République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

جامعة بجاية
Tasdawit n Bgayet
Université de Béjaïa

Université A. MIRA-BEJAIA

Faculté de Technologie

Département de Génie Electrique

Mémoire de Fin d'études

En vue de l'obtention du diplôme de Master en Automatique

Spécialité : Automatique et Système

Thème :

Etude et développement d'applications pour le robot Khepera IV

Réalisée par :

Melle : HARIR Kenza

Encadrées par :

Pr : MENDIL Boubekeur
Mr : DJENADI Ali

Examinateurs :

Mr : SADJI Mustapha
Mr : NAIT MOHAND Nacim

Année Universitaire 2018/2019

Remerciement

Je remercie en premier lieu le DIEU le tout puissant qui m'a donné le courage et la volonté pour réaliser ce modeste travail.

Mes expressions de remerciements sont adressées à mes encadreurs Mr Mendil Boubekeur et Mr Djenadi Ali, pour leurs conseils et orientations et surtout leurs disponibilités.

Mes remerciements vont également aux membres de jury, Mr Sadji Mustapha et Mr Nait Mohand Nacim, qui ont accepté de juger mon travail, ainsi que tous les enseignants qui ont contribué à ma formation.

Merci

Dédicace

Je dédie ce modeste travail à :

- *Mon père et ma mère. Les mots me manquent pour qualifier leurs sacrifices. Que dieu les garde pour nous.*
- *Mes frères et leurs femmes , qui ont toujours été à mes côtés.*
- *Ma chère sœur qui ne cesse de m'encourager et qui a été toujours à mon secours.*
- *A tous mes amis, qu'ils trouvent ici l'expression de mes amitiés sincères.*

H. Kenza.

Liste des figures

Liste des tableaux

Table des matières :

Introduction Générale

La robotique touche aujourd'hui de nombreux secteurs de la vie. Elle a extrêmement progressé durant le siècle passé. Une branche importante concernent les robots mobiles. Ces derniers sont utilisés dans plusieurs domaines, tels que la santé, l'espace, l'agriculture et les applications ménagères et l'automobile.

Dans ce projet, notre objet d'étude est le robot mobile Khepera IV, qui est le 4ème de la série des robots destinés à l'enseignement et la recherche. C'est un robot mobile différentiel à roues, réalisé en janvier 2015. Ce dernier est compact et conçu pour des surfaces planes (sol du laboratoire, table etc.). Il embarque ce qui se fait de mieux en termes de capteurs et actionneurs ce qui le rend un bon champ d'exploitation en termes d'éducation.**[7]**

Notre objectif est d'abord la mise en marche et la familiarisation avec ce robot et le développement d'applications. Il s'agit principalement des tâches : aller vers un position cible ou de suivre une trajectoire constituée de plusieurs points (way-points).

De ce fait, le mémoire est organisé en quatre chapitres. Le premier est consacré à l'étude générale du domaine de la robotique mobile (définition, historique, les différents types…etc.).

Le deuxième présente le robot Khepera IV, sa description technique, ses composants (capteurs et actionneurs) ainsi qu'un aperçu sur ses extensions.

Le troisième chapitre est dédié à l'installation des outils softwares du robot Khepera IV, puis la configuration et les protocoles de communication (USB et Bluetooth) avec ce dernier. Le chapitre se termine avec une petite application qui nous introduit à la programmation de ce robot.

Le quatrième chapitre comporte quelques applications et une série d'essais et de réglages permettant de calibrer l'odométrie.

Enfin, on clôture notre étude par une conclusion générale.

Chapitre I : Généralités sur les robots mobiles

I.1. Introduction

De nos jours, la robotique mobile est utilisée, un peu partout, dans plusieurs domaines voir militaire, médicale et l'industrie. Ce chapitre donne un aperçu général sur ce domaine, tels que les définitions, les classifications, les constituants d'un robot, etc.

I.2. Définition

Robot: Etymologiquement, le mot robot vient du tchèque robota qui signifie travail. Actuellement, ce mot désigne un système automatique asservi à une unité de commande informatisée. Un robot est un ensemble de mobiles associés dont les mouvements sont commandés numériquement et synchronisés. Chaque mouvement élémentaire est un axe de commande numérique. Cependant un robot n'est qu'une machine programmable qui ne fait qu'exécuter ce que l'homme lui a appris. **[8]**

Robot mobile: Un robot mobile est une machine automatique capable de se mouvoir dans un environnement donné.

En général, on peut définir un robot mobile comme étant une machine équipée de capacités de perception, de décision et d'action qui lui permettent d'agir de manière autonome dans son environnement en fonction de la perception qu'il en a **[5]**

I.3. Historique [2]

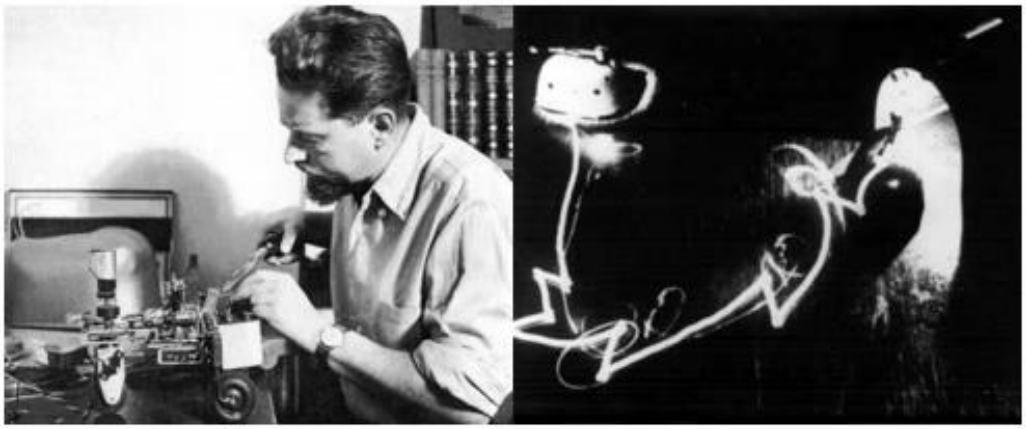


Figure I.1 : La tortue de Grey Walter (nommée "machina speculatrix" et surnommée Elsie).

Le terme de robot apparaît pour la première fois dans une pièce de Karel Capek en 1920 : « Rossum's Universal Robots ». Il vient du tchèque 'robota' (~ servitude) et présente une vision

des robots comme serviteurs dociles et efficaces pour réaliser les tâches pénibles mais qui déjà vont se rebeller contre leurs créateurs.

La Tortue construite par Grey Walter dans les années 1950 (Figure I.1), est l'un des premiers robots mobiles autonomes. Grey Walter n'a utilisé que quelques composants analogiques, dont des tubes à vide, mais son robot est capable de se diriger vers une lumière qui marque un but, de s'arrêter face à des obstacles et de recharger ses batteries lorsqu'il arrive dans sa niche. Toutes ces fonctions sont réalisées dans un environnement entièrement préparé, mais restent des fonctions de base qui sont toujours des sujets de recherche et de développement technologiques pour les rendre de plus en plus génériques et robustes.

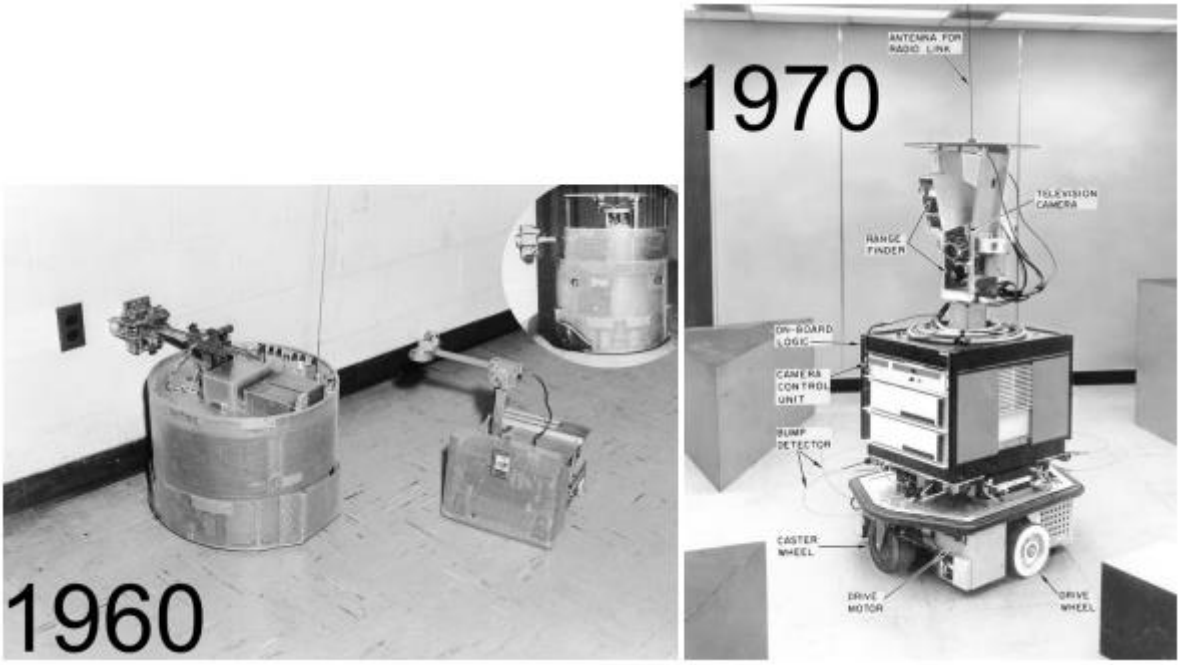


Figure I.2 : Robot "Beast" de l'université John Hopkins (1960) à gauche et le robot Shakey de Stanford (1969) à droite.

L'apparition des transistors durant les années 60, a conduit à des robots plus complexes. Ainsi, le robot "Beast" (Figure I.2) de l'université John Hopkins est capable de se déplacer au centre des couloirs en utilisant des capteurs ultrason, de chercher des prises électriques (noires sur des murs blanc) en utilisant des photo-diodes et de s'y recharger. Les premiers liens entre la recherche en intelligence artificielle et la robotique apparaissent à Stanford en 1969 avec Shakey (Figure I.2). Ce robot utilise des télémètres à ultrason et une caméra et sert de plate-forme pour la recherche en intelligence artificielle, qui à l'époque travaille essentiellement sur des approches symboliques de la planification. La perception de l'environnement, qui à l'époque est considérée comme un problème séparé, voire secondaire, se révèle

particulièrement complexe et conduit là aussi à de fortes contraintes sur l'environnement. Ces développements de poursuivent avec le Stanford Cart dans la fin des années 1970, avec notamment les premières utilisations de la stéréo-vision pour la détection d'obstacles et la modélisation de l'environnement. En France, le robot Hilare est le premier robot construit au LAAS, à Toulouse (Figure I.3).

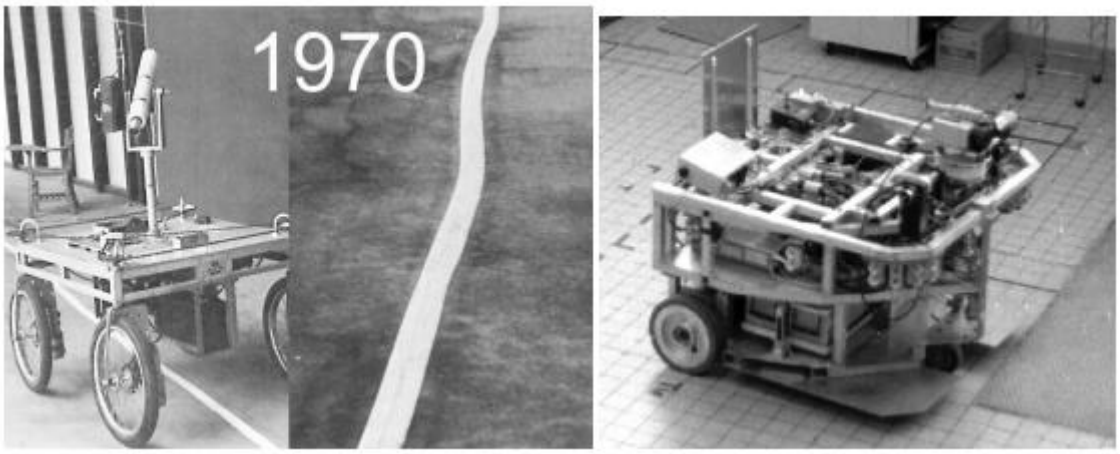


Figure I.3 : Le Stanford Cart (1970) et le robot Hilare du LAAS (1977).

