



Rapport TER

---

## Robot avec détection d'obstacles

---

*Auteurs :*

BENSOUSSAN Chloé  
BOUKOU Grace  
GRÉAU Alexandre  
WILHELM Andreina

*Responsables :*

M. FORMENTI  
Mme. PELLEAU

Juin 2018

# Table des matières

<b>1</b>	<b>Introduction</b>	<b>2</b>
<b>2</b>	<b>État de l'art</b>	<b>2</b>
<b>3</b>	<b>Travail effectué</b>	<b>4</b>
3.1	Hardware . . . . .	4
3.1.1	Conception des pièces . . . . .	5
3.1.2	Difficultés rencontrées à la conception . . . . .	9
3.2	Software . . . . .	10
3.2.1	Dummy Scrappy . . . . .	10
3.2.2	A spark of intelligence . . . . .	11
3.2.3	Follow the leader . . . . .	13
3.3	Deuxième Robot . . . . .	13
3.4	Matériel utilisé . . . . .	14
<b>4</b>	<b>Conclusion</b>	<b>14</b>
<b>5</b>	<b>Perspectives d'améliorations</b>	<b>15</b>
<b>6</b>	<b>Remerciements</b>	<b>15</b>
<b>7</b>	<b>Annexe (Tutoriel)</b>	<b>16</b>
7.1	Composants pour Scrappy . . . . .	16
7.2	Assemblage . . . . .	16
7.3	Les branchements . . . . .	16

# 1 Introduction

Ce TER a pour objectif l'amélioration d'un robot explorateur capable de détecter et d'éviter des obstacles. La version prise en main était réalisée avec trois capteurs, chacun calculant les distances entre le robot et un obstacle devant lui. Cependant, cette version rencontrait quelques problèmes quant à la détection d'obstacles inclinés, aux boucles infinies dans un cul de sac et à sa structure bancale. Notre cahier des charges comprends le fait de détecter les obstacles présentant sur son chemin et de les éviter sans erreurs lors de son exploration. Il est ajouté à cela le rendu d'un robot solide, mais dont les éléments puissent facilement être remplacées. De plus, une communication avec un deuxième robot est également envisagée afin de composer une formation entre eux.

Notre groupe est composé de BENSOUSSAN Chloé, BOUKOU Grace, GRÉAU Alexandre et WILHELM Andreina, et a été encadré par Madame PELLEAU, ainsi que Monsieur FORMENTI.

Lors de ce projet, toutes nos idées ont été discutées en groupe et en demandant quelques conseils de nos tuteurs. La mise en oeuvre de la partie programmation a été réalisée par Chloé et Andreina, la partie hardware réalisée par Grace et Alexandre.

Dans ce rapport nous commencerons par décrire le point de départ du projet ainsi que les composants utilisés, puis nous expliquerons les travaux que nous avons effectués avant de conclure et d'évoquer les différentes améliorations possibles.

## 2 État de l'art

Le robot de départ comprenait trois capteurs à ultrasons placés en arc de cercle à égale distance du centre du robot. Chacun des capteurs utilise des ultrasons pour déterminer la distance d'un objet se trouvant face à lui. Son fonctionnement n'est influencé ni par la lumière du soleil, ni par les matériaux sombres mais est sensible aux matériaux absorbant les ondes. La température peut aussi influencer la propagation des ondes mais les capteurs que nous utilisons rectifient ces petites variations au moyen d'une puce intégrée.

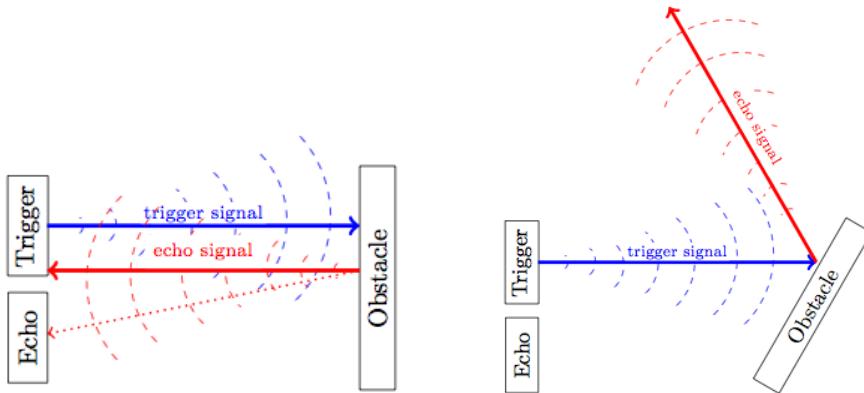


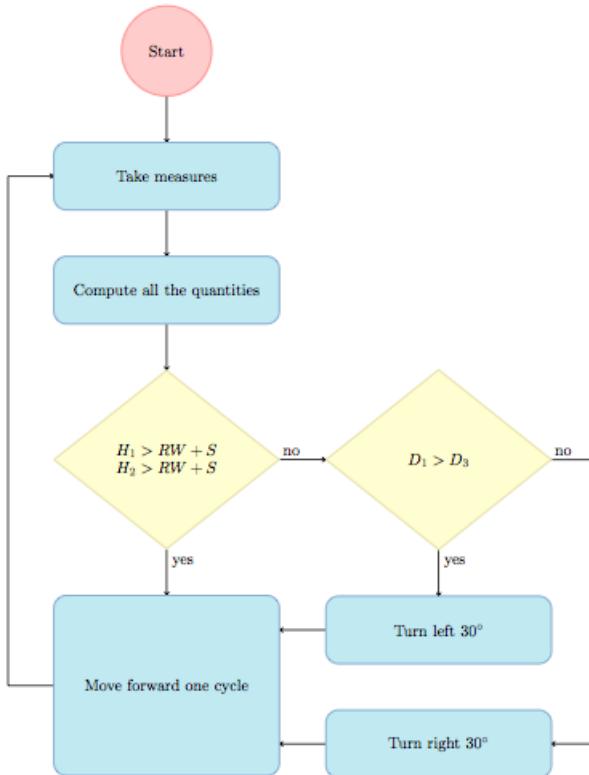
FIGURE 1 – Fonctionnement des capteurs et les problèmes liés.

Ce système basé sur l'écho, comme expliqué à la figure 1, offre une excellente plage de détection sans contact, avec des mesures précises et stables. En revanche, il rencontre des difficultés lorsque l'obstacle se trouve incliné par rapport au capteur; cette restriction donne naissance à la première problématique du sujet : comment détecter ces obstacles particuliers.

Afin de remédier à ces inconvénients, diverses améliorations avaient été envisagées :

- Utilisation de six senseurs au lieu de trois, avec mise en place d'un protocole efficace de communication entre les senseurs et l'Arduino.
- Utilisation de deux Arduino, l'un pour les moteurs, l'autre pour les senseurs, avec la mise en place d'un protocole efficace de communication entre les Arduino.
- Conception d'un châssis en plexiglas pour le positionnement optimal des composants.
- Modélisation et impression 3D du châssis.

