

**Rapport technique**

**Robot piloté à distance**

**Rédigé par :** Flora Silberzan, Samuel Gerente, Etienne Loiselet, Florian Motyl, Gregoire Guyon

**Encadré par l’équipe du LEnsE**

# Table des matières

[Table des matières 2](#_Toc100166786)

[1 Introduction 3](#_Toc100166787)

[2 Réponse au cahier des charges 3](#_Toc100166788)

[3 Guide d’utilisateur 4](#_Toc100166789)

[4 Communication sans fil 4](#_Toc100166790)

[5 Branchements et câblage 5](#_Toc100166791)

[5.1 Ordinateur – Nucléo 1 – Module RF 5](#_Toc100166792)

[5.2 Module RF - Nucléo 2 (robot) 5](#_Toc100166793)

[6 Motorisation du robot 5](#_Toc100166794)

[6.1 Déplacement du robot 5](#_Toc100166795)

[6.2 Etalonnage de la distance et des rotations 5](#_Toc100166796)

[6.3 Asservissement des moteurs 6](#_Toc100166797)

[7 Système de coordonnées 7](#_Toc100166798)

[8 Difficultés rencontrées 8](#_Toc100166799)

[9 Idées pour le futur 8](#_Toc100166800)

# Introduction

Ce projet PROTIS a pour objectif de créer les différents modules et codes nécessaires pour contrôler un robot à distance. L’objectif est que le robot soit capable d’aller d’un point A à un point B avec éventuellement des étapes intermédiaires dans un hangar fictif (créé exprès pour le projet). Pour faciliter le déplacement du robot au sein du hangar, il est important que la communication entre le robot et l'ordinateur se fasse sans fil. Enfin pour que n’importe qui puisse utiliser le robot, nous avons mis en place une interface graphique simple d’utilisation.

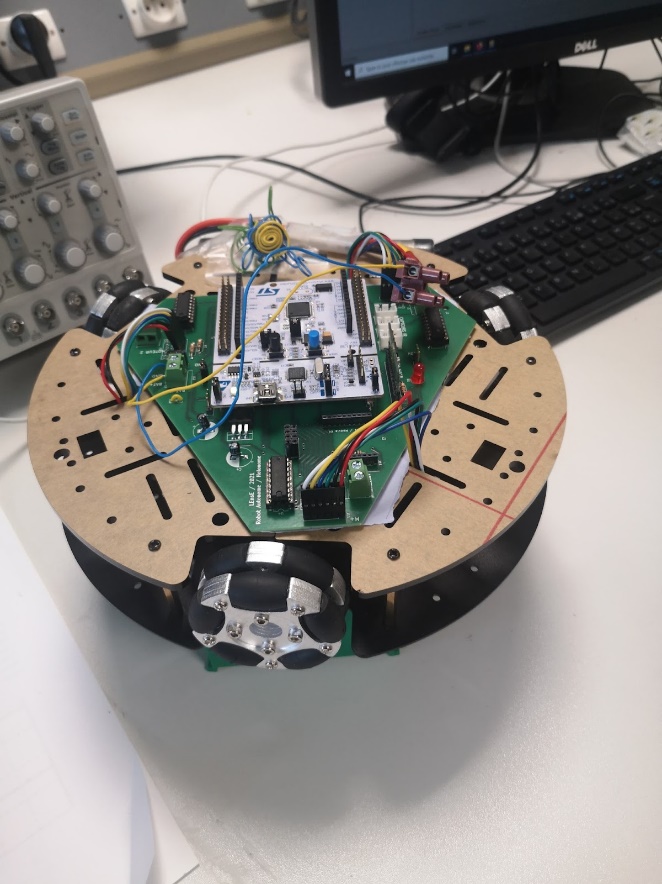


Figure 1 : Robot dans son ensemble

Le matériel que nous utilisé :

* Robot (3 servomoteurs, 3 omniwheels, 1 structure)
* Carte Nucléo (x2)
* Module RF (x2)
* Câblage requis
* Batterie