Une étape importante est à signaler au début des années 1990 avec la mise en avant de la robotique réactive, représentée notamment par Rodney Brooks. Cette nouvelle approche de la robotique, qui met la perception au centre de la problématique, a permis de passer de gros robots très lents à de petits robots (Figure I.4), beaucoup plus réactifs et adaptés à leur environnement. Ces robots n'utilisent pas ou peu de modélisation du monde, problématique qui s'est avérée être extrêmement complexe.

Figure I.4 : Genghis, développé par Rodney Brooks au MIT (1990).

Ces développements ont continué et l'arrivée sur le marché depuis les années 1990 de plateformes intégrées telles que le pionnier de la société Mobile Robots a permis à de très nombreux laboratoires de travailler sur la robotique mobile et a conduit à une explosion de la diversité des thèmes de recherche. Ainsi, même si les problèmes de déplacement dans l'espace et de modélisation de l'environnement restent difficiles et cruciaux, des laboratoires ont pu par exemple travailler sur des approches multirobot, la problématique de l'apprentissage ou sur les problèmes d'interactions entre les hommes et les robots.

I.4. Classification des robots mobiles [1]

La classification des robots mobiles se fait suivant plusieurs critères (degré d'autonomie, système de locomotion, énergie utilisée, etc.). La classification la plus utilisée est selon leur degré d'autonomie.

- Véhicule télécommandé par un opérateur qui lui impose chaque tâche élémentaire à réaliser.
- Véhicule télécommandé au sens de la tâche à réaliser. Le véhicule contrôle automatiquement ses actions.
- Véhicule semi-autonome réalisant sans l'aide de l'opérateur des tâches prédéfinies
- Véhicule autonome qui réalise des tâches semi-définies. Ce type de véhicule pose des problèmes d'un niveau de complexité élevé de représentation des connaissances, de capacité décisionnelle et de génération de plans qui sont résolus à bord dans la mesure du possible. L'ensemble des problèmes particuliers liés à la conception de tels robots sont :
    - La conception mécanique liée à la mobilité.
    - La détermination de la position et de la latitude (orientation).
    - La détermination du chemin optimal pour atteindre le lieu de la tâche.

I.5. L'architecture des robots mobiles [10]

L'architecture des robots mobiles se structure en quatre éléments :

- La structure mécanique et la motricité
- Les organes de sécurité
- Le système de traitement des informations et gestion des tâches.
- Le système de localisation.

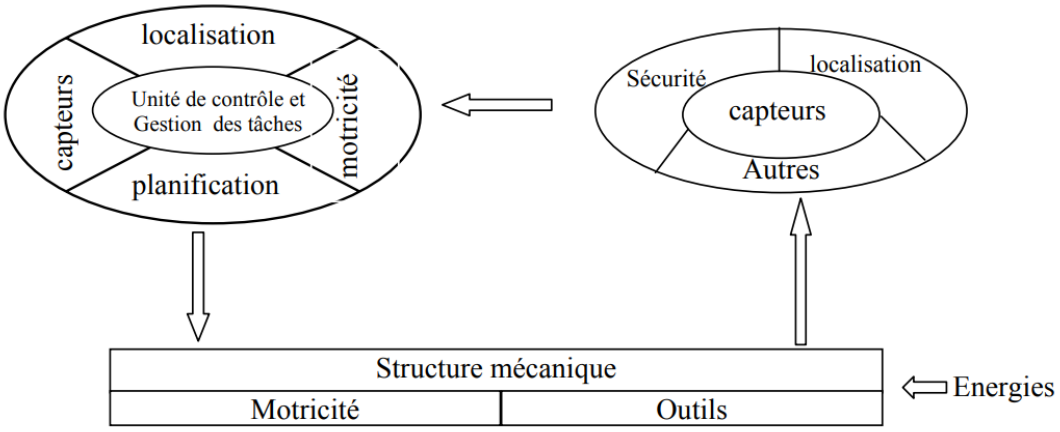


Figure I.5 : Architecture d'un robot mobile

I.5.1. La structure mécanique et la motricité

Il existe quatre types de structures mécaniques assurant la motricité :

Les mobiles à roues :

La mobilité par roues est la structure mécanique la plus communément utilisée pour des raisons de simplicité. Cette technique assure selon l'agencement et les dimensions des roues un déplacement qui se fait uniquement selon la tangente au mouvement des roues.

Les mobiles à chenilles :

L'avantage de ces robots est le fait qui ils ont la meilleure adhérence au sol et une faculté de franchissement d'obstacles. La commande est réalisée en imposant une déférence de vitesse aux chenilles droite et gauche.

Les mobiles marcheurs :

La plupart du temps, ce sont des robots destinés à réaliser des tâches variées dont l'accès au site est difficile, dangereux ou impossible à l'homme. Leur anatomie à nombreux degrés de liberté permet un rapprochement avec les robots manipulateurs. La conception et la commande de tels mécanismes sont complexes.

Les mobiles rampants :

Ses robots sont utilisés pour la progression dans des galeries ou des tuyaux. Le système est composé d'un ensemble de modules ayant chacun plusieurs mobilités. Les techniques utilisées découlent des méthodes de locomotion des animaux.

- Le type scolopendre constitue une structure inextensible articulée selon deux axes orthogonaux.

- Le type lombric comprend trois articulations, deux rotations orthogonales et une translation dans le sens du mouvement principal.
- Le type péristaltique consiste à réaliser un déplacement relatif d'un module par rapport aux voisins.

La motricité et l'énergie

Les déplacements des robots sont réalisés par des moteurs de types électrique, thermique ou hydraulique. L'énergie électrique la plus fréquemment employée offre l'avantage d'une commande aisée. Cependant le transport et la génération présentent des difficultés. Plusieurs méthodes sont employées :

- Par batteries qui sont soit rechargées périodiquement de manière automatique ou manuelle, soit par un échange avec d'autres lorsqu'elles sont déchargées.
- Par groupe électrogène embarqué dont l'inconvénient constitue la masse élevée. L'énergie de base est alors thermique.
- Par cordon ombilical qui réduit l'autonomie du robot. L'énergie thermique est essentiellement employée par des véhicules de forte puissance comme énergie de base pour la traction ou pour activer un compresseur hydraulique.

I.5.2. Les organes de sécurité

II est dangereux de laisser le robot mobile complètement libre. Donc, il est obligatoire qu'il soit doté d'organes garantissant la sécurité. Deux types de capteurs sont employés :

- Les capteurs proximétriques assurent la détection avant collision (ultrasons, hyperfréquences, infrarouge...).
- Les capteurs à contact détectent une collision ou un choc avec l'environnement (contact électrique sur pare-chocs, résistance variable, fibre optique...). L'organisation de la sécurité d'un robot mobile est représentée sur le schéma suivant :

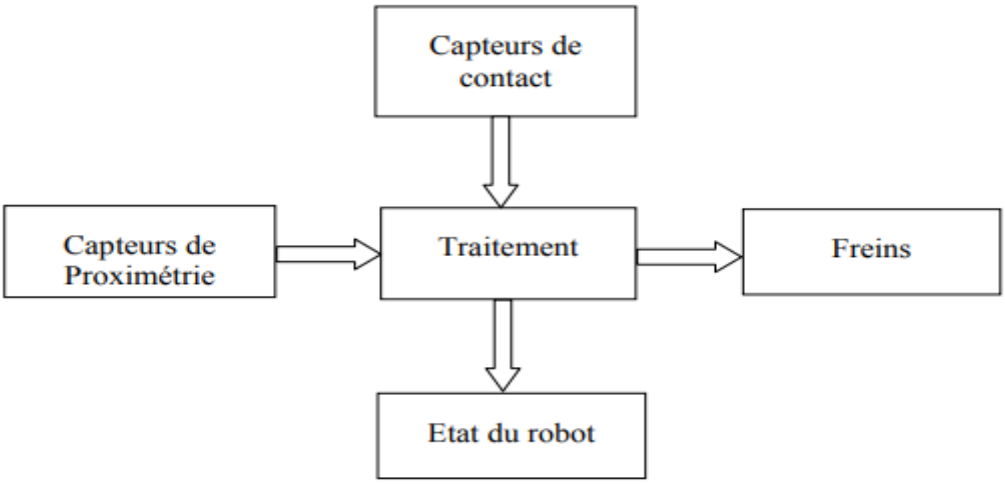


Figure I.6 : Synoptique de la sécurité

Le comportement du robot mobile lors de la détection d'un obstacle s'effectue selon plusieurs cas :

- Si le capteur à contact est sollicité, le robot s'immobilise soit définitivement soit tant que le contact persiste, ou il effectue un mouvement opposé au contact.
- Si on détecte une présence, la stratégie consiste soit à immobiliser le robot en attendant que la personne s'éloigne, soit à ralentir le mouvement si la personne n'est pas très proche, soit à choisir un autre chemin qui l'éloigne de la personne.

I.5.3. Traitement des informations et gestion des tâches

L'ensemble de traitement des informations et gestion des tâches constitue le noyau du module d'informatique central qui établit les commandes permettant au robot mobile de réaliser un déplacement et d'activer les divers organes en accord avec l'objectif. A ce niveau le problème qui se pose est celui de génération de plan qui consiste à établir la manière dont le robot se déplace par rapport à des connaissances aprioriques « statiques » ou obtenues en cours d'évolution « dynamiques » . La génération des plans repose sur trois concepts :

- La stratégie de navigation.
- La modélisation de l'espace.
- La planification

La navigation :

La navigation est une étape très importante en robotique mobile. Bien entreprise, elle permet une large autonomie à un robot mobile. Le système de navigation comporte plusieurs modules qui peuvent être traités différemment et parmi lesquels on distingue celui de la localisation et celui de l'évitement d'obstacles.

La modélisation de l'environnement :

La connaissance du milieu dans lequel évolue le robot mobile n'est établie en général qu'après avoir effectué une campagne de mesure de l'ensemble des éléments constituant l'environnement. Cette procédure fastidieuse peut être évitée si le robot construit lui-même son modèle d'environnement de manière dynamique. Par contre, la planification de trajectoire n'est pas utilisable tant que le robot ne dispose pas d'un modèle de l'espace d'évolution ce qui handicape très fortement l'utilisation du robot. A partir de cette base d'informations et d'une loi évaluant les erreurs de représentation, le planificateur peut générer des sous-trajectoires faisables dans certaines parties et modifier les sous-trajectoires dans d'autres parties à l'aide des informations locales issues de la mesure des capteurs d'environnement. Lors de l'exécution d'une trajectoire, le robot acquiert des informations qui vont permettre de reconstituer le plus fidèlement possible le modèle de l'environnement de manière récursive à l'aide d'un algorithme approprié.

La planification de la trajectoire :

On voit ainsi au travers de cette première approche assez théorique apparaître un problème essentiel : *la planification de la trajectoire*. Différentes approches sont envisageables selon que le robot évolue en milieu connu ou inconnu.

L'évolution en territoire cartographié simplifie évidemment la tâche des concepteurs. Une fois la carte de la zone d'évolution rentrée dans la mémoire d'un ordinateur communiquant avec le robot ou bien dans une mémoire intégrée au robot lui-même, des algorithmes de routage permettent de diriger le robot.

Il en va tout autrement dans le cas de l'évolution en territoire inconnu. Le robot doit alors analyser son environnement au moyen de différents capteurs, détecter sa position par rapport à son but et décider de sa trajectoire. Cette localisation peut s'effectuer par différentes méthodes : triangulation de signaux émis par des balises déposées au cours du déplacement ou/et repérage d'obstacles à distance et construction d'une carte du site, mesures odométriques et estimation

de la position. On applique ensuite des algorithmes complexes pour diriger le robot. Ceux-ci peuvent amener des résultats plus ou moins bons. Le principal problème étant la non-convergence de certaines boucles de déplacement. Si aucun algorithme de secours n'a été prévu, l'intervention humaine est alors nécessaire.