Le diagramme suivant résume les caractéristiques du logiciel embarqué sur le robot que nous a été confié.



### 3 Travail effectué

La première étape de ce projet fut de prendre en main la technologie existante, puis d'établir un plan de route en se répartissant les tâches. Les premières réunions furent aussi l'occasion d'émettre toutes nos idées afin d'identifier les problèmes et de débattre entre nous des meilleures solutions, sans que cela ne soit définitif.

La mission étant de construire un robot et son algorithme, deux axes de travail se sont dégagés. Nous avons donc constitué deux groupes : l'un s'occupera de la partie hardware, l'autre de l'implémentation logicielle.

#### 3.1 Hardware

Grâce au tout récent FabLab, nous avions à notre disposition une imprimante 3D. Nous pouvions donc créer une maquette virtuelle de notre robot puis imprimer chaque pièce une à une. Nous les avons dessinées avec pour soucis de pouvoir remplacer une pièce cassée ou un composant en panne facilement. Par exemple, si l'un des capteurs venait à ne plus fonctionner correctement, il suffit de déclipser le couvercle, débrancher le capteur et procéder à son remplacement. De même, si le support actuel ne convient pas à la nouvelle pièce il suffit de redessiner un nouveau support en gardant le même système de fixation et de le substituer à l'ancien.

Une des problématiques de ce sujet était de résoudre le problème de perte de signal face à un obstacle incliné. Plusieurs solutions ont été imaginées, comme ajouter des "antennes" tel un somnambule, ou ajouter une "tête" pivotante au robot, mais nous avons préféré rajouter un capteur à l'avant pour élargir la surface de réception et ainsi, avoir plus de chance de récupérer le signal. En effet, les deux capteurs centraux seront vus par l'Arduino comme un seul grand capteur. Cette paire est disposée en V afin de voir les objets inclinés, droits.

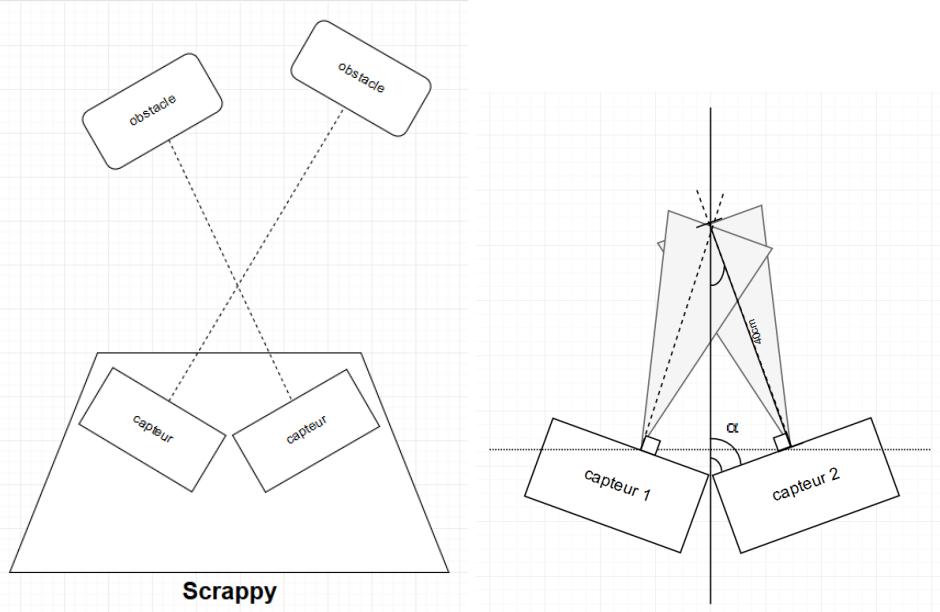


FIGURE 2 – Le dispositif en V et ses inconvénients.

Cependant cette disposition entraîne des interférences entre capteurs. Puisque les capteurs utilisent la même fréquence, les deux ondes des deux émetteurs se détruisent mutuellement. L'angle de départ

initialement calculé à 60 degrés pour une distance de sécurité de 40 cm, a dû être augmenté à 75 degrés. Nous avons de plus placé les transmetteurs côté à côté pour maximiser la largeur de réception et éviter qu'un signal émis soit capté directement par le récepteur de son compère.

Le robot ayant ensuite besoin de pouvoir "voir" sur ses flancs, nous avons rajouté deux capteurs sur les côtés, debout, pour réduire la place prise par ce composant et supporter un éventuel capot.

Toujours dans l'optique de pouvoir remplacer facilement un composant, chaque capteur s'encastre dans une boîte sans pour autant y être fixé, et y est maintenu par un couvercle.

Le robot est propulsé par deux roues motrices fixées à l'arrière et une roue pivot, encastrée dans une coupe, a été ajoutée à l'avant.

Voici une vue d'ensemble du robot principal.