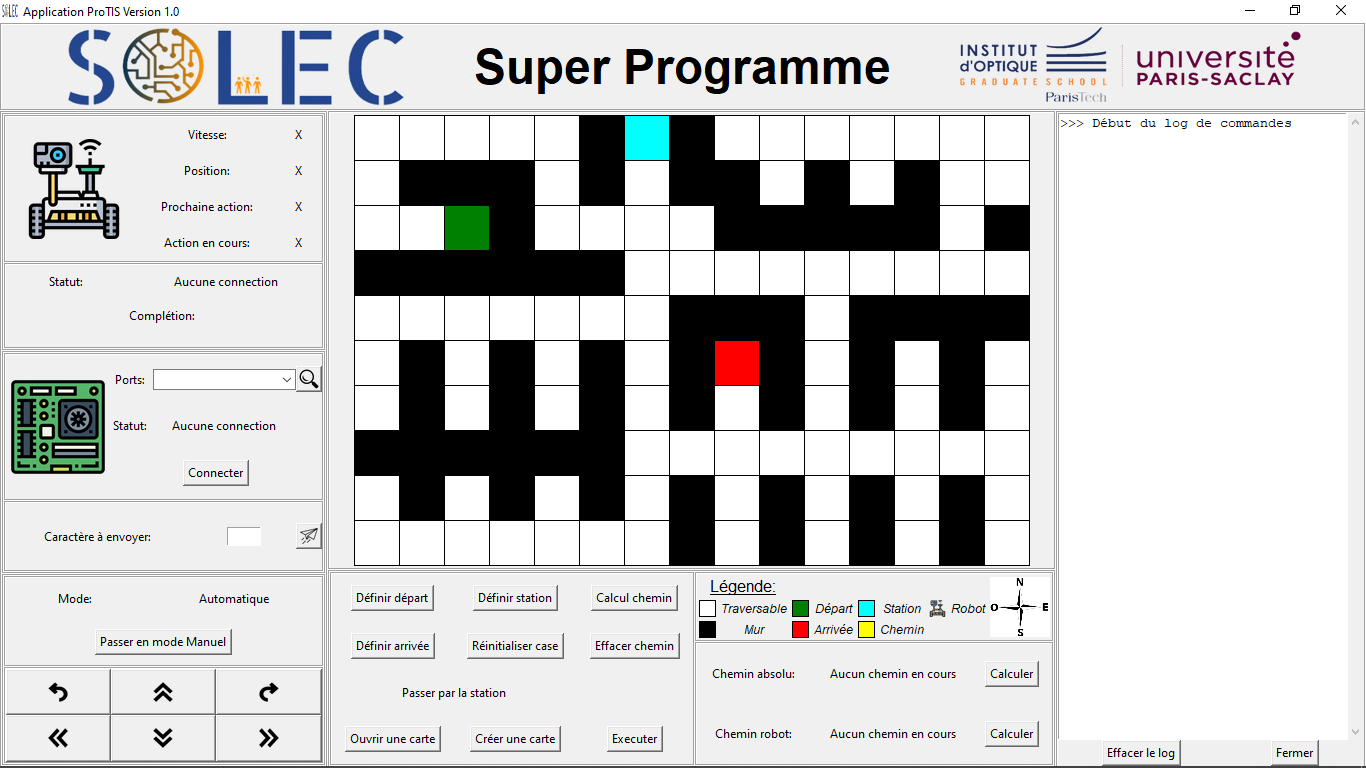


Figure 2 : Interface graphique pour l’utilisation du robot à distance

# Réponse au cahier des charges

Notre robot se devait toutefois de répondre à un certain nombre de contraintes auxquelles nous avons essayé de répondre.

Voici ces contraintes :

* Le déplacement du robot doit être en ligne droite, sans système de guidage prédéfini ;
* Rotations d’angles -90°, +90° et -180° sur lui-même ;
* En cas de problème, un retour sur une base de rechargement doit être prévue ;
* Les consignes doivent provenir d’un ordinateur à distance ;

De plus, on attendait de notre robot de respecter des performances en lien avec son utilisation, i.e. le transport de charges dans un hangar de *Solec* d’un point (A) à un point (B).

Voici ces performances :

* Rapidité : vitesse du robot entre 10 et 30 cm/s.
* Fiabilité : erreur maximale de 2 cm tolérée sur la position et de 3° sur l’angle.
* Autonomie : possibilité de parcourir de 1 km sans recharge des batteries.
* Ergonomie : interface Humain-Machine utilisable sans formation préalable.

# Guide d’utilisateur

Un guide détaillé d’utilisation, d’installation et de démarrage du logiciel est disponible en annexe.