I.5.4. Systèmes de localisation :[4]

Pour que le robot soit autonome, en réalisant certaines tâches données, il doit pouvoir se localiser dans son environnement. La localisation est l'un des problèmes majeurs de la robotique mobile. Pour qu'un robot soit capable de réaliser un déplacement, il doit obligatoirement savoir sa position actuelle pour calculer sa trajectoire afin d'aller à la position cible. En plus de son déplacement, le robot doit mettre à jour sa trajectoire en calculant la position tout au long de son déplacement, et ceci pour savoir s'il est arrivé à sa destination cible ou pas. On cite les moyens de localisation les plus utilisés.

L'odométrie :

La technique de l'odométrie est la plus utilisée pour la localisation des robots mobiles à roues. Elle permet de déterminer la position et le cap d'un véhicule par intégration de ses déplacements élémentaires, et ce, par rapport à un repère lié à sa configuration initiale. Le point faible de cette méthode est l'accumulation d'erreurs d'orientation, qui va engendrer des erreurs importantes sur la position. Cette dernière augmente proportionnellement à la distance parcourue. Malgré son point faible, cette méthode reste présente est plus utilisée, à cause de sa simplicité de mise en œuvre et son faible coût.

Capteurs de distance :

Les télémètres ou les capteurs de distance sont très utilisés eux aussi dans la robotique mobile. Ils donnent la possibilité d'avoir l'information de distance entre l'obstacle le plus proche et le robot. On trouve les capteurs ultrasons qui reposent sur l'utilisation des ondes acoustiques dont la fréquence est trop élevée et non audible par l'être humain, et les capteurs utilisant la lumière qui détectent l'intensité de la lumière réfléchie.

Les caméras :

Désigné par la localisation visuelle, cette méthode consiste à identifier des amers et à se localiser en utilisant par exemple une carte de caractéristiques. Cette dernière représente l'environnement par les coordonnés globales d'amers (points, lignes, polygones…etc.), ou par une carte topologique qui caractérise l'environnement par un graphe.

I.6. Les moyens de perception en robotique mobile

La perception est un domaine crucial de la robotique. C'est autour de ce concept qu'est bâtie la structure d'un robot apte à exécuter des tâches complexes ou à évoluer dans un univers inconnu ou mal connu. L'élément de base du système de perception est le capteur qui a pour objet de traduire, en une information exploitable, des données représentant des caractéristiques de l'environnement.

Les moyens utilisés pour la perception de l'environnement sont nombreux. On cite :

- Les systèmes de vision globale.
- Les télémètres laser et ultrasonores.
- Les capteurs optiques et infrarouges.
- Les capteurs tactiles.

I.7. Domaines d'applications actuels et émergeant des robots mobiles

Actuellement, le marché commercial de la robotique mobile est toujours relativement limité, en dehors des robots aspirateurs vendus à plusieurs millions d'exemplaires. Cependant, il existe de nombreuses visions de développement qui en feront probablement un domaine important dans le futur. Les applications des robots peuvent se trouver dans de nombreuses activités "ennuyeuses, salissantes ou dangereuses" (3 D's en anglais pour Dull, Dirty, Dangerous), mais également pour des applications ludiques ou de service, comme l'assistance aux personnes âgées ou handicapées.

Figure I.7 : Exemples de robots commerciaux ou de recherche

Parmi les domaines d'applications possibles de la robotique, on cite : **[6]**

Nucléaire : maintenance, démantèlement d'installations, décontamination, inspection et intervention en cas d'accident …

Spatial : pose d'une lentille sur le télescope Hubble, exploration martienne et petit robot Sojourner, … etc.

Sous-marin : torpillerie légère autonome, inspection et réparations de structures offshore, exploration du TITANIC, cartographie des fonds, missions en émissaires (eaux, égouts), pose de câbles de télécommunications et de puissance, … etc.

Agriculture : robots cueilleurs de fruit, planteurs, désherbage robotisé, guidage de véhicules agricoles, traite automatique des vaches laitières, … etc.

Activités ludiques : compétitions de robots (« Robocop ») qui mobilisent de nombreux chercheurs et étudiants, … etc.

Santé : Positionnement dans l'espace des capteurs d'échographie ou des patients eux- mêmes, fauteuils roulants intelligents, chirurgie cardiaque, oculaire, du cerveau, applications de formation, d'évaluation, d'entrainement dans lesquelles on associe réalité virtuelle et gestes assistés par robotique, neuroprothèses.

Véhicules automatiques : Assistance à la conduite automobile (utilisant largement les résultats des travaux sur les robots mobiles en localisation, évitements d'obstacles, planification de mouvement), petits véhicules volants ou drones (dirigeables, avions, hélicoptères) pour applications militaires, de cartographie automatique, d'inspection de lignes haute tension ou de localisation d'accidentés en montagne.

I.8. Conclusion

Dans ce chapitre, on a exposé les notions de base concernant la robotique mobile, telles que les différentes architectures et classification des robots mobiles. Le prochain chapitre donne une description technique du robot mobile Khepera IV qui fait l'objet de notre étude.

Chapitre II : Présentation du robot Khepera IV

II.1. Introduction

Dans ce chapitre, on va étudier la description technique du robot Khepera IV, en termes de composants, de leurs fonctionnalités et de ses extensions.

II.2. Définition[3]

Khepera IV est le robot Khepera le plus récent de la série des robots mobile « Khepera » et aussi le plus avancé sur le plan technologique. Ce robot est le fruit de plus d'une décennie d'expériences et de conceptions améliorées intégrant une technologie de pointe dans un format compact. Le Khepera IV est une idéale plate-forme pour effectuer des recherches en intérieur avec un seul robot ou dans un essaim dans un espace restreint tel que sur le sol d'un laboratoire ou sur une table.

Figure II.1 : Evolution et différentes versions du robot Khepera

II.3. Description technique[3]

Le Khepera IV est un petit robot à 2 roues conçu pour l'utilisation à l'intérieur. Il est sous forme d'un cylindre de diamètre 14.08 cm et hauteur de 5.77 cm (Les roues sont incluses). Sa coque extérieure est composée de deux pièces en plastiques dur.

A l'intérieur, il suit un dessin empilé de circuits imprimés. Le robot complet pèse 566g. La figure II.2 montre les différentes faces du robot. Les 2 roues actionnées ont un diamètre de 42mm (y compris les joints toriques qui agissent comme pneus) et sont centrées sur chaque coté du robot, séparée de 10.54cm. En plus, deux autres roues pivotantes en avant et en arrière rendent le robot très stable et empêchent son utilisation sur toute surface qui n'est pas réellement plane et lisse.

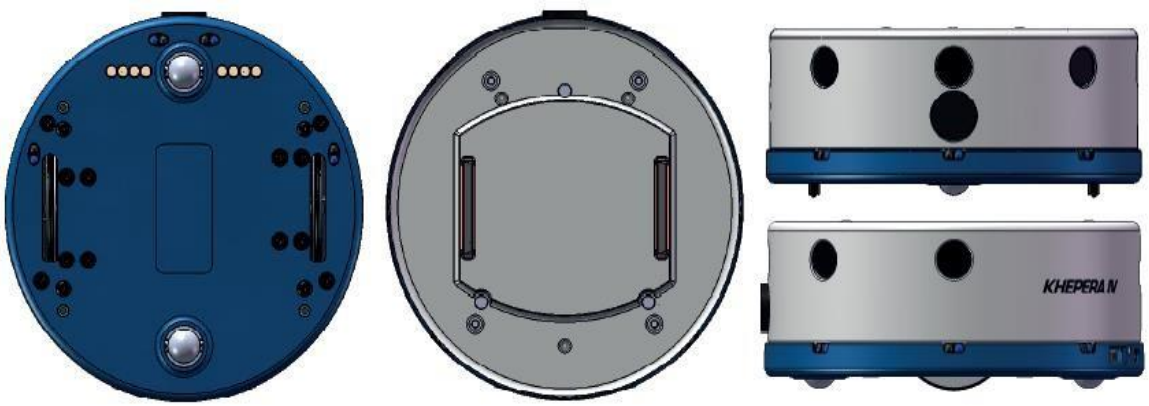


Figure II.2 : Vue du bas, du haut, de l'avant et de la gauche du robot

II.4. Description électronique[3]

Le cerveau du robot est un Gumstix Overo FireSTORM-P COM, un ordinateur embarqué standard. Cet ordinateur sur module dispose d'un DSP supplémentaire pour effectuer des tâches spéciales, ainsi que des fonctionnalités Bluetooth et Wifi (antenne SMD montée sur le Khepera). Gumstix est fourni avec un système Linux déjà installé.

Le robot dispose une carte microSD préinstallée de 4 Go pour les programmes utilisateur et les données. Les aspects de bas du robot sont gérés par un microcontrôleur dsPic33FJ64 GS608 qui établit un pont entre l'ordinateur intégré est le matériel intégré. Des périphériques additionnels peuvent être connectés via un bus d'extension au-dessus du robot, ainsi que par un port USB.

L'énergie est fournie par une batterie 3400 mAh et 7.4V. celle-ci ne peut pas être remplacée et peut être rechargée en environ 5heures en utilisant la prise de charge. Un support est également fourni pour la charge à partir du bus d'extensions et à partir d'un ensemble de contacts sous le corps du robot.

II.5. Capteurs et Actionneurs[7]

Le robot Khepera IV est équipé d'un riche ensemble de dispositifs actionneurs/ capteurs.

II.5.1. Capteurs infrarouges

Douze capteurs infrarouges sont montés sur le robot. Huit capteurs sont placés autour du robot pour détecter les obstacles et mesurer la lumière ambiante. Ces capteurs permettent au robot de détecter des obstacles de 2 à environ 250mm, tout dépend des conditions ambiantes et la couleur de l'obstacle. Tandis que les 4 autres sont placés sur le bas du robot et permettent de réaliser des expériences comme suivi de ligne et évitement de tomber.

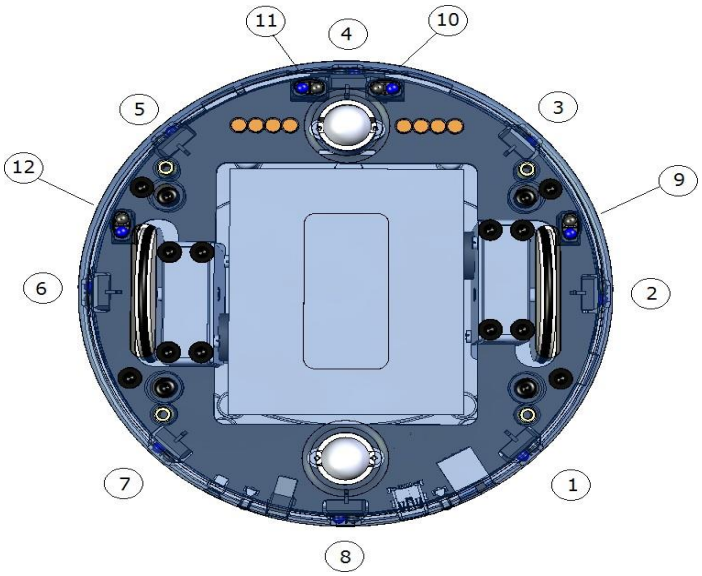


Figure II.3 : Les douze capteurs infrarouges du robot Khepera IV. Ceux qui entourent le robot sont numérotés de 1 à 8 et ceux du bas de 9 à 12.

II.5.2. Capteurs à ultrasons

Ce robot est doté de cinq capteurs ultrasons, deux de chaque côté et un en avant, comme montré par la figure II.4. Avec ces capteurs, le robot peut détecter les obstacles d'environ 250 à environ 2500mm

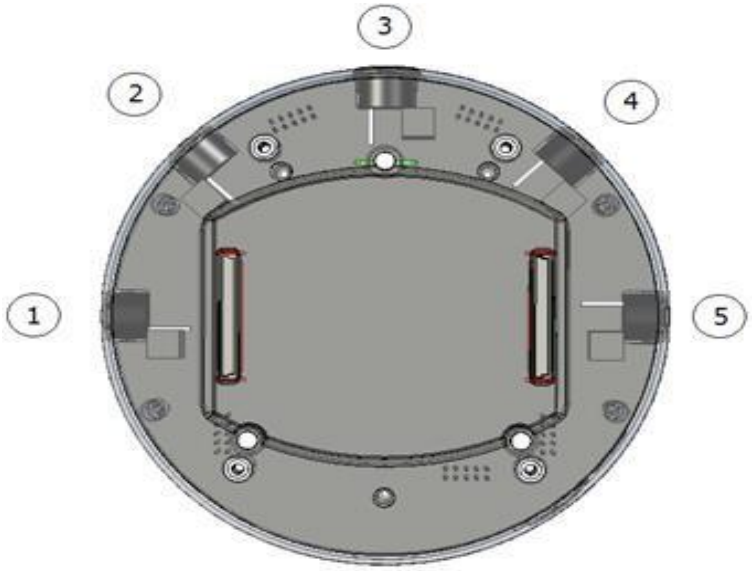


Figure II.4 : Capteurs ultrasoniques autour du robot numérotés de 1 à 5.