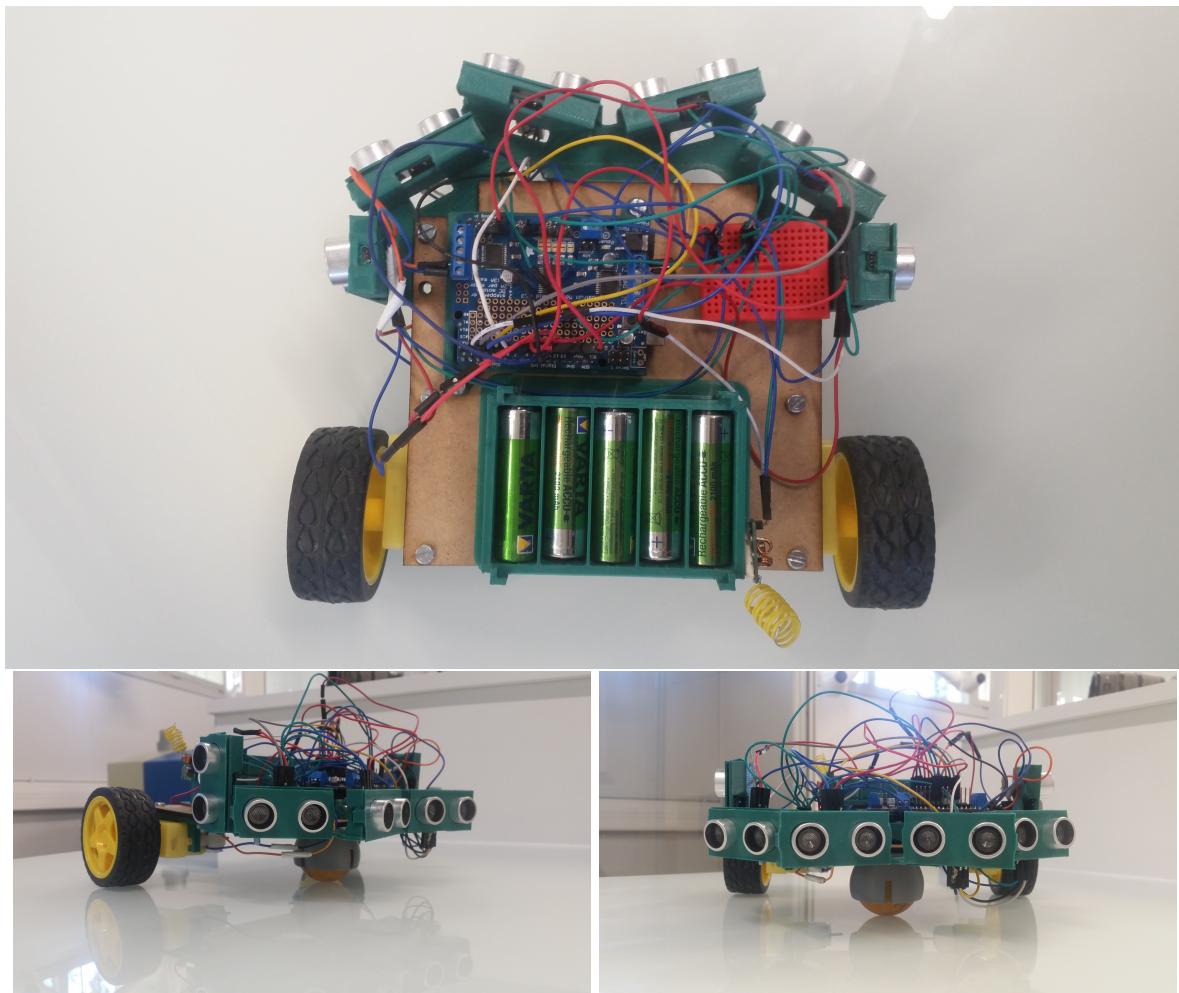


FIGURE 3 – Trois images du robot principal.

### 3.1.1 Conception des pièces

Chaque pièce a été dessinée sur le logiciel Tinkercad avant d'être imprimée. Il a fallu pour cela mesurer les composants un à un afin de modéliser convenablement chaque pièce.

Voici une présentation sommaire des supports de chaque composant du robot principal.

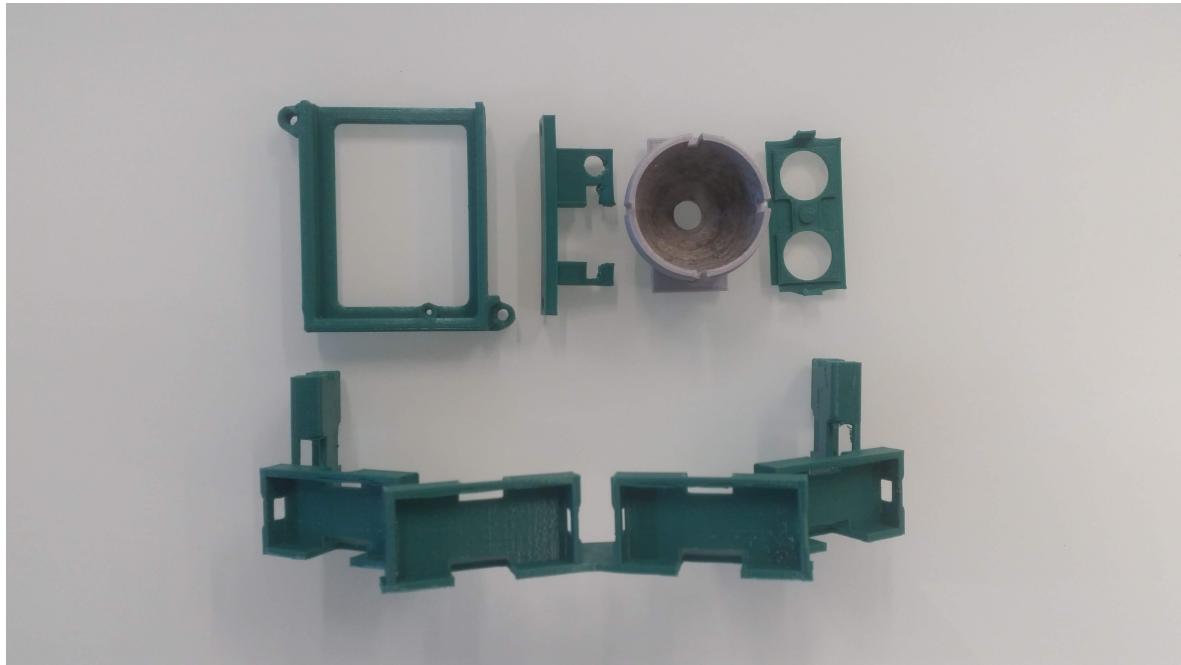


FIGURE 4 – Ensemble des différents composants créés.

Nous pouvons voir en bas de cette photo le "nez" du robot. Chaque "boîte" est conçue pour accueillir un capteur et ne laisser dépasser que les fils de branchement, tout en respectant leurs dispositions calculée précédemment.

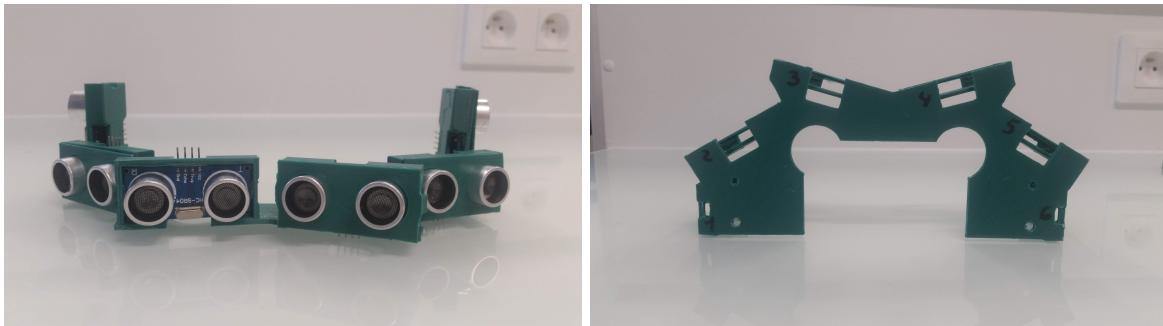


FIGURE 5 – Disposition des capteurs.

Pour des raisons esthétiques mais aussi pour protéger les capteurs, ces boîtes sont recouvertes par des capuchons clipsables (figure 6).