# Communication sans fil

Pour commander le robot à distance depuis l’interface python, nous utilisons des modules radiofréquence de type *KAPPA M868* (documentation en annexe), permettant d’envoyer et de recevoir des informations sous forme de caractères. Nous avons besoin de 2 modules RF et de deux cartes nucléo pour réaliser cette communication. Le principe de fonctionnement du module RF fait que nous ne pouvons envoyer qu’un seul caractère à la fois. Nous avons donc fait en sorte que chaque commande envoyée de l’interface vers le robot, et chaque information envoyée du robot vers l’interface soit sous forme d’un caractère spécifique. La liste des caractères utilisés ainsi que leur signification sont regroupées en annexe dans *Le grand livre des caractères*.

La première carte nucléo est reliée à l’ordinateur servant d’interface homme-machine par une liaison série et sert uniquement à la transmission et réception de caractère (voir code en annexe). On connecte le module RF à cette carte en la branchant sur une borne d’essais. Le câblage de ce module RF est donnée dans la section ***Branchements***

La seconde carte nucléo est intégrée sur un circuit imprimé facilitant le branchement des composants nécessaires au bon fonctionnement du robot. Il suffit donc de connecter le module RF à l’endroit prévu à cet effet (voir ***Branchements***)

Il est intéressant d’utiliser la première carte nucléo pour effectuer des tests avec TeraTerm et vérifier la bonne transmission et réception des caractères et débugger les différents programmes de nucléo.

# Branchements et câblage

## Ordinateur – Nucléo 1 – Module RF

La première carte nucléo est reliée à l’ordinateur par un câble USB/MicroUSB. On connecte ensuite la carte nucléo au module RF en suivant le schéma de la Figure 3