II.5.3. Unité de mesure inertielle (IMU)

Une IMU est montée au centre de rotation, dotée d'un accéléromètre 3D et un gyroscope 3D. L'accéléromètre renvoie une valeur positive pour l'axe X quand le robot marche en avant, l'axe Y est positif à gauche et finalement l'axe Z est négatif au sens de la gravité.

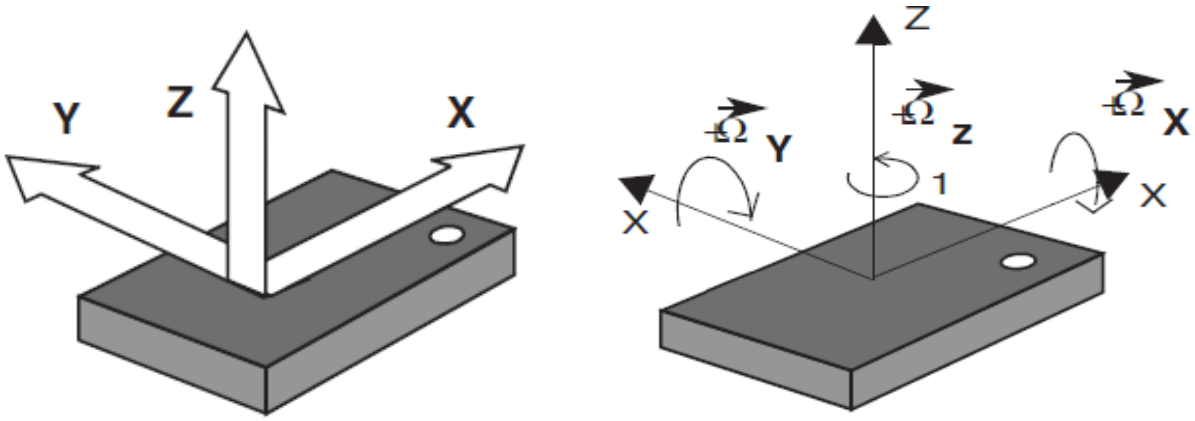


Figure II.5 : Directions des vitesses linéaires et sens de rotations des vitesses angulaires.

Les directions des vitesses angulaires pour le gyroscope sont définies autour des axes de l'accéléromètre, selon Figure II.5.

II.5.4. Microphones

Deux microphones amplifiés sont placés sur les côtés du robot. Ces microphones omnidirectionnels ont un gain de 20 dB et une plage de fréquence de 100-10000 Hz. Le RSB nominal est de 59 et la sensibilité est de -22dBV dans 1 kHz.

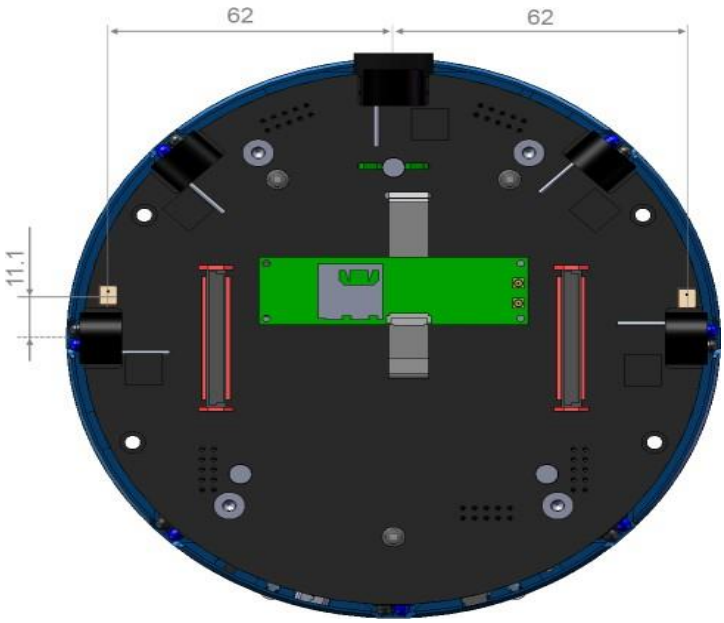


Figure II.6 : Positions des microphones.

II.5.5. Caméra

Une caméra en couleur est disposée sous le capteur ultrason en avant. La taille de l'image active est 4.51x2.88 mm et sa résolution est 752x480.

II.5.6. Moteurs

Le mouvement du robot est assuré par deux moteurs à courant continu qui conduit ses roues, un entrainant chaque roue. Chaque moteur a une puissance nominale de 1.96W, transmise par deux boites de vitesse avec un rapport de transmission global de 66.3 %, donnant 1.3W de puissance utilisable par roue.

Les moteurs sont couplés à des codeurs haute résolution avec une révolution complète de roue correspondant à 19456 impulsions. Le diamètre de la roue étant de 42 mm (le périmètre étant alors de 131,94 mm), on obtient 147,4 impulsions par millimètre. La vitesse des moteurs

est régulée par modulation de largeur sur différents modes : contrôle de vitesse en boucle fermée, contrôle de profile de vitesse ainsi qu'un contrôle de position en boucle ouverte

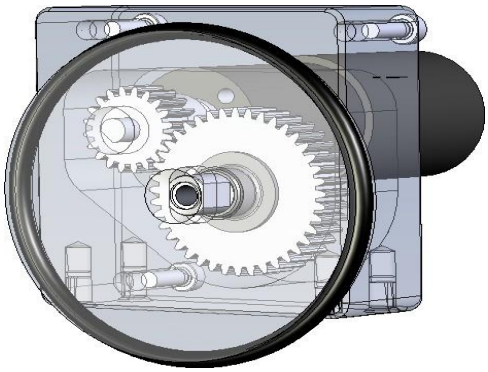


Figure II.7 : Bloc du moteur avec roue.

II.5.7. LED RVB

Le robot est équipé de 3 LED RVB montées sur le sommet du robot en forme du triangle isocèle. La couleur des LED peut être contrôlée avec une résolution de 6 bits sur chaque canal, ce qui les rend utiles pour le suivi et l'identification.

II.5.8. Haut-parleurs

Enfin, une enceinte PUI Audio SMS-1308MS-2-R, de puissance nominale 1 W et plage de fréquences de 400 à 20000 Hz peuvent être utilisés pour la communication ou les interactions.

II.6. Cartes d'extension [9]

La fonctionnalité native du robot peut être étendue via l'utilisation de l'USB ou Bluetooth ou en concevant des cartes personnalisées. K-Team (le fabricant des robots Khepera) commercialise plusieurs cartes, y compris une pince, un télémètre laser et un module de localisation intérieure Stargazer.

(a) **(b)** **(c)**

Figure II.8 **:** Modules d'extension : **(a)** Pince **(b)** Télémètre laser **(c)** Stargazer

II.7. Conclusion

Bien que le robot Khepera IV et de tout petite taille, il est doté de plusieurs et différents types de capteurs, ainsi d'une puissance de calcul appréciable. Ce qu'il le rend très utilisé pour la recherche.

Khepera IV a un mini cerveau programmable qu'on peut commander. Ce qu'on va découvrir dans le 3ème chapitre.

Chapitre III: Configuration et mise en marche du robot Khepera IV

III.1. Introduction

Après avoir vu la description technique et les différents composants du Khepera IV, dans ce qui suit on s'intéressa au cerveau de ce petit robot et on va présenter notre première tentative de le commander. Mais, avant tout, on prépare le robot en installant d'abord tout le software nécessaire.

III.2. Installation des Logiciels

Actuellement, il existe deux bibliothèques open source pour le développement d'applications : *libkhepera* et le *toolbox*. Les deux offrent des fonctionnalités de base similaires et constituent des améliorations par rapport aux anciennes bibliothèques. Elles sont indépendantes et les programmes qui les utilisent peuvent coexister dans le même robot, bien qu'il ne soit pas recommandé de les exécuter en parallèle.

III.2.1. Installation de « *light toolchain* »

L'installation de ce logiciel est nécessaire pour utiliser les outils de développement. Pour ce faire, on a besoin des fichiers suivants que on peut trouver dans le DVD du robot ou sur internet (***http://ftp.kteam.com/KheperaIV/software/***):

- Le fichier « light_toolchain/ » :

  poky-glibc-i686-khepera4-image-cortexa8hf-vfp-neon-toolchain-1.8.sh

- Le fichier « library/ » :

  libkhepera_2.X.tar.bz2

III.2.2. Installation du répertoire du développement

Le répertoire du développement va être le fichier de base pour nos programmes développés. Il contiendra des liens et scripts qui aident à faire nos projets facilement avec le « *cross compiler* ».

Pour créer ce fichier, on exécute, sur le terminal Linux, la commande :

```
mkdir ~/khepera4_development

cd ~/khepera4_development
```

III.2.3. Installation du cross-compiler (toolchain) :

- On exécute les commandes suivantes sur le terminal :

  chmod +x poky-glibc-i686-khepera4-image-cortexa8hf-vfp-neon-toolchain-1.8.sh

  sudo ./poky-glibc-i686-khepera4-image-cortexa8hf-vfp-neon-toolchain-1.8.sh

- On suit les instructions du programme d'installation en utilisant les paramètres par défauts.
- On peut s'assurer que l'installation est correcte, en exécutant le cross-compiler. Mais avant, on doit créer un environnement de variables avec la commande :

  source /opt/poky/1.8/environment-setup-cortexa8hf-vfp-neon-poky-linux-gnueabi

  puis on vérifie la version du cross-compiler avec la commande :

  arm-poky-linux-gnueabi-gcc –version

  cette commande doit retourner :

  arm-poky-linux-gnueabi-gcc (GCC) 4.9.2

III.2.4. Installation de la bibliothèque du robot :

La bibliothèque est déjà installée dans le robot, pour l'installer sur ordinateur, on suit les instructions suivantes :

- Extraire la bibliothèque « ***libkhepera_2.X.tar.bz2*** » dans notre fichier de développement par

  tar -xjf libkhepera_2.X.tar.bz2 -C ~/khepera4_development.

- On peut recompiler la bibliothèque complète par les commandes :

  cd ~/khepera4_development/libkhepera-2.X

  make clean

  make all

III.3. Communication du robot avec le PC

Il existe trois méthodes (USB, Bluetooth, WIFI) pour assurer la communication avec le robot et avoir accès à la console. Vue les difficultés rencontrées lors de la configuration de la connexion WIFI, dans ce qui suit on explique les deux autres (USB et Bluetooth) qui ont fonctionné.

III.3.1. Communication via USB :

1. Brancher le câble USB à l'ordinateur et le coté mini-USB au robot.
2. Allumer le robot.
3. Sous Linux on utilise **minicom.** Pour l'installer sur Ubuntu, on copie la commande suivante dans le terminal :

   sudo apt-get install minicom
4. Ensuite on doit configurer minicom :

On introduit la commande ***minicom -s*** sur le terminal et on obtient le menu de Figure III.1.