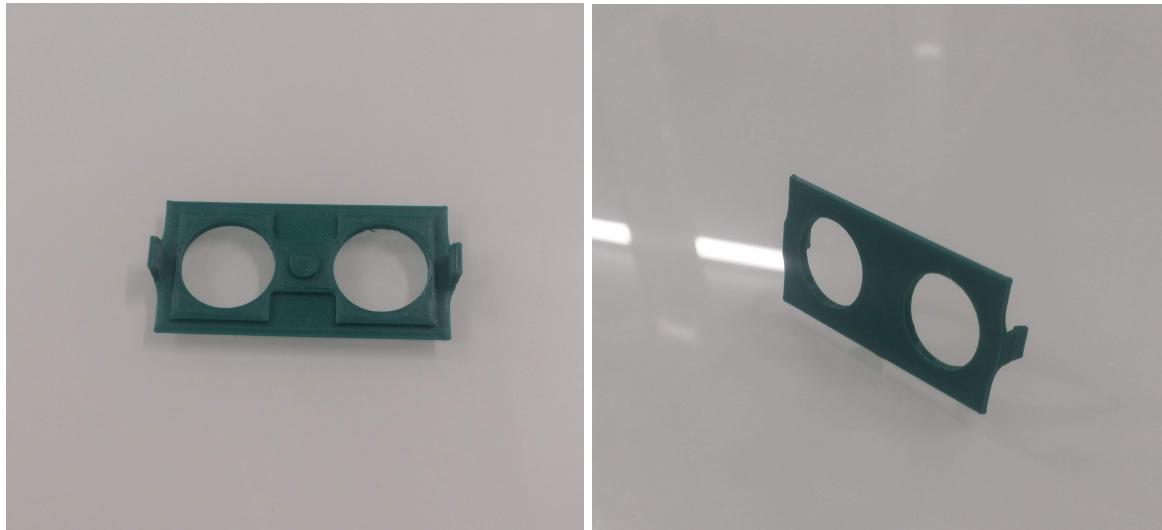


FIGURE 6 – Capuchon protecteur.

Les capteurs sont reliés à l'Arduino qui repose au centre du robot. Celle-ci est glissée dans la rainure de son étui. Elle peut ensuite y être vissée (orifice intérieur gauche) pour plus de sûreté.



FIGURE 7 – Support de la carte Arduino (elle se glisse ici de gauche à droite).

L'Arduino est le cerveau du robot, c'est elle qui transmet les directives à chacun des composants suivant l'algorithme qu'elle contient et les données reçues.

Si l'Arduino en est le cerveau, la batterie de cinq piles en est le coeur. Ces cinq piles sont logées dans un support pouvant être empilé si le robot venait à être trop gourmand et nécessiterait cinq piles supplémentaires. Nous avons ensuite réalisé le câblage permettant au courant de circuler (figure 8).



FIGURE 8 – Support destiné à accueillir cinq piles AA.

Le robot est propulsé par deux moteurs, montés sur des étriers grâce à la même partie du composant qui fixe les moteurs en lui-même à la roue qu'il entraîne, sans gêner l'arbre (figure 9).



FIGURE 9 – Fixation des moteurs.

Une troisième roue est nécessaire pour équilibrer le robot. Comme elle doit pouvoir tourner sur tout les axes, l'idée d'encastrer une balle dans une coupe s'est très vite imposée.



FIGURE 10 – Voici la roue dite pivot, réceptacle d'une balle de ping-pong.

Le dispositif de communication est quant à lui, collé aux batteries avec de la Patafix pour optimiser l'espace mais aussi pour ne pas perturber la communication.

Le châssis a été conçu sur Tinkercad puis découpé au laser.

### 3.1.2 Difficultés rencontrées à la conception

Une des grandes difficultés rencontrées a été le temps pris par les impressions de ces composants. A raison de 6 heures pour le nez du robot, 3 heures pour le socle de piles et 20 minutes pour les petits capuchons clipsables, la moindre erreur d'impression ou erreur de notre part reportait l'assemblage de quelques heures.

Le matériel a aussi parfois su être capricieux, certains câbles, notamment au niveau du moteur et de l'antenne de communication produisaient de faux contacts. Nous les avons donc soudés.

Nous avons également dû prendre en compte les limitations énergétiques du modèle, comme la répartition du courant et de son intensité, et dessiner le plan des branchements en conséquence.

Nous avons vite réalisé l'importance de ce dernier point lorsque nos composants ont commencé à fonctionner anormalement au fur et à mesure de leur ajout : des capteurs qui ne fonctionnaient plus, une communication erronée, des moteurs au ralenti...

## 3.2 Software

Nous avons amélioré l'algorithme d'exploration en plusieurs passes.

### 3.2.1 Dummy Scrappy

Tout d'abord nous avons implémenté un premier algorithme évitant tout obstacle. Cet algorithme utilise les mesures des quatre capteurs placés en M (voir figure 11) pour prendre une décision. Le robot calcule la distance devant chacun de ses capteurs de façon périodique, détermine la zone la plus dégagée en comparant les distances mesurées, puis s'y dirige.