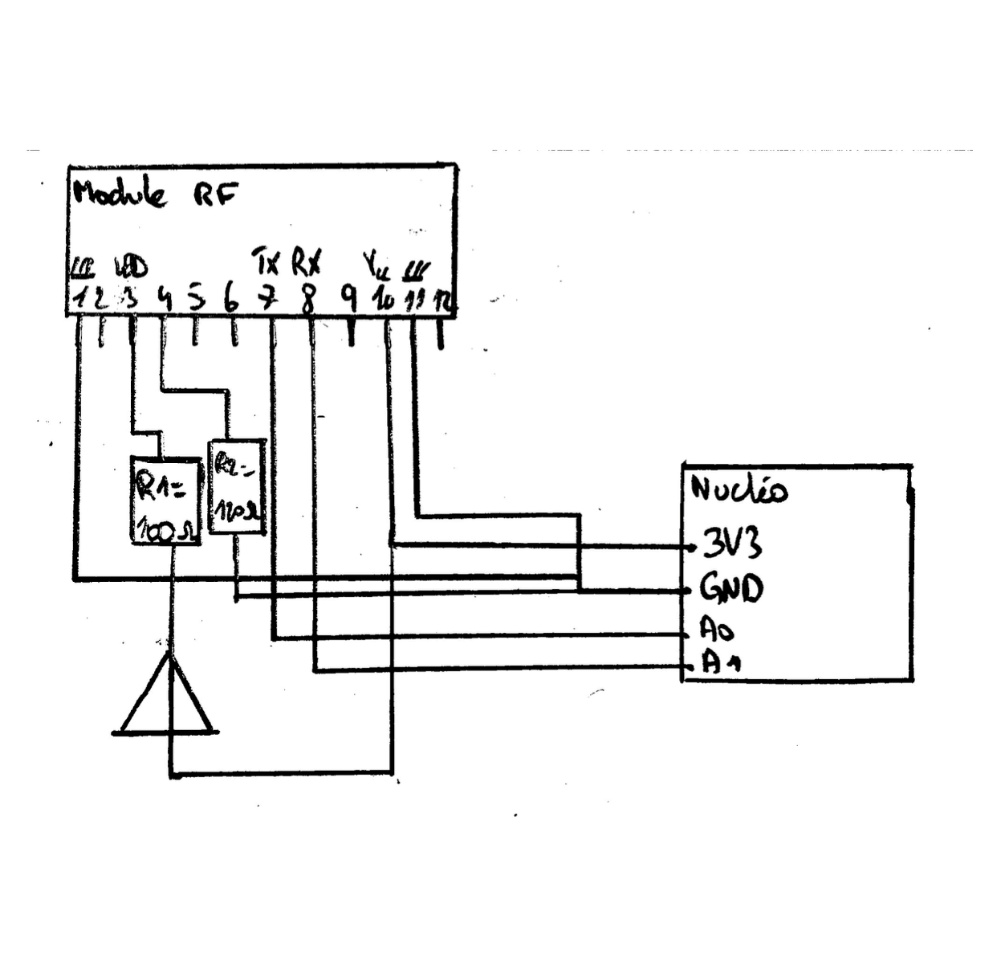


Figure 3 : Schéma de câblage du module RF 1 avec la carte Nucléo 1

## Module RF - Nucléo 2 (robot)

La seconde carte Nucléo est beaucoup plus simple à câbler puisqu’elle est montée sur un circuit imprimé, il suffit d’insérer le module RF dans le support prévu à cet effet (voir Figure 4).

# Motorisation du robot

## Déplacement du robot

Un robot muni de trois roues omnidirectionnelles peut se déplacer dans toutes les directions sans avoir à tourner sur lui-même. Un document fourni en annexe explique comment calculer la commande à appliquer aux moteurs en fonction de la direction de déplacement voulu.

Dans notre projet, on a choisi une approche plus simple : le robot peut soit se déplacer en ligne droite soit tourner sur le même. Pour se déplacer en ligne droite, seulement deux moteurs sont commandés, à la même vitesse mais dans un sens opposé. Et pour tourner sur lui-même, les trois moteurs sont commandés, dans le même sens et à la même vitesse.

## Etalonnage de la distance et des rotations

La commande des moteurs se fait en nombre d’impulsion, il faut donc déterminer une relation entre un nombre d’impulsion et une distance parcourue ou un angle de rotation.

Ces valeurs d’étalonnage peuvent être modifié dans le code du robot, dans les variables distance\_case et *distance\_quart\_tour.*