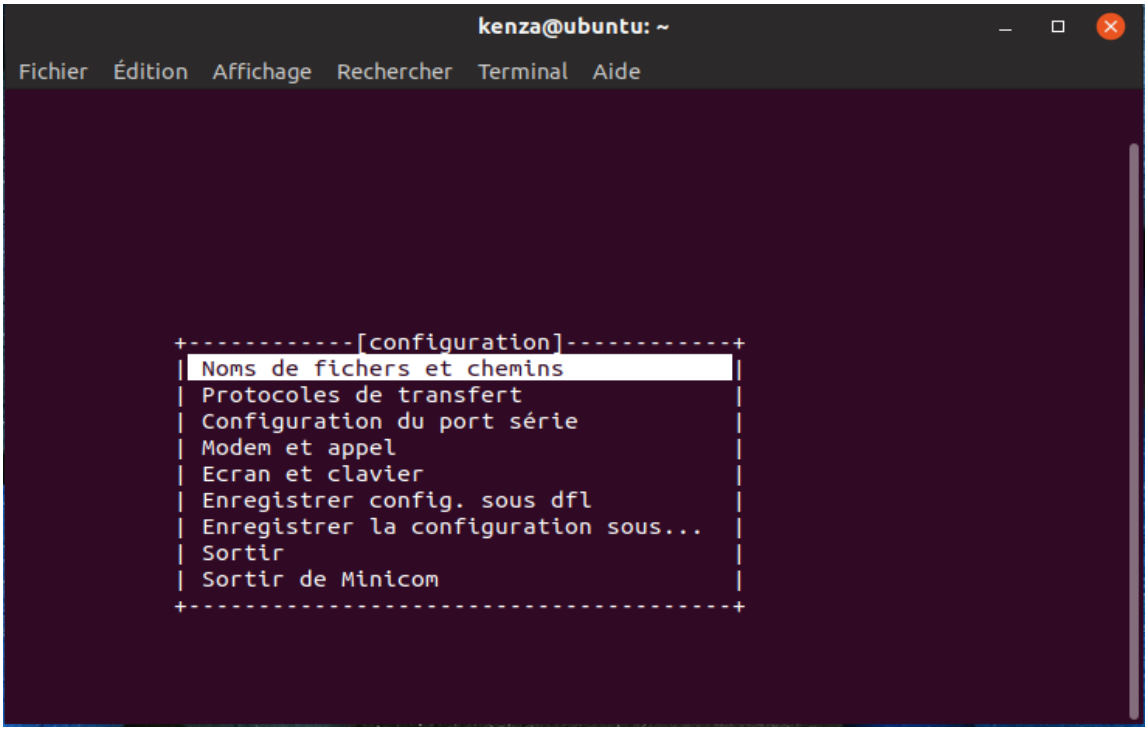


Figure III.1 **:** Menu de configuration du **minicom**

5. Puis on choisit « ***configuration port série*** »

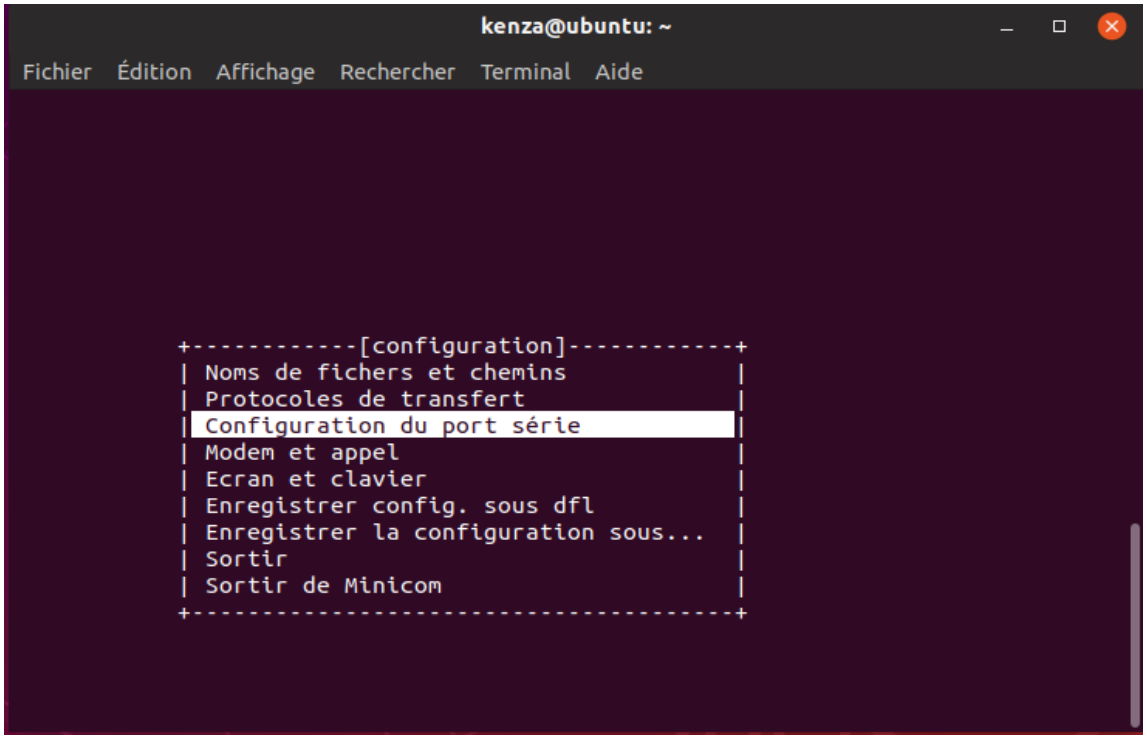


Figure III.2 : Minicom et configuration du port série

On ajuste les paramètres du port selon la configuration de Figure III.3

Figure III.3: Configuration des paramètres du port minicom

6. Enfin, on enregistre cette configuration, par défaut, sous « dfl » selon FigureIII.4.

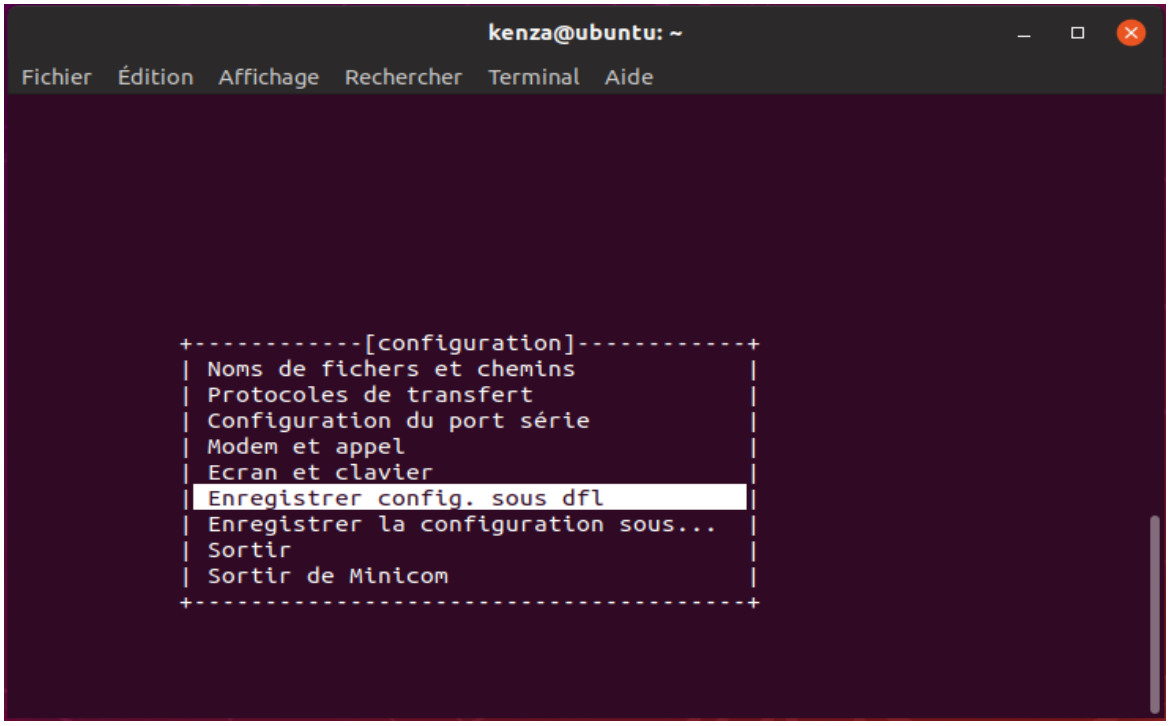


Figure III.4 : Enregistrement des paramètres USB

Une fois ces étapes sont achevées, on attend l'établissement de la connexion entre le robot et le PC. Une fois s'est faite, on accède au robot en utilisant :

Nom d'utilisateur : ***root***

Mot de passe : pas de mot de passe, on appuie sur « echap ».

Par default, on se retrouve dans le répertoire **home/root**. On peut exécuter un programme par l'instruction suivante : `./nom du programme`(./khepera4_test)

III.3.2. Communication via Bluetooth :

- Comme première étape, on doit installer le package « Irzsz », qui contient les programmes de communication avec l'instruction suivante :

```
sudo apt-get install lrzsz
```

- Puis on installe aussi le manager de bluetooth en Linux « blueman » par :

```
sudo apt-get install blueman
```

- On allume le robot.

- On ouvre blueman et on cherche les nouveaux appareils. Le robot va apparaitre comme « ***khepera4-1242*** ». On réalise la connexion selon la procédure A2. Et on configure le port Bluetooth exécutant les étapes suivantes :

1. On ouvre le minicom mais cette fois-ci avec la commande : ***`sudo minicom -o`*** et on procède à la configuration.
2. On sélectionne « configuration du port série »

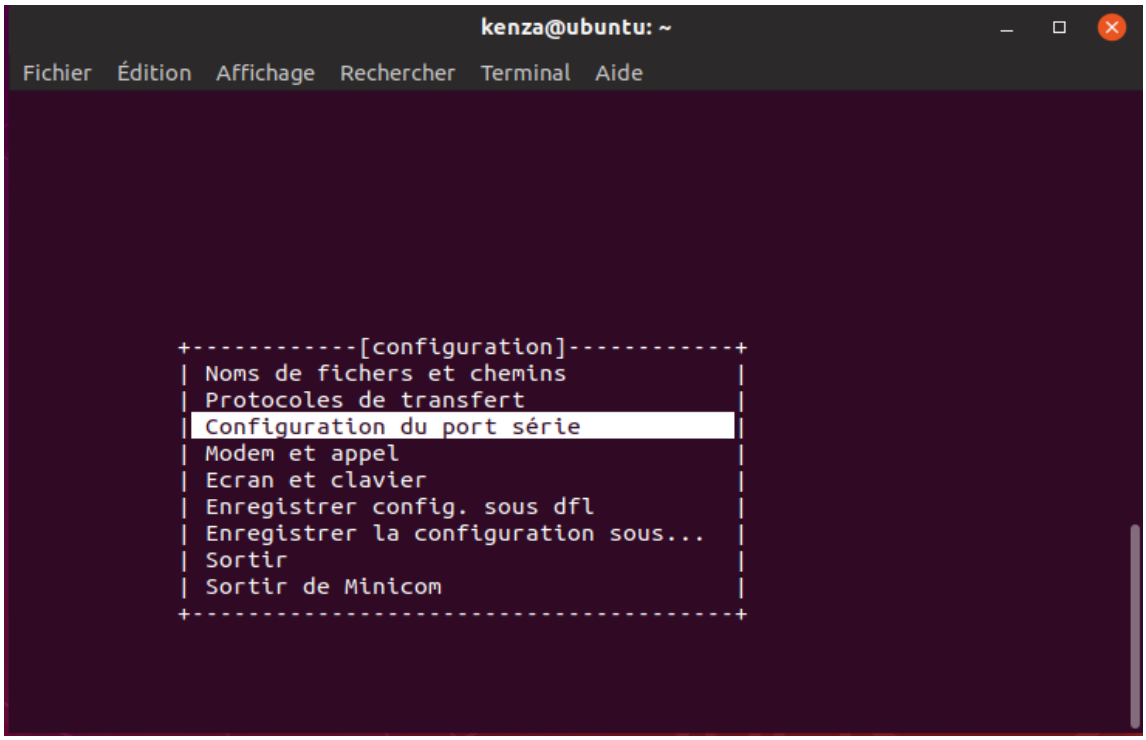


Figure II.5 : Minicom, configuration port série pour Bluetooth

3. Puis, on fait la configuration du port selon las paramètres de Figure III.6.

Figure III.6 : Configuration des paramètres de Bluetooth.