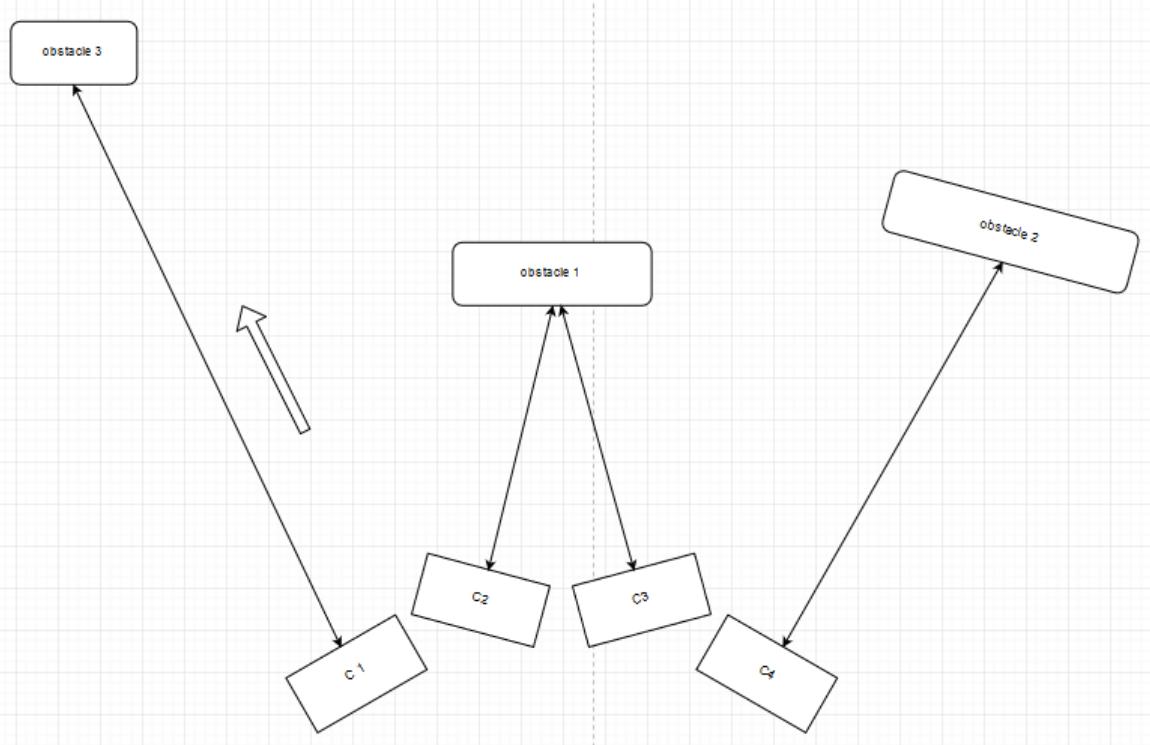


FIGURE 11 – Prise de décision du robot.

Nous avons défini une distance dite de sécurité en nous basant sur la taille et la vitesse de déplacement du robot. Elle représente la distance minimale nécessaire pour avoir le temps de rectifier sa trajectoire sans percuter l'obstacle.

Dès que les capteurs centraux détectent un objet situé dans sa zone de sécurité, le robot analyse ses alentours. Il compare ensuite les distances mesurées par les capteurs latéraux et se réoriente vers la zone la plus dégagée. Il effectue la même action même s'il ne trouve aucun espace libre. Dans ce cas, il tournera jusqu'à détection d'une zone accessible. Néanmoins, si le robot croise un objet trop près de lui, il reculera immédiatement afin de rétablir la distance de sécurité, avant de manoeuvrer pour éviter cet obstacle.

Le code initial rencontrait des problèmes lors de ce dernier point. Le robot reculait pour maintenir la distance de sécurité, mais dès cette distance atteinte, il détectait un espace libre devant lui qui le faisait avancer. Cette répétition le faisait rentrer dans une boucle avant-arrière infinie. Nous avons réussi à résoudre ce problème en changeant la direction du robot systématiquement après une marche arrière.

Cependant, comme le robot choisissait toujours la direction ayant le plus d'espace, il pouvait s'orienter à droite, puis à gauche, puis encore à droite sans jamais s'arrêter. Nous avons traité ce cas en ne faisant conduire le robot que d'un seul côté. Ainsi, le robot peut effectuer des tours complets à la recherche de zone libre.

Voici le pseudo-code de la fonction `explore()` :

---

**Algorithm 1** Choisir direction

---

```

1: if capteurDevant > safetyDistance + tailleRobot then
2:   return avancer
3: else if capteurDevant > safetyDistance then
4:   if capteurGauche > capteurDroit then
5:     return tourner à gauche
6:   else
7:     return tourner à droite
8:   end if
9: else
10:  return reculer
11: end if

```

---

### 3.2.2 A spark of intelligence

Nous avons ensuite voulu remplacer ces déplacements aléatoires par une véritable exploration. Notre deuxième algorithme consiste à longer un obstacle pendant un laps de temps donné, pour ensuite sortir de ce cycle en se redirigeant vers une zone libre.

Cet algorithme nécessite un contact permanent avec l'obstacle se trouvant à droite ou à gauche du robot. Pour cela, nous avons choisi d'ajouter deux nouveaux capteurs de chaque côté du robot. Grâce à ces derniers, le robot est capable de "voir" ses flancs et se maintenir à une distance de sécurité de son obstacle. S'il venait à le perdre de vue, il tournerait dans sa direction pour le retrouver. Au bout d'un certain temps (temps moyen pour faire entre un et deux tours de sac), le robot marquera un temps d'arrêt de quelques ticks, avant d'entamer une nouvelle exploration dans l'espace où il se trouve. Il se dirigera vers la direction inverse de l'obstacle précédent pour s'en éloigner débutant une nouvelle aventure.

Le code pour cet algorithme est plus complexe que le précédent. Voici son pseudo-code :

▷ *objectDetected* représente le côté où se trouve l'obstacle. S'il n'y en a pas, est égal à 0.

▷ *marge* représente la distance de sécurité.

---

**Algorithm 2** Longer l'obstacle à gauche

---

```

1: if capteurGauche > safetyDistance then           ▷ Le robot a dépassé l'objet ou est trop loin.
2:   return tourner à gauche
3: else if capteurGauche < safetyDistance - marge or capteurDevant < safetyDistance then
4:   return tourner à droite                         ▷ Si il est trop près de l'objet ou trouve un objet devant lui.
5: else
6:   return avancer
8: end if

```

---

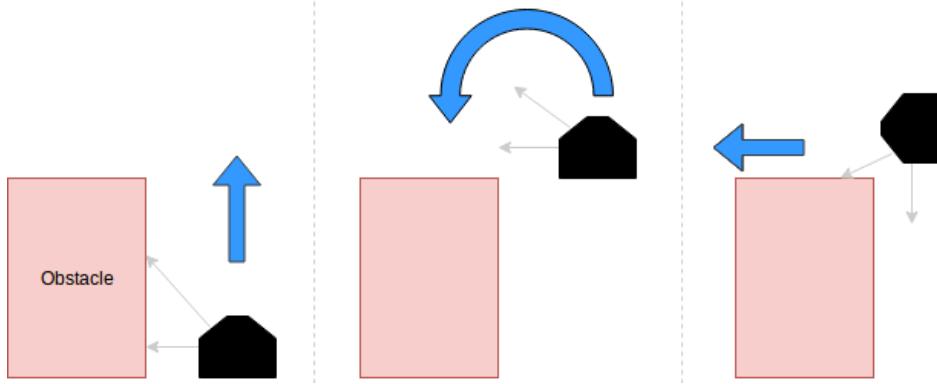


FIGURE 12 – Dépassement de l'objet.

Cet algorithme est schématisé par les deux figures 12 et 13. Le robot suit un objet qui se trouve à sa gauche. Dès lors il avance un peu trop, il le perd de vue. Dans ce cas de figure, il s'orientera dans une direction afin de le retrouver (figure 12). Ce schéma représente le premier **if** de l'algorithme 2.

