarer la broche signal du télémètre à ultrasons.

```
const int ultrasons = 5;
```

Une variable type **long** doit être initialisé. Cette variable servira à stocker la valeur de la distance mesurée par le capteur à ultrasons.

```
long distance=0;
```

La broche signal du capteur à ultrasons doit ensuite être déclarée comme une sortie dans la fonction setup().

```
//paramétrage des pins en entrée/sortie
```

```
pinMode (ultrasons, OUTPUT); //le capteur Ultrason est un capteur spécial qui a besoin d'être défini comme une sortie
```

La sous-fonction qui gère le module ultrasons s'exécute sous cette forme :

```
long mesure_distance()// Sous fonction Mesure distance
```

```
{
    pinMode(ultrasons, OUTPUT);
    digitalWrite(ultrasons, LOW);
    delayMicroseconds(2);
    digitalWrite(ultrasons, HIGH);
    delayMicroseconds(10);
    digitalWrite(ultrasons, LOW);
    pinMode(ultrasons, INPUT);
    distance = pulseIn(ultrasons, HIGH)/58; // affectation de la distance dans la variable distance
    return (distance); // retour de la valeur distance dans le programme principal
}
```

Cette sous-fonction mesure une distance et la stocke dans la variable distance. La valeur de la variable distance est directement calculée en centimètre.

Le programme gérant l'arrêt du robot est une condition sur la variable distance.

```
//Boucle infinie
```

```
void loop()
```

```
{
```

```
distance = mesure_distance(); // aller dans la sous-fonction mesure_distance
```

```
// et renvoyer la valeur mesurée par le capteur ultrason dans la variable distance
```

```
if (distance<=10) //Si la distance est < 10 cm, le robot s'arrête si non il avance
```


```
{
    arret();
}
```

```
else
```

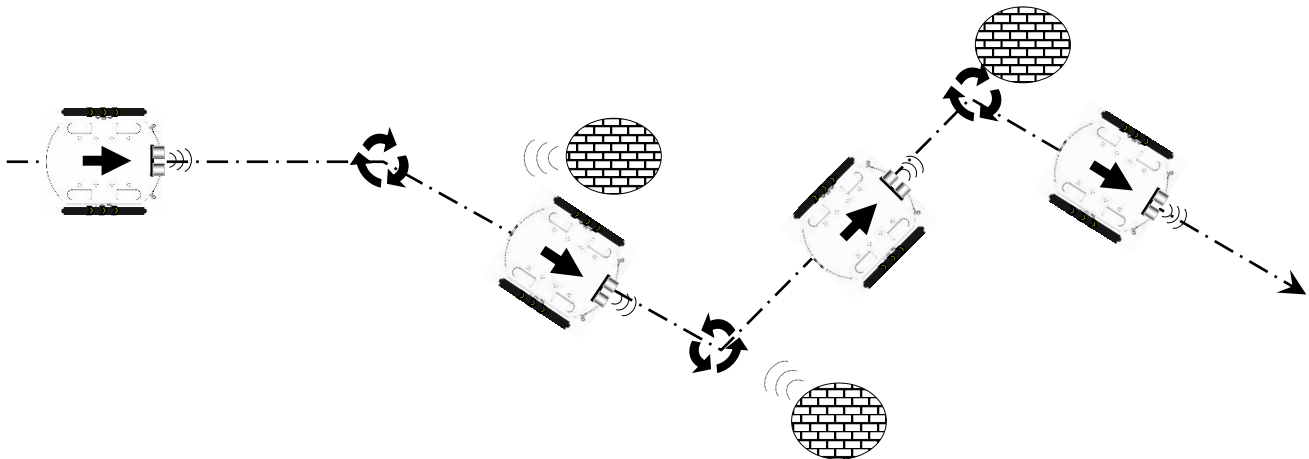
```
{
    avancer();
}
```

```
} // Fin du programme principal
```


17. Fiche technique N°13 – U2-Slalom

But de l'application	Evoluer en ligne droite et alterner un changement de direction à droite puis à gauche à l'approche d'un obstacle
Notions de programmation abordées	Utilisation d'une variable dans une condition
Programme associé	Code source Arduino Uno <i>U2-Slalom.ino</i> (Fichiers téléchargeables sur www.a4.fr)
Capteur mis en jeu	 Module capteur à ultrasons (détection d'obstacle à distance).