4. Enfin, on enregistre avec la commande « Enregistrer la configuration sous… » sous le nom '***Bluetooth***'

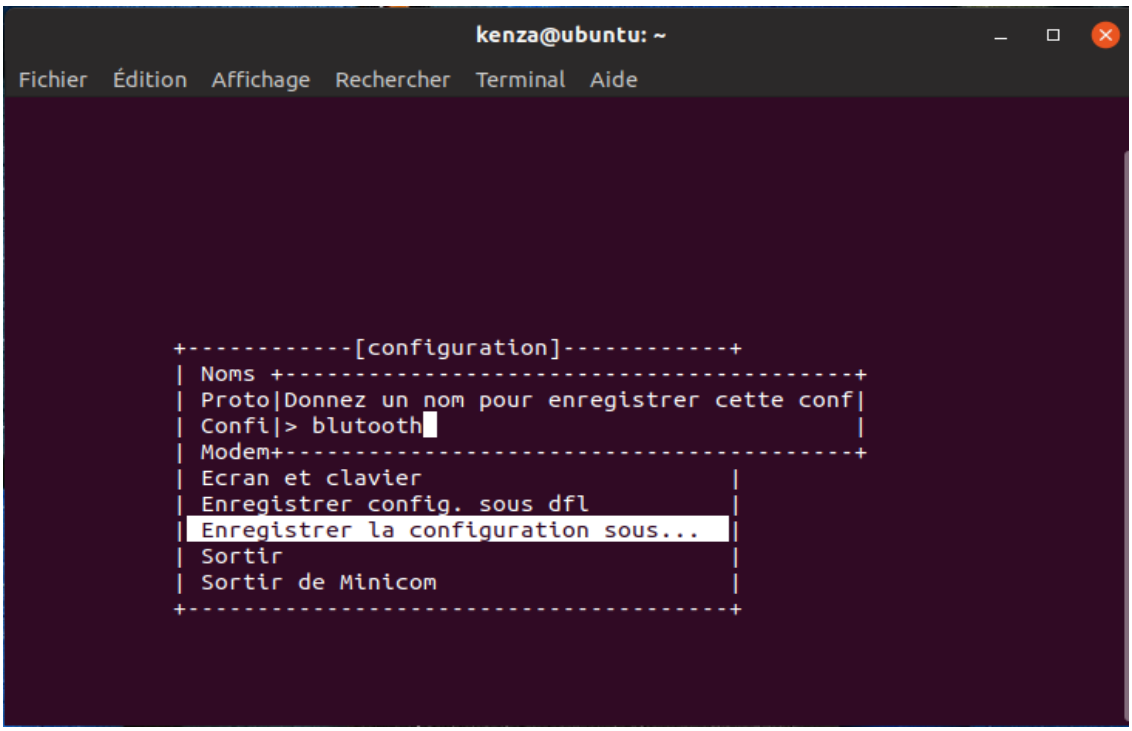


Figure III.7 : Enregistrement des paramètres Bluetooth

On pourra après lancer cette configuration avec : ***sudo minicom –o bluetooth***

On appuie sur la touche « echap » et on se retrouve dans le robot :

Login : root

Mot de passe : « echap »

On est dans la console du robot.

III.3.3. Arrêt du robot :

Lorsque on est connecté à la console du robot, il est préférable de faire un bon arrêt que d'éteindre le robot, pour s'assurer que les fichiers qui sont ouverts vont être fermées et enregistrés correctement. Ceci ce fait en exécutant la commande : ***poweroff***.

Après 15s (si on est connecté au câble USB, on attend jusqu'à ce que la mise hors tension apparaisse), on peut éteindre le robot.

III.4 Programmation et première application

Le robot Khepera IV peut être programmé sous Windows ou sous Linux. Sous Linux la programmation se fait avec langage C en utilisant Eclipse comme un logiciel, qui fait aussi la compilation.

La création, la rédaction et la compilation du premier programme se fait selon la procédure de l'annexe A1[9]. Il s'agit d'inclure toutes les bibliothèques nécessaires par le cross-compiler. Après l'édition du programme et sa compilation avec Eclipse ,on cherche le fichier exécutable et l'envoie au robot selon la procédure exposée en annexe A3.

Dans le programme primitif, on a utilisé une des fonctions intégrées dans le robot afin d'avoir la position du robot. C'est une première tentative de faire marcher le robot en ligne droite vers l'avant en essayant d'éviter les obstacles.

Dans ce premier test, on a calculé les vitesses des roues en fonctions des écarts entre les positions réelles des roues et les positions désirées (erreurs de position) avec un paramètre de proportionnalité Kp= 0.03. Les vitesses sont envoyées aux moteurs en nombres entiers impulsions/sec. On a nommé les positions désirés « lpt » et « rpt » pour la roue gauche et la roue droite respectivement (lpt : left position target et rpt : rignt position target).

Les positions réelles des roues « p_l » et « p_r » sont lues en temps-réel, par la fonction « ***kh4_get_position*** » :

Kh4_get_position(&p_l,&p_r,dsPic) ;

La portion du programme qui calcule les vitesses et qui les normalisent à 120 est donnée par :

```
lst=0.03*(lpt-p_l);

if(lst>120) {lst=120;};
if(lst<-120) {lst=-120;};

rst=0.03*(rpt-p_r);

if(rst>120) {rst=120;};
if(rst<-120) {rst=-120;};
```

Ces vitesses sont envoyées aux moteurs par la fonction « kh4_set_speed »

```
left_speed = (lst);
right_speed = (rst);

kh4_set_speed(left_speed ,right_speed ,dsPic );
```

Nb: la distance est calculée selon le périmètre de la roue 2πr.

Test 1 : Translation en ligne droite sans obstacles

On pose le robot dans une posture initiale. On réinitialise les encodeurs (positions réelles initiales p_l = 0 et p_r =0). On introduit les distances désirées des roues. Dans cet exemple on a pris 400 mm qui correspond à 58960 impulsions soit : lpt =58960 et rpt = 58960. Ce qui correspond à une ligne droite.

En absence d'obstacles, la série de 4 photos de Figure III.8 montre que le robot a bien atteinte sa destination.

Test 2 : Translation avec rotation à droite sans obstacles

Dans cet exécution, on a fixé la position désirée de la roue gauche à 400 mm et celle de la roue droite à 200 mm soit : lpt =58960 et rpt = 29480.

Le robot fait une ligne droite sur la première moitié (200 mm) du traget (photos 1 et 2 de Figure III.9). Puis, la roue droite s'arrête et la roue gauche fait la deuxième moitie (photos 3 et 4).

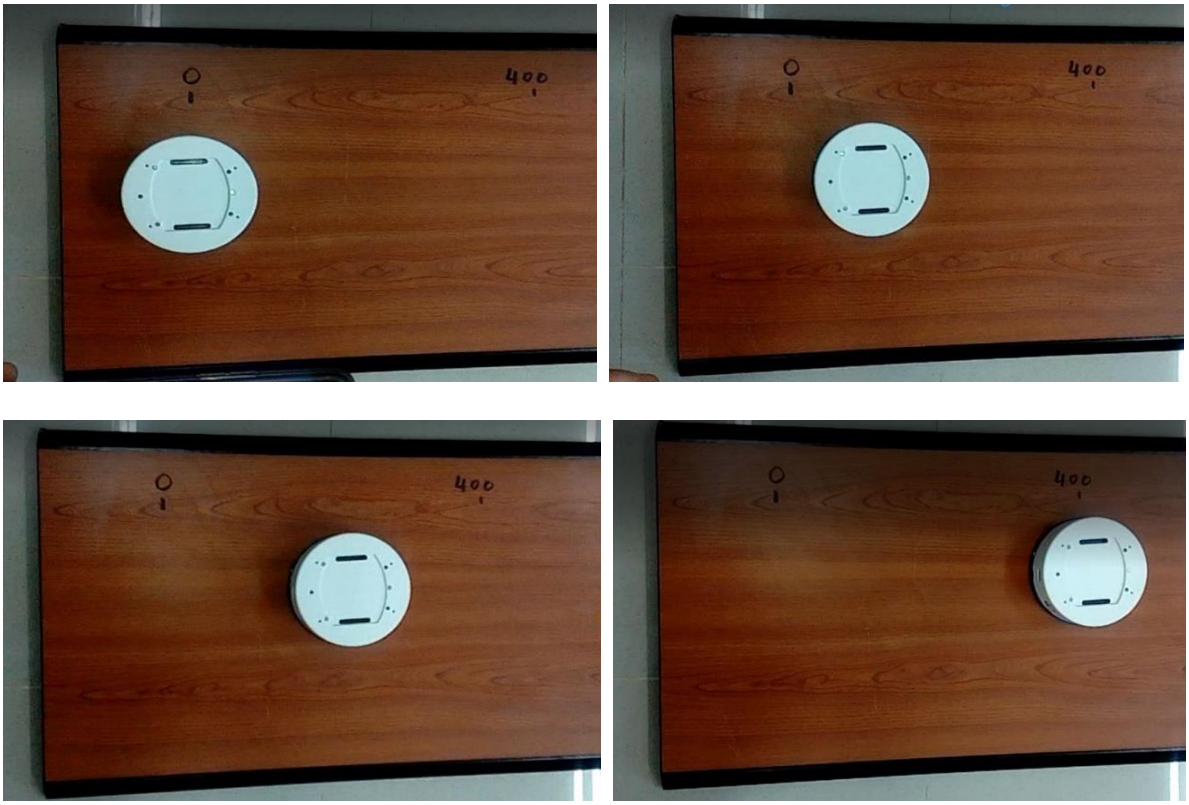


Figure III.8 : Translation en ligne droite en absence d'obstacles.

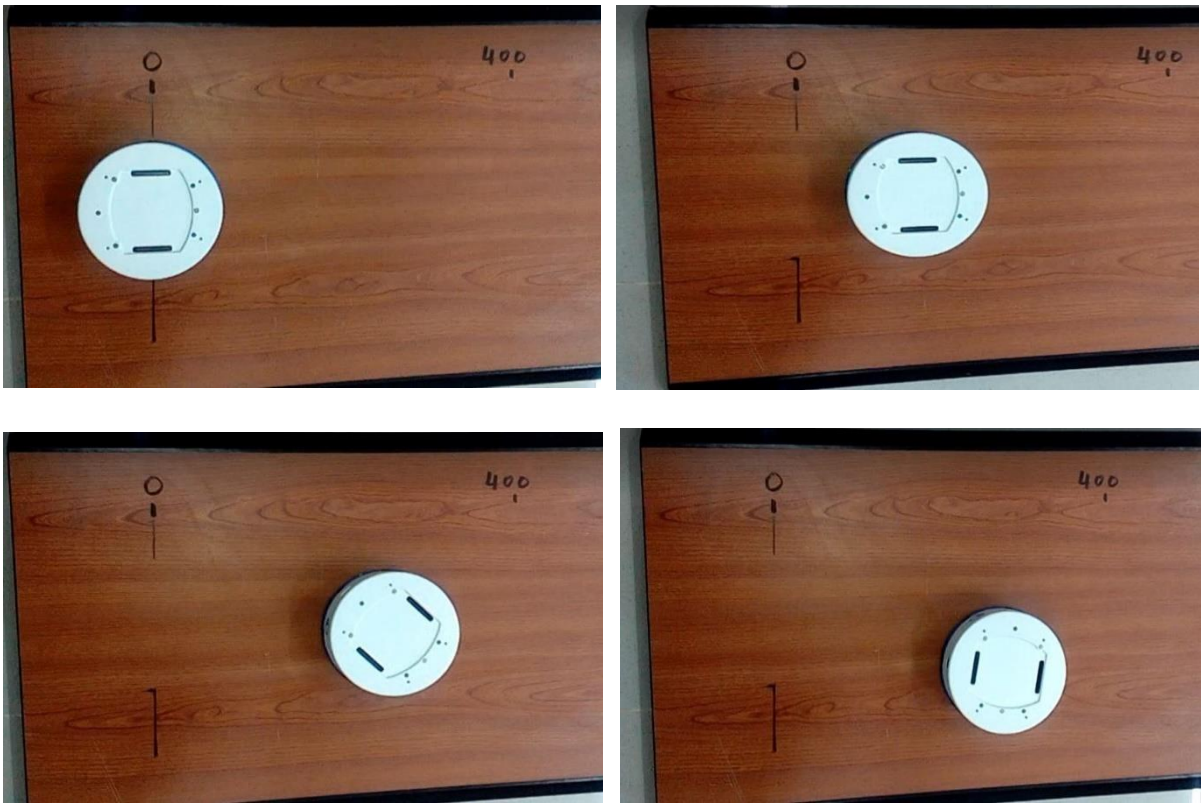


Figure III.9 : Translation et avec rotation à droite en absence d'obstacles.