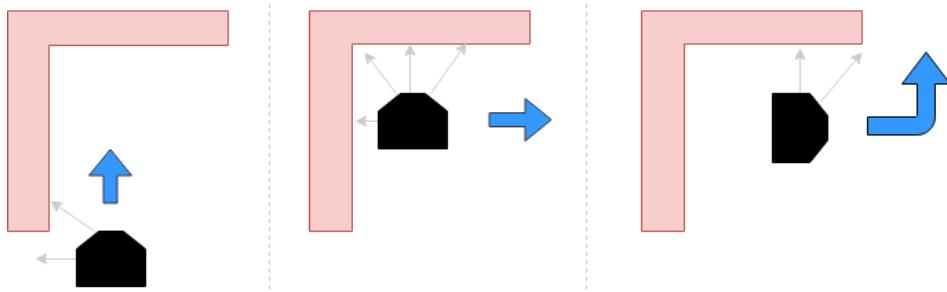


FIGURE 13 – Présente d'un objet sur deux côtés.

La figure 13 représente le deuxième **if** de l'algorithme. Le robot est entrain de suivre un objet mais détecte un obstacle devant-lui. Il s'orientera dans la direction inverse de l'objet pour pouvoir continuer son chemin. Lorsque l'obstacle se trouve du côté droit du robot, nous effectuons ces mêmes procédures en inversant les directions.

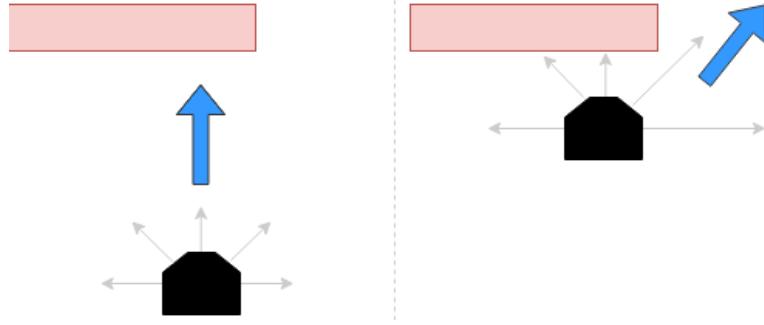


FIGURE 14 – Choix de direction lorsqu'un objet se trouve devant lui.

Le robot ne suivant aucun objet se dirige tout droit. S'il détecte un objet devant lui, il comparera ses espaces latéraux pour déterminer sa nouvelle direction (figure 14). Ce choix est implémenté grâce à l'algorithme 3.

---

**Algorithm 3** Détection d'obstacle

---

```
1: if capteurGauche < safetyDistance then                                ▷ Objet trouvé à gauche.  
2:     objetDetected = gauche  
3: else if capteurDroit < safetyDistance then                            ▷ Objet trouvé à droite.  
4:     objetDetected = droite  
5: else if capteurAvant < safetyDistance then                           ▷ Objet trouvé à l'avant.  
6:    ▷ Comparer et prendre la direction ayant le plus d'espace.  
7:     if capteurGauche > capteurDroit then  
8:         objectDetected = droite  
9:         return tourner à gauche  
10:    else  
11:        objectDetected = gauche  
12:        return tourner à droite  
13:    end if  
14: end if  
15: return avancer
```

---

### 3.2.3 Follow the leader

Nous avons aussi implémenté un dernier algorithme mettant en œuvre une communication entre deux robots. Le robot maître utilise le premier algorithme qui transmet sa direction à son suiveur. Le robot junior se déplace quant à lui en suivant les données reçues. Il vérifie tout de même s'il a le champs libre via son unique capteur. Dans le cas où il ne recevrait plus de données (en raison d'une erreur technique), il effectuera la dernière direction reçue jusqu'à nouvel ordre.

Cependant, la transmission des données étant instantanée, les deux robots tournent au même moment. Ils se retrouvent donc côté à côté et effectuerons par la suite des mouvements synchronisés.

## 3.3 Deuxième Robot

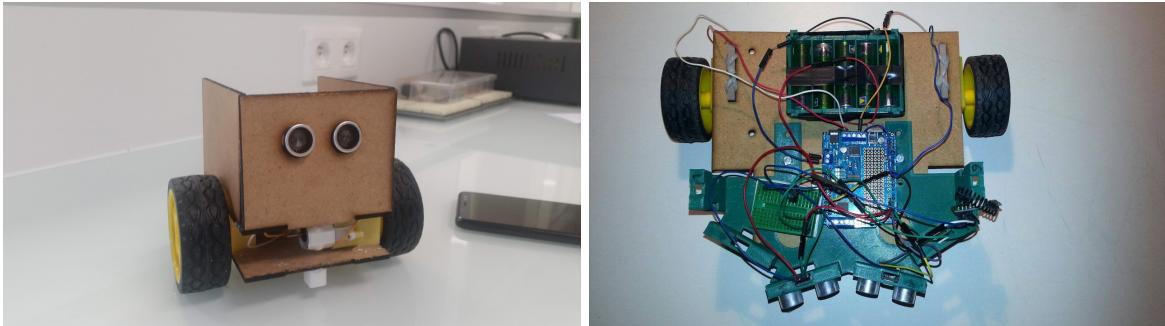


FIGURE 15 – À gauche la première version du robot junior, et son évolution à droite.

Le deuxième robot et son algorithme sont à l'image d'un malvoyant et son guide. Il ne fait que suivre les instructions que le guide lui transmet, avec en plus un capteur en guise de sécurité tel le bâton de l'aveugle. Il peut ainsi s'arrêter si un obstacle venait à surgir devant le robot, mais aussi à détecter son aîné et se maintenir à une certaine distance de celui-ci.

Nous avions au départ construit un robot de petite taille n'ayant qu'une carte Arduino, sa batterie, un capteur, et un dispositif de communication. Cependant, lorsqu'il reçoit une instruction pour tourner,

celui-ci effectue la même rotation que son maître en termes de tours, mais, étant beaucoup plus petit, l'angle de rotation est beaucoup plus grand. C'est pourquoi nous avons construit une deuxième version ayant la même taille que son maître.

Par manque de temps, la deuxième version du robot junior est un recyclage d'une ancienne version du premier robot, les deux robots ont donc presque la même taille, ce qui limite l'erreur de rotation.

### 3.4 Matériel utilisé

Avec la supervision de nos tuteurs nous avons pu avoir accès au tout récent FabLab, comprenant une imprimante 3D, la UltiMaker3 à disposition pour imprimer des pièces conçues avec le site Tinkercad.

Nous avons aussi utilisé une découpeuse laser pour la création des chassis des deux robots.

Les données des capteurs sont traitées par une carte Arduino, la gestion des moteurs et des roues étant faites par une carte spécifique de marque Adafruit.

Le développement a été réalisé en langage C en utilisant l'IDE Arduino. Nous avons utilisé plusieurs librairies pour manipuler les différents composants. Les références précises sont fournies en Annexe, nous détaillons ici leur utilisation.

Pour le déplacement du robot, la librairie `AdaFruit_MotorShield` a été utilisée, car elle permet de contrôler les moteurs DC afin de faire avancer, reculer, arrêter et changer la vitesse des roues.