Illustration



Suggestion de modification	Modifier la distance maximum d'arrêt du robot lors de la détection d'un obstacle
----------------------------	--



Pour effectuer un changement de direction à chaque détection d'un obstacle, il est nécessaire de déclarer une variable qui selon sa valeur déterminera la direction du robot.

`byte i=0;` | `i` peut prendre la valeur 0 ou 1 d'où la déclaration de `i` en byte

Le programme contient une condition dans une condition.


```
//Boucle infinie
void loop()
{
    distance = mesure_distance(); // aller dans la sous-fonction mesure_distance et renvoyer la valeur mesurée
                                   //par le capteur ultrason dans la variable distance

    if (distance<=15) //Si la distance est < 15 cm, le robot évite l'obstacle si non il avance
    {
        if( i == 1)
        {
            droite();
            delay(200); // tourner à droite pendant 200 ms
            i = 0; // à la prochaine détection, le programme tournera à gauche
        }
        else if (i == 0)
        {
            gauche();
            delay(200); // tourner à gauche pendant 200 ms
            i = 1; // à la prochaine détection, le programme tournera à droite
        }
    }

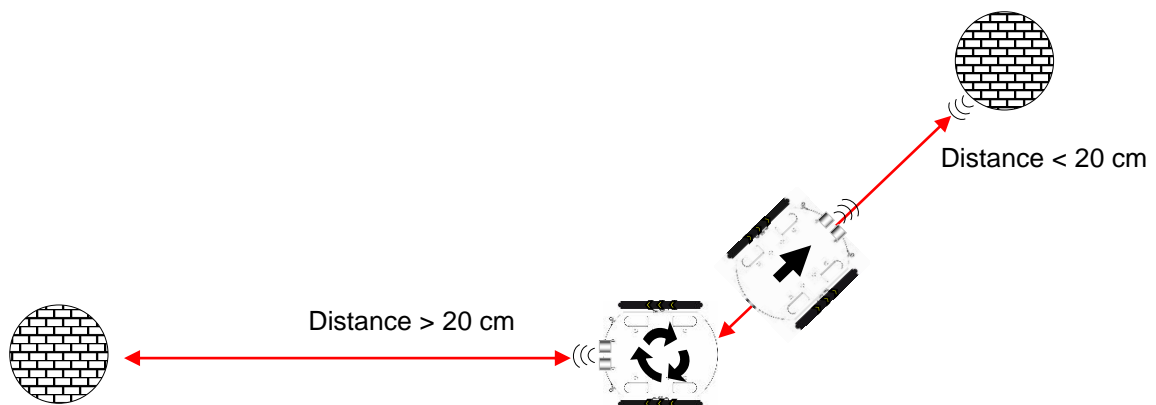
    else // si le capteur ne détecte pas de présence à moins de 15 cm, alors le robot avance
    {
        avancer();
    }
}

} // Fin du programme principal
```

18. Fiche technique N°14 – U3-Cible

But de l'application	Balayer une zone pour détecter la présence d'une cible située à une distance inférieure à 20 cm et se diriger vers elle
Notions de programmation abordées	Evoluer en ligne droite et alterner un changement de direction à droite puis à gauche à l'approche d'un obstacle
Programme associé	Code source Arduino Uno U3-Cible.ino (Fichiers téléchargeables sur www.a4.fr)
Capteur mis en jeu	 Module capteur à ultrasons (détection d'obstacle à distance).

Illustration



Suggestion de modification	Permettre au robot de s'arrêter lorsqu'il est à proximité de l'obstacle
----------------------------	---



Robot CoDa scrute et tourne sur lui-même, s'il aperçoit un objet à moins de 20 cm, il avance directement dessus.

```
//Boucle infinie
void loop()
{

distance = mesure_distance(); // aller dans la sous-fonction mesure_distance et renvoyer la valeur mesurée
                               // par le capteur ultrason dans la variable distance

if (distance<=20) //Si la distance est < 20 cm, le robot avance sur la cible
{
    avancer();
}

else // si le capteur ne détecte pas de présence à moins de 20 cm, alors il tourne sur place
{
    droite();
}

} // Fin du programme principal
```