Test 3 : Présence d'obstacles.

Ce test est la reproduction de l'exemple 1 (positions désirées des deux roues de 400mm), mais on perturbe le robot avec la main (obstacle). Les roues parcourent les distances 400mm, mais le robot n'arrive pas à la destination. Car, une partie du trajet est effectuée pour éviter l'obstacle, à gauche puis à droite (Figure III.10).

Cette expérience nous montre que le contrôle en position du robot ne peut pas se faire en utilisant une commande des positions des roues (distances exprimées en nombres de tours). Les distances doivent être exprimées dans le repère de base R0 associé au plan du travail (x,y).

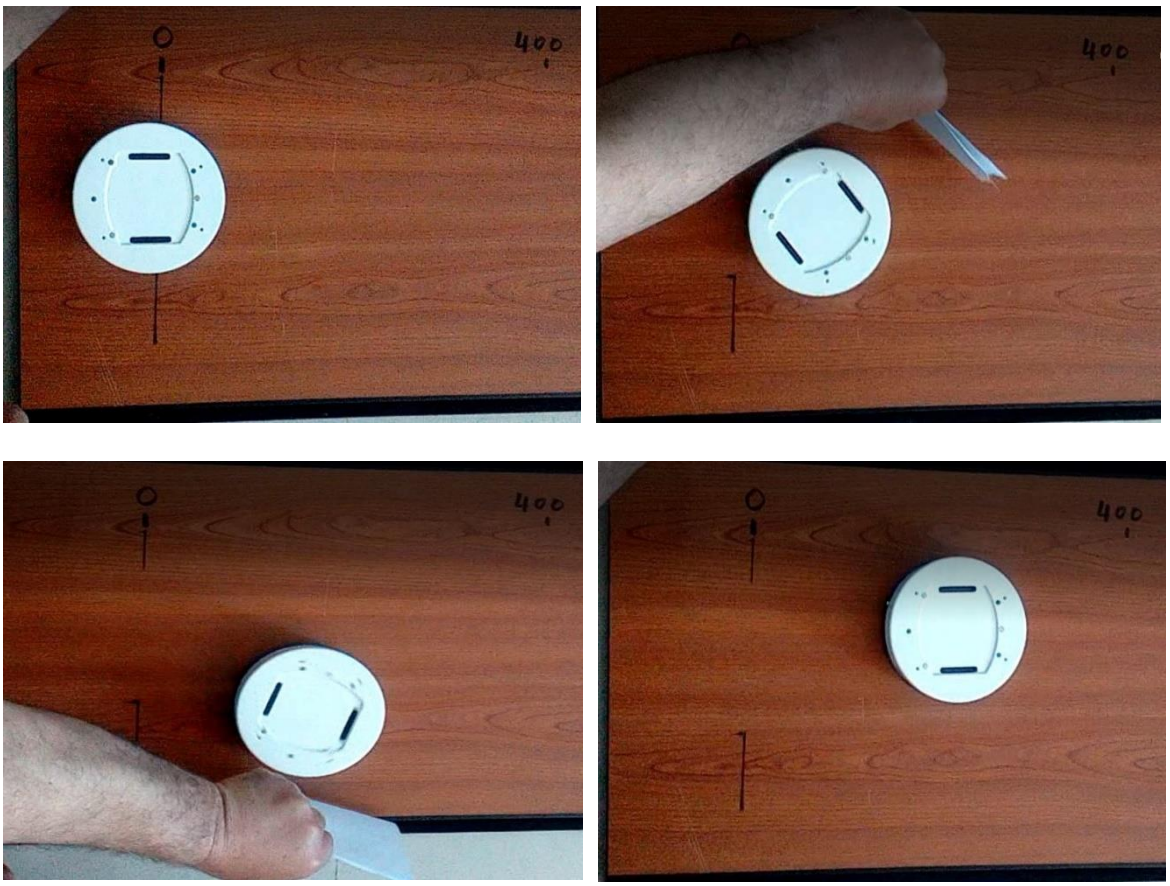


Figure III.10 : Echec d'exécution de trajectoires en présence d'obstacles.

III.5 Conclusion :

Ce chapitre est consacré principalement à la préparation du robot Khepera IV, en procédant à l'installation du software nécessaire et la configuration notamment des protocoles de communications (USB et Bluetooth), et la mise en marche de ce dernier.

La suite concerne une première application de la commande des déplacements du robot en utilisant la commande des positions des roues (distances exprimées en nombres de tours). Mais, la présence des obstacles rend cette approche inutilisable. Le prochain chapitre traite la commande en position du robot directement dans le repère de base R0 associé au plan du travail, en utilisant la technique de l'odométrie avec ses problèmes de calibrage.

Chapitre IV : Commande en position par odométrie du robot Khepera IV

IV.1. Introduction

Ce chapitre expose notre travail correspondant à deux applications : aller vers un point cible et suivre une trajectoire « waypoints ». On va expliquer la procédure et les moyens de programmation ainsi que les différentes étapes de réglage et d'amélioration de nos applications.

IV.2. Application 1 : Aller vers un point cible

Contrairement à l'application exposée à la fin du chapitre III, cette fois-ci le contrôle de position se fait directement dans le repère de base associé plan de travail. La commande se fait à base des écarts dx et dy exprimant la distance entre la position actuelle du robot et la position cible dans le plan de travail (x,y).

Soient la position cible $P_t = [x^*, y^*]$ et de la posture actuelle du robot P = [x,y, θ], en utilisant les équations du modèle cinématique d'un robot unicycle, on calcule les vitesses linéaire et angulaire en utilisant les coordonnées polaires :

$$\theta^* = \text{atan2}\,[(y^* - y), (x^* - x)] \quad \text{(IV.1)}$$

$$v = K_v .dist = K_v\,[(x^* - x)^2 + (y^* - y)^2]^{1/2} \quad \text{(IV.2)}$$

$$w = K_w\,(\theta^* - \theta) \quad \text{(IV.3)}$$

Où : θ est l'angle d'orientation actuelle du robot, θ^* est l'angle d'orientation finale désirée, *v* est la vitesse linéaire de translation du robot et *w* est sa vitesse angulaire (orientation).

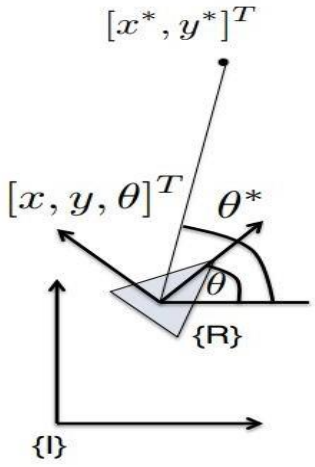


Figure IV.1 : Illustration des coordonnées du robot et du point cible dans le plan de travail.

Puis, on calcule les vitesses angulaires des roues du robot :

$$\dot{\varphi}_g = \frac{v - w}{R.Vmm} \quad \text{(IV.4)}$$

$$\dot{\varphi}_d = \frac{v + w}{R.Vmm} \quad \text{(IV.5)}$$

Où R est le rayon de la roue et Vmm est un facteur permettant d'exprimer les vitesses des roues en imputions/sec.

Le calcul de la posture actuelle du robot se fait avec la technique de l'odométrie basée sur l'intégration numériques des vitesses selon les axes (X,Y). Pour un pas d'échantillonnage Te de 10 msec et en utilisant les équations d'Euler :

$$x(k+1) = x(k) + Te * v * \cos(\theta) \quad \text{(IV.6)}$$

$$y(k+1) = y(k) + Te * v * \sin(\theta \quad \text{(IV.7)}$$

$$\theta(k+1) = \theta(k) + Te * \omega \quad \text{(IV.8)}$$

L'implémentation des équations sous Eclipse en langage C, en notations adaptées, est donnée le code de la table IV.1.

Table IV.1. Portion de programme concernant la commande par odométrie.

```
dx=xt-x;  dy=yt-y;
dist= sqrt(dx*dx+dy*dy);

beta= atan2(dy,dx);
      while(beta>PI){ beta-=2*PI;  };
      while(beta<-PI){ beta+=2*PI; };

v=Kv*dist*fwSpeed;

            if (v>fwSpeed){v=fwSpeed;};

            w=Kw*(beta-teta);
            lst=(v-L*w)/(vmm*R);
            rst=(v+L*w)/(vmm*R);

kh4_set_speed(left_speed ,right_speed ,dsPic );

            x=x+0.01*v*cos(teta)/Txy;
            y=y+0.01*v*sin(teta)/Txy;
            teta=teta+0.01*w/Tw;
```

La fonction ***kh4_set_speed(left_speed,right_speed,dsPic)*** permet de transmettre les vitesses angulaires calculées au microcontrôleur qui gère les boucles de régulation en vitesse des roues.

La quantité fwspeed représente la vitesse maximale tolérée. Les constantes « kv, kw, Txy et Tw » sont des coefficients de calibrage qui seront ajustés pour améliorer le contrôle des mouvements du robot .

Vue les difficultés rencontrées durant l'implémentation et le calibrage de l'algorithme de l'odométrie, dans la suite de ce travail, le problème d'évitement d'obstacles n'a pas été considéré.

Test 1 : Translation selon l'axe X sans calibrage :

Dans ce cas, on a considéré (x*, y*)= (300mm, 0) et les valeurs initiales des paramètres suivantes : Txy=1, Tw=1, kw=10, et kv=1. De Figure IV.2, on constate que le robot s'arrête bien avant d'atteindre la destination (arrive à 55mm au lieu de 300 mm prévue).

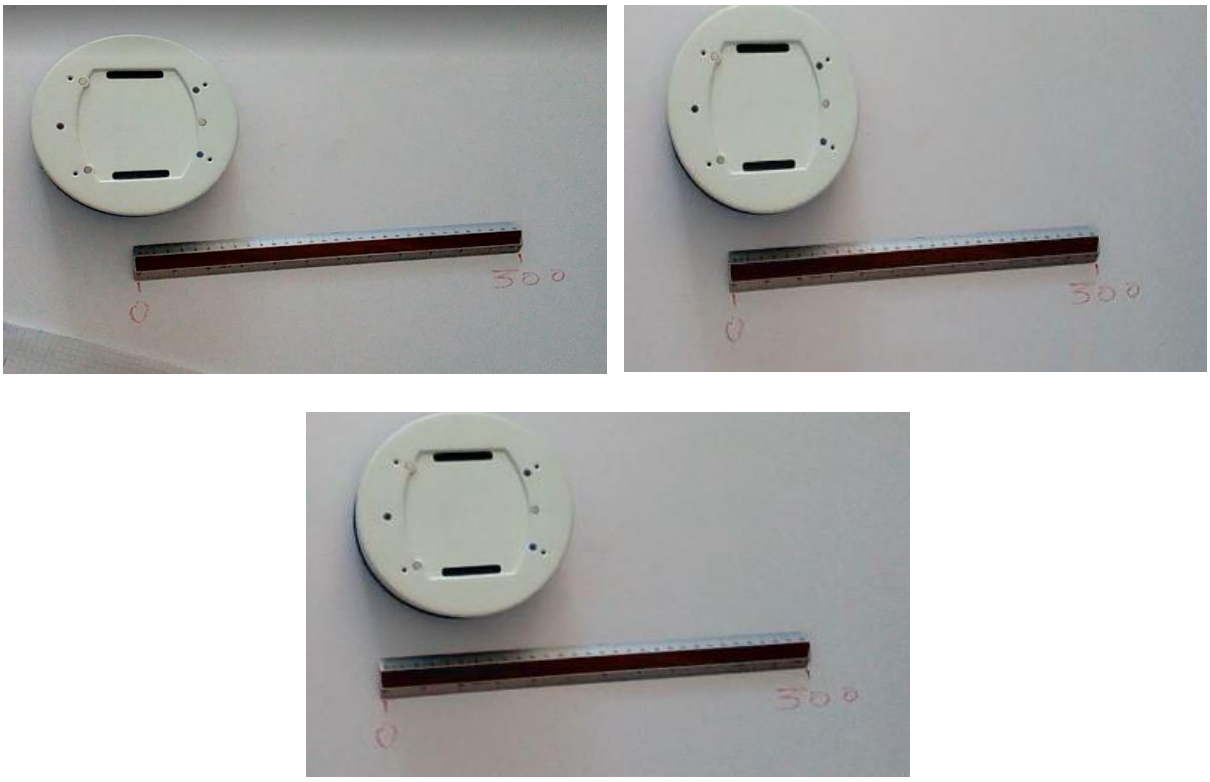


Figure IV.2 Translation selon l'axe X du robot sans calibrage.