En ce qui concerne la communication entre robots, la librairie `VirtualWire` avait été utilisée lors de la première phase, car elle fournissait toutes les fonctionnalités nécessaires pour envoyer et recevoir des messages courts par radio-fréquence. Une fois implémentée, la communication fonctionnait parfaitement de manière isolée : le maître envoyait sa direction à son prochain, qui le recevait bien. Cependant, en effectuant des tests avec tous les composants (c'est à dire avec les capteurs et les moteurs), nous remarquions que le suivant ne recevait plus les messages.

Lors de l'exécution de tests plus rigoureux et de recherches plus approfondies sur ce problème, nous avons constaté que cette librairie est incompatible avec la librairie utilisée pour les roues. C'est pourquoi notre deuxième robot était erroné : soit il recevait le message, soit il bougeait les roues sans jamais effectuer les deux actions simultanément. Afin de résoudre ce problème, nous avons trouvé une autre librairie, appelée `RadioHead`, offrant les mêmes fonctionnalités que la précédente et cette fois, bien compatible avec les autres librairies.

Lors de l'utilisation du troisième algorithme, le robot maître envoie un seul entier à la fois. Chaque direction est affectée à un entier pour faciliter la manœuvre et la synchronisation des mouvements entre les deux robots.

Si vous souhaitez construire le même robot chez vous, rendez-vous en annexe ou sur le site GitHub où vous trouverez le tutoriel détaillé des assemblages pas à pas de notre création. Vous aurez également accès à tous les plans des composants 3D et les algorithmes développés décrits plus tôt.

## 4 Conclusion

Ce sujet nous a permis d'explorer une branche de l'informatique plus concrète. Beaucoup de code d'une part, avec toute la partie programmation de l'Arduino, mais aussi beaucoup d'aspects plus "concrets" que l'informatique académique. Nous avons pu toucher à d'autres domaines comme l'électronique, pour déterminer l'intensité nécessaire au bon fonctionnement de l'ensemble des composants, la soudure, et la conception des pièces en trois dimensions.

Cette partie de conception nous a confrontée à la différence difficilement prévisible entre théorie et réalité, avec souvent pour conséquence de devoir redessiner et réimprimer entièrement un support,

nous coûtant de précieuses heures. Ce projet a aussi demandé une bonne organisation et une bonne communication avec nos tuteurs mais aussi et surtout au sein de l'équipe, où il était important que chacun émette ses idées sans gêne et se sente impliqué.

Nous avons conçu un robot facilement maintenable et améliorable, capable de non seulement circuler, mais aussi d'explorer une zone selon les différents algorithmes. Nous avons également créé un deuxième robot suivant les ordres transmis par radio de son aîné. Malgré tout, quelques améliorations peuvent être envisagées.

Nous avons également mis à disposition un tutoriel à destination des plus curieux et de nos successeurs, afin de leur donner les marches à suivre pour reproduire notre robot.

## 5 Perspectives d'améliorations

Par manque de temps, plusieurs idées n'ont pas pu être mises en œuvre. Différentes améliorations sont donc possibles :

- Un algorithme explorateur plus efficace se basant sur le *mapping* : création en mémoire d'une carte de la zone à explorer, sous forme de matrice comptant le nombre de passages sur chaque case. Le robot créerait d'abord une carte de ses alentours proches, puis circulerait sur les cases non parcourues.
- Le design et le code du deuxième robot peuvent être repensé pour, à l'image de son aîné, accueillir d'autres composants et élargir ses possibilités d'action.
- Le robot actuel ne détecte que les obstacles se trouvant sur le même champs que lui. Peut-être serait-il intéressant de le rendre capable de détecter des trous ou des obstacles en hauteur.
- Nous pouvons également imaginer une exploration de la pièce divisée entre plusieurs robots communiquant entre eux.
- Améliorer l'esthétique de l'ensemble en dessinant un couvercle ou en ajoutant différents composants comme des LED (clignotants)... Attention cependant à la répartition de la tension et de l'intensité.

Si une des modifications citées ci-dessus oblige un changement de support, il suffit de le redessiner en gardant le système de fixations que la pièce a remplacer.

## 6 Remerciements

Nous tenons à remercier les personnes et organisations qui nous ont aidé dans la réalisation de ce projet. En premier lieu, nous remercions nos tuteurs pour leurs conseils et le temps qu'ils nous ont consacré, ainsi que M. MONTICELLI qui nous a aimablement permis d'utiliser une découpeuse laser.

Nous exprimons notre gratitude au laboratoire I3S pour avoir mis à disposition une salle à plusieurs reprises tout au long du TER. Nous souhaitons également remercier l'ensemble du département informatique pour leur patience, pour nous avoir fourni tout le matériel dont nous avions besoin et l'intérêt qu'ils ont porté au robot tout au long de sa création.

## 7 Annexe (Tutoriel)

### 7.1 Composants pour Scrappy

- |   |   |
|---|---|
| — 1 Arduino UNO<br>— 1 Shield Motor AdaFruit<br>— 2 moteurs avec les roues adaptées<br>— 1 roue libre<br>— 6 capteurs ultrasonique sensor HC-SR04 | — 1 module émetteur sans fils 433Mhz<br>— 1 boîtier pour 5 piles AA (1,2 V)<br>— des câbles jumper<br>— 1 mini breadboard<br>— 1 châssis (plaqué et le 'nez' pour capteurs) |
|---|---|

Pour réaliser le robot Scrappy junior, vous aurez besoin des mêmes composants en ne préparant qu'un seul capteur ultrasonique et un module récepteur à la place de l'émetteur.

### 7.2 Assemblage

Commencez par visser les moteurs et la roue pivot sur le châssis. Insérez les capteurs dans leur emplacements puis placez la carte Arduino, la breadboard et les piles sur le support. Une fois le tout fixé, passons maintenant aux branchements de ces éléments.