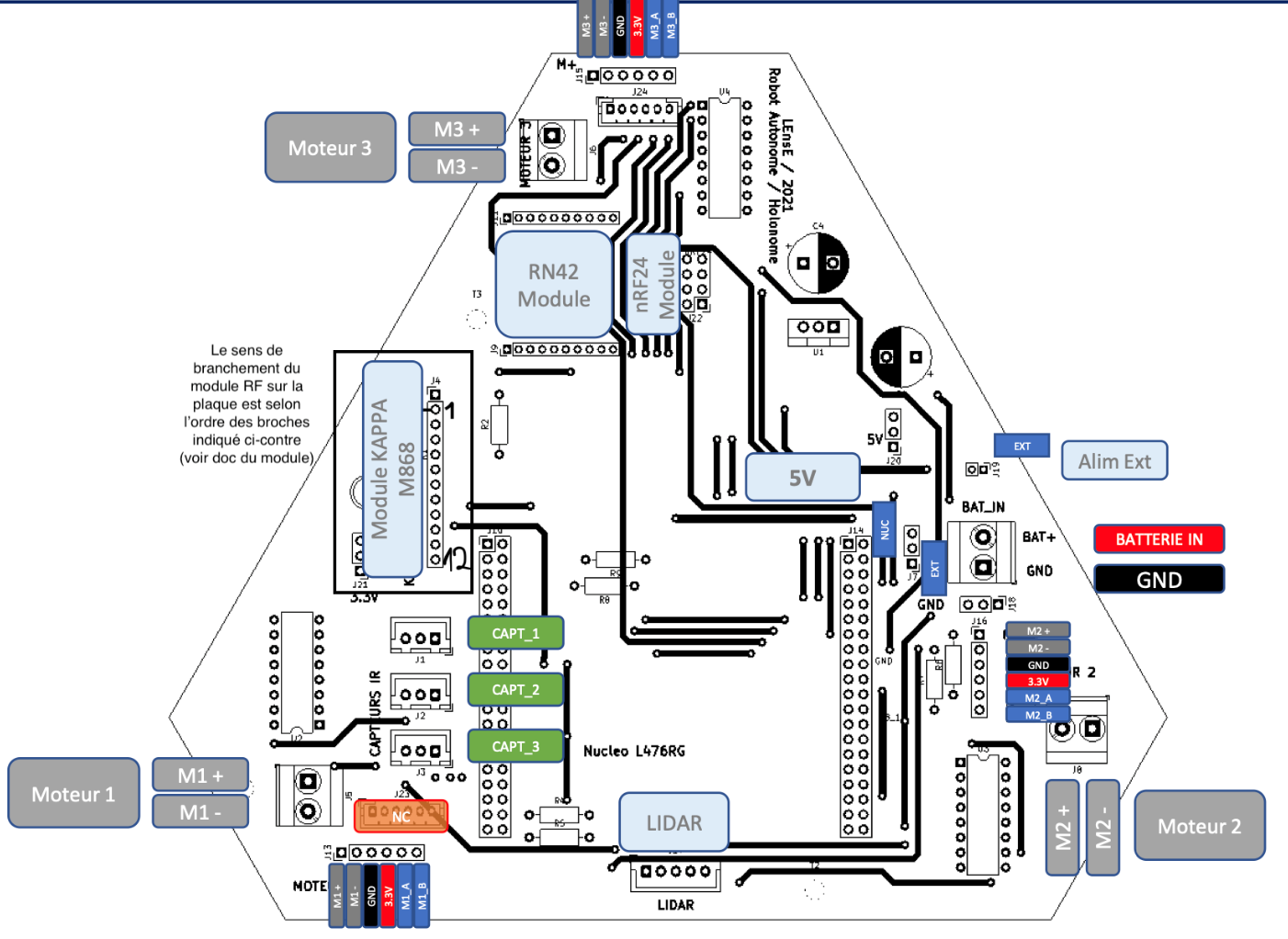


Figure 4: Schéma du circuit imprimé sur le robot (document issue du site internet du LEnsE)

## Asservissement des moteurs

Pour s’assurer que les moteurs tournent à la même vitesse (afin que le robot avance droit), on a décidé de les commander par un asservissement en vitesse et la consigne de vitesse est imposée par une loi sinus ou trapézoïdale (loi trapézoïdale non fonctionnelle).

Pour mesurer la vitesse d’un moteur, on utilise un encodeur. L’encodeur permet de déterminer l'évolution de la position angulaire de la roue en comptant les fronts montants et descendants sur un de ses signaux et le sens est déterminé en regardant l’état de l’autre signal lors du front montant (ou descendant), voir schéma.

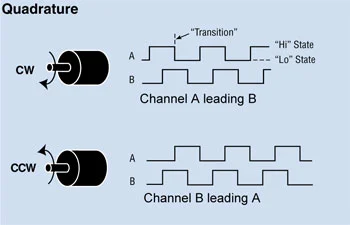


Figure 5: Principe de fonctionnement d’un en codeur

(source : https://www.dynapar.com/technology/encoder\_basics/quadrature\_encoder/)

La vitesse est obtenue en comptant le nombre d’impulsion sur un temps donné (vitesse en impulsion par seconde).

L'asservissement en vitesse est basé sur une correction PID et la période d'échantillonnage peut être choisie (variable “*Te*” dans le code).

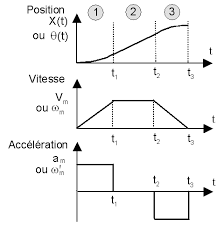


Figure 6: Loi de vitesse trapézoïdale

La consigne de vitesse est déterminée via une loi de vitesse trapézoïdale qui est fonction du pourcentage d’avancement par rapport à la consigne en position.

Les moteurs et l'asservissement sont stoppés quand le pourcentage d’avancement dépasse 100%, il n’y a pas de dépassement car la vitesse des moteurs est très faible grâce à la loi de vitesse utilisée.

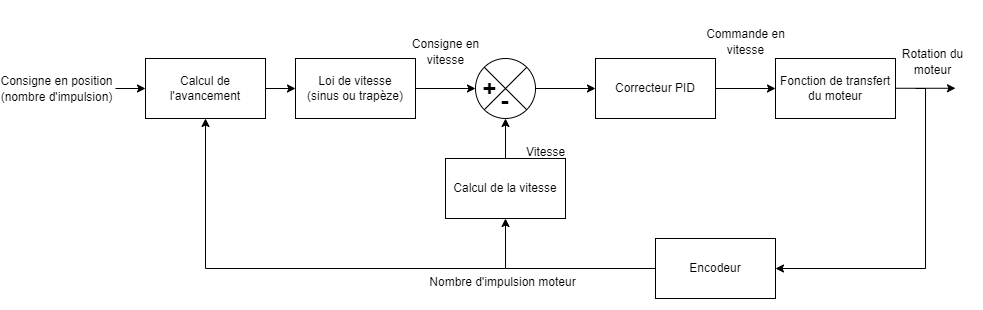


Figure 7: Schéma de l'asservissement

# Système de coordonnées

Dans les différents algorithmes que nous utilisons, le chemin parcouru par le robot n’est pas toujours référé de la même façon. L’algorithme de meilleur chemin nous renvoie une liste de cases (liste de tuples contenant les deux coordonnées) à parcourir.

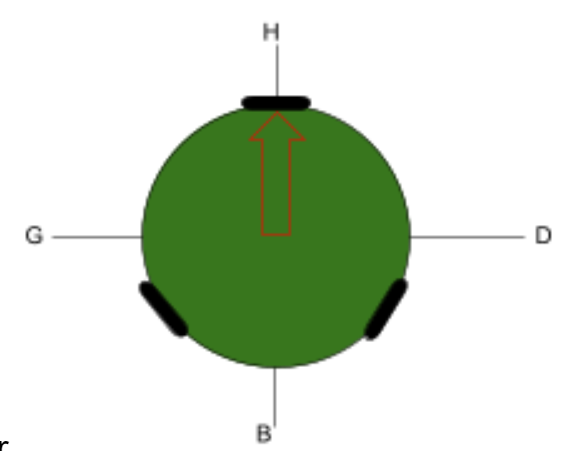


Figure 8 : Direction absolue sur la carte, et sens d’utilisation du robot ici en position “H”.

Nous devons ensuite transformer cette liste dans le système de coordonnées que nous appelons *“absolues”* par rapport à la carte et qui correspond aux directions “Haut, Bas, Gauche, Droite”.

Finalement pour transmettre ces informations aux robots nous transformons ces instructions en coordonnées *“relatives”* aux robots, c'est-à dire aux caractères qui correspondent aux actions du robots.

On peut finalement noter que par convention le robot doit toujours faire face vers le Haut lorsque l’on débute son utilisation.

# Difficultés rencontrées

Au cours des différentes semaines de projet, nous avons été face à certaines difficultés telles que :

* Le partage du matériel avec les différents groupes. En effet, vous devrez partager tout au long de votre projet la voiture avec les autres groupes. Il est donc possible que certains de vos câblages soient débranchés. Nous vous conseillons de toujours garder une trace de votre travail d’une semaine sur l’autre et d'éventuellement écrire vos noms sur la boîte.
* Pour respecter le cahier des charges et avoir un robot qui se déplace sans connexion filaire, nous avons dû ajouter des batteries pour l’alimenter. Toutefois, la tension délivrée par les batteries dépend du niveau de charge. Nous avons donc dû réaliser un asservissement en position : avancer d’une distance d équivaut à x incrémentation pour les moteurs.
* De plus, la trajectoire du robot n’étant pas rectiligne nous avons dû réaliser un asservissement en vitesse en dérivant le retour position du capteur sur un temps Te (50 ms)
* Afin de donner plusieurs consignes de meilleur chemin au robot, il fallait que l’on sache quelle était l’orientation du robot avant d’envoyer les commandes. Pour répondre à cela, nous avons décidé de donner au robot l’instruction de se placer dans une direction artificielle une fois arrivée à sa case finale. Cette direction artificielle sera ainsi la direction initiale du robot pour sa course suivante.

# Idées pour le futur

Nous sommes très fiers de ce projet et sommes très contents qu’il soit repris. Pour le continuer nous vous suggérons quelques pistes de travail pour poursuivre et améliorer le projet :

* Ajouter un retour de vitesse et de position de la part du robot à l'ordinateur que ce cela s’affiche sur l’interface graphique en temps réel
* Nous avons eu aussi l’idée que lorsqu'il y a un problème au niveau de la connexion ordinateur-robot, le robot retourne automatiquement à la “base de recharge” (une zone arbitraire sur la carte).
* Utiliser pour la communication PC/Robot un module Bluetooth connecté à la carte nucléo du robot qui se connecte directement à l’ordinateur. On évite comme cela d’utiliser 2 cartes Nucléo et 2 modules RF.
* Pouvoir envoyer plusieurs instructions en même temps et ne pas attendre un laps de temps défini pour passer à l’instruction suivante.
* Il aurait aussi été intéressant d’intégrer la procédure d’urgence lorsque le robot ne possède presque plus de batterie : il faudrait donc savoir en temps réel le pourcentage de batterie restant. Ainsi, le robot pourrait automatiquement revernir à sa base de rechargement.