Test 2 : Translation en diagonale dans le plan (X,Y) sans calibrage :

Dans ce cas on a posé (x*, y*)= (300mm, 300mm).

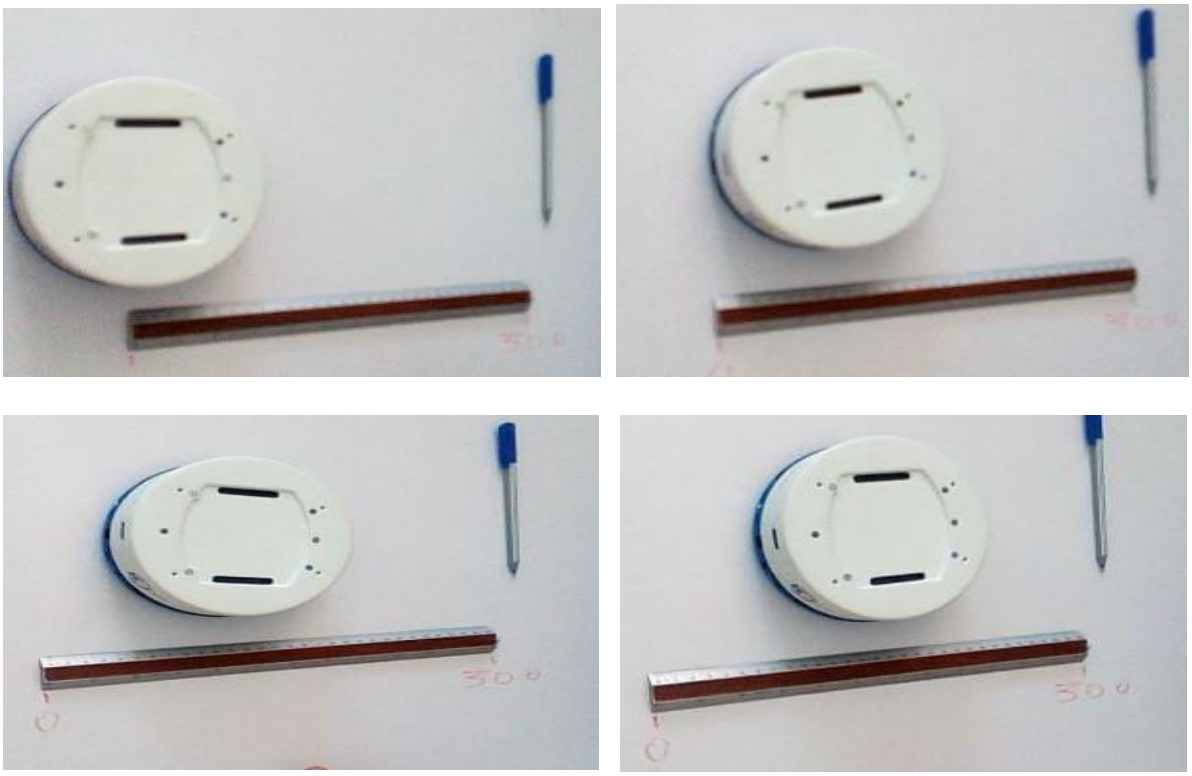


Figure IV.3. Translation dans le plan (X,Y) du robot sans calibrage.

Après une distance de 100 mm selon l'orientation initial (selon l'axe 'x') le robot devient instable avec des oscillations autour de l'axe z à gauche et à droite qu'on ne peut illustrer avec des photos.

On a constaté que les variables x et y intégrées par les équations d'Euler convergent vers leurs valeurs finales, mais les données réelles (distance parcourue par le robot) ne correspondent pas (inferieure). Alors nos équations d'intégrations des variables ont besoin d'un calibrage. On les a ralentis de telle sort à correspondre aux données réelles.

Test 3 : Translation selon l'axe X et calibrage :

Afin de faciliter le calibrage, on a procédé d'abord par l'étude de la translation selon l'axe X (x*= 300mm, y*= 0). Après plusieurs essais, on a trouvé que la valeur optimale pour Txy (coefficient de calibrage des variables x et y) est égale à **5.6**. de Figure IV.4, on voit bien que le robot arrive à sa destination parfaitement.

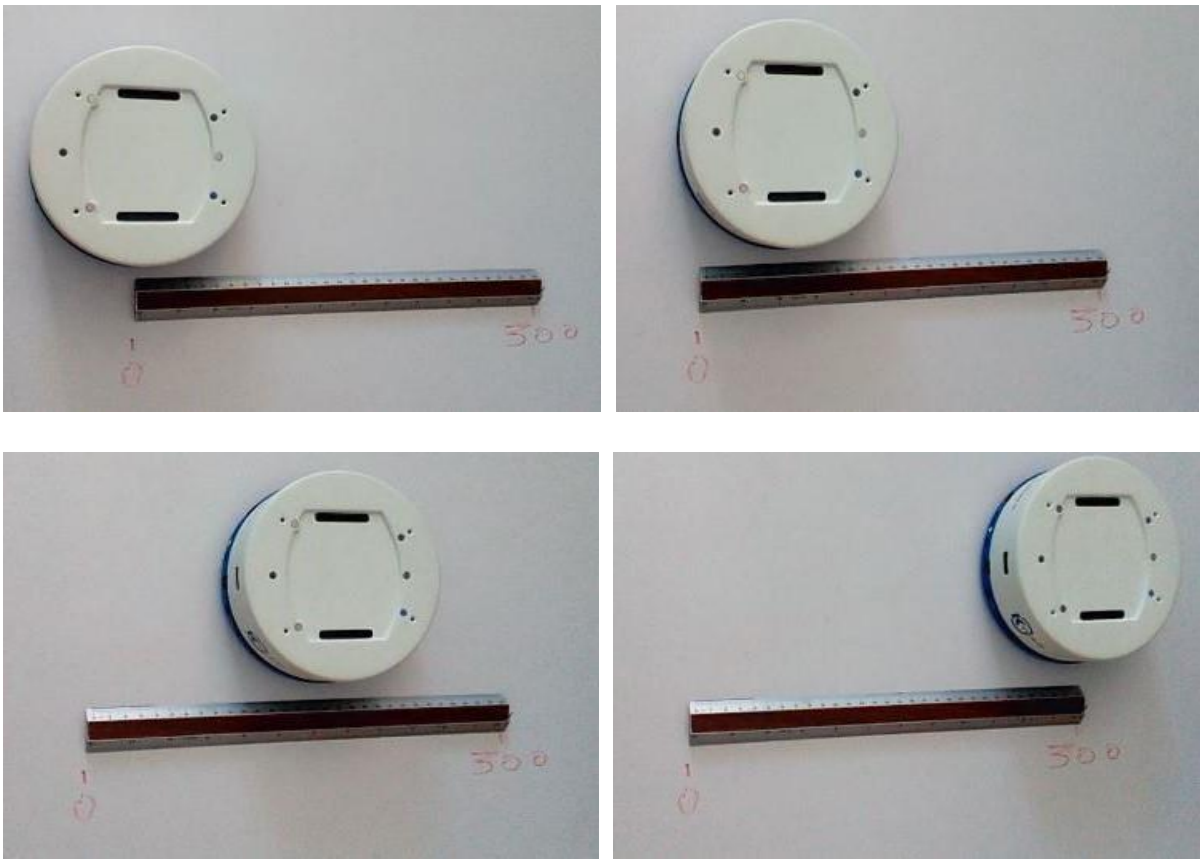


Figure IV.4 Illustration du mouvement de translation selon l'axe X avec calibrage.

Test 4 : Déplacements dans le plan (X,Y) sans calibrage d'orientation:

Déplacement dans le plan (X,Y) (x* = y* = 300mm) sans calibrage de l'orientation nous a offert une mauvaise surprise. De Figure IV.5, on constate que le robot n'arrive pas la destination, à cause du mauvais contrôle d'orientation. Pour cela on ajuste le Tw, afin de ralentir l'équation d'intégration de la variable 'θ'.

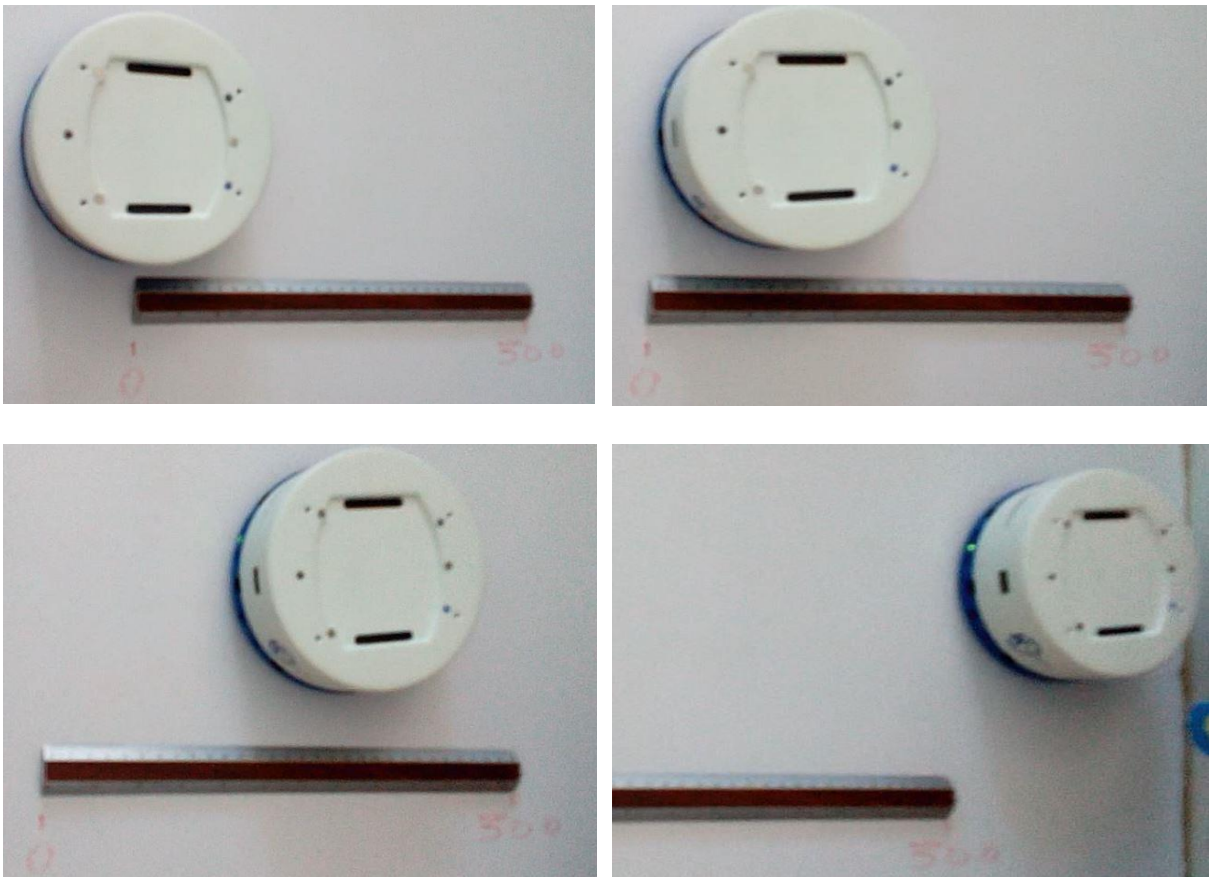


Figure IV.5. Illustration du mouvement de translation sans calibrage de l'orientation.

Test 5 : Déplacements dans le plan (X,Y) avec calibrage d'orientation:

Après plusieurs essais, la valeur optimale pour le facteur de calibrage Tw = 8.5. De la série de photos de Figure IV.6, on constate que le robot arrive à sa destination. Mais le problème, il lent. On veut augmenter sa vitesse, alors on joue sur la valeur du paramètre kv pour augmenter la vitesse linéaire.