### 7.3 Les branchements

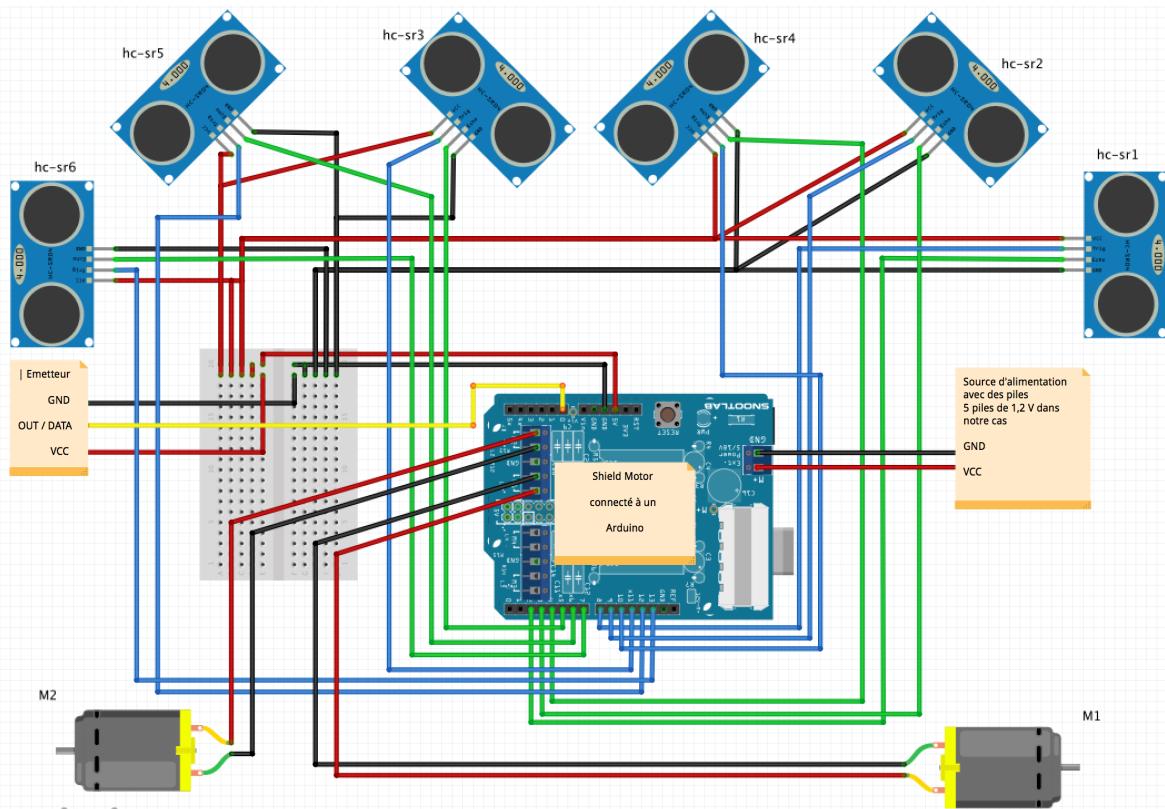


FIGURE 16 – Circuit du robot maître.

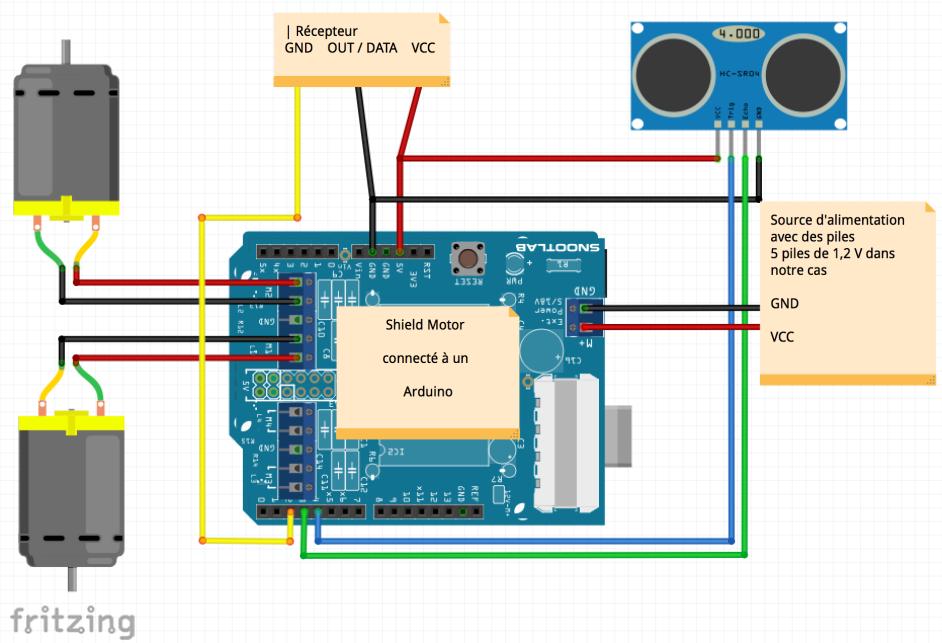


FIGURE 17 – Circuit du robot esclave.

Les circuits sont correctement décrits dans les figures 16 et 17. Nous conseillons de commencer par brancher les moteurs, les connecteurs GND (en noir) et VCC (en rouge). Continuez avec les branchements de `trig` (en bleu) et `echo` (en vert) des capteurs ultrasonique. Pour les branchements de Scrappy, les capteurs sont numérotés dans l'ordre de droite à gauche ; leur `echo` correspondent aux pins *numéro*+1 et les `trig` aux pins *numéro*+7. Terminez par le branchement de l'émetteur pour enfin alimenter le tout, soit par port USB, soit sur batterie. Si les roues tournent à l'envers, inverser les câbles des moteurs afin d'inverser la polarité.

Amusez-vous bien !

## Références

- [1] Lien vers le GitHub du projet :  
<https://github.com/master1-ififi-semestre2/TER>

### Lien des vidéos :

- [2] Algorithme 1 :  
[https://drive.google.com/open?id=1\\_9H6Q1YPMo7iLDxHySAQrTVMImpq7Iah](https://drive.google.com/open?id=1_9H6Q1YPMo7iLDxHySAQrTVMImpq7Iah)  
<https://drive.google.com/open?id=1o61uti5SyZaFpk5Q8PKH7WBNsOEiu68U>
- [3] Algorithme 2 :  
[https://drive.google.com/open?id=1nRt\\_Izcs40Q7nmCkDLtQk9FvdONo01NS](https://drive.google.com/open?id=1nRt_Izcs40Q7nmCkDLtQk9FvdONo01NS)

### Matériel utilisé :

- [4] Information sur la carte Arduino utilisé :  
<https://www.arduino.cc/en/Main/ArduinoBoardUno>
- [5] Information sur le Shield Motor :  
<https://www.adafruit.com/product/1438>
- [6] Information sur les Sensors :  
<https://randomnerdtutorials.com/complete-guide-for-ultrasonic-sensor-hc-sr04/>
- [7] Site de Tinkercad pour la modélisation d'objets en 3D :  
<https://www.tinkercad.com/>
- [8] Site de Cura pour générer le code pour l'imprimante :  
<https://ultimaker.com/en/products/ultimaker-cura-software>
- [9] L'imprimante 3D utilisée :  
<https://ultimaker.com/>

### Librairies utilisées :

- [10] Librairie AdaFruit MotorShield :  
<https://learn.adafruit.com/adafruit-motor-shield/library-install>
- [11] Librairie AdaFruit Servo Driver :  
<https://github.com/adafruit/Adafruit-PWM-Servo-Driver-Library>
- [12] Librairie RadioHead :  
<http://www.airspayce.com/mikem/arduino/RadioHead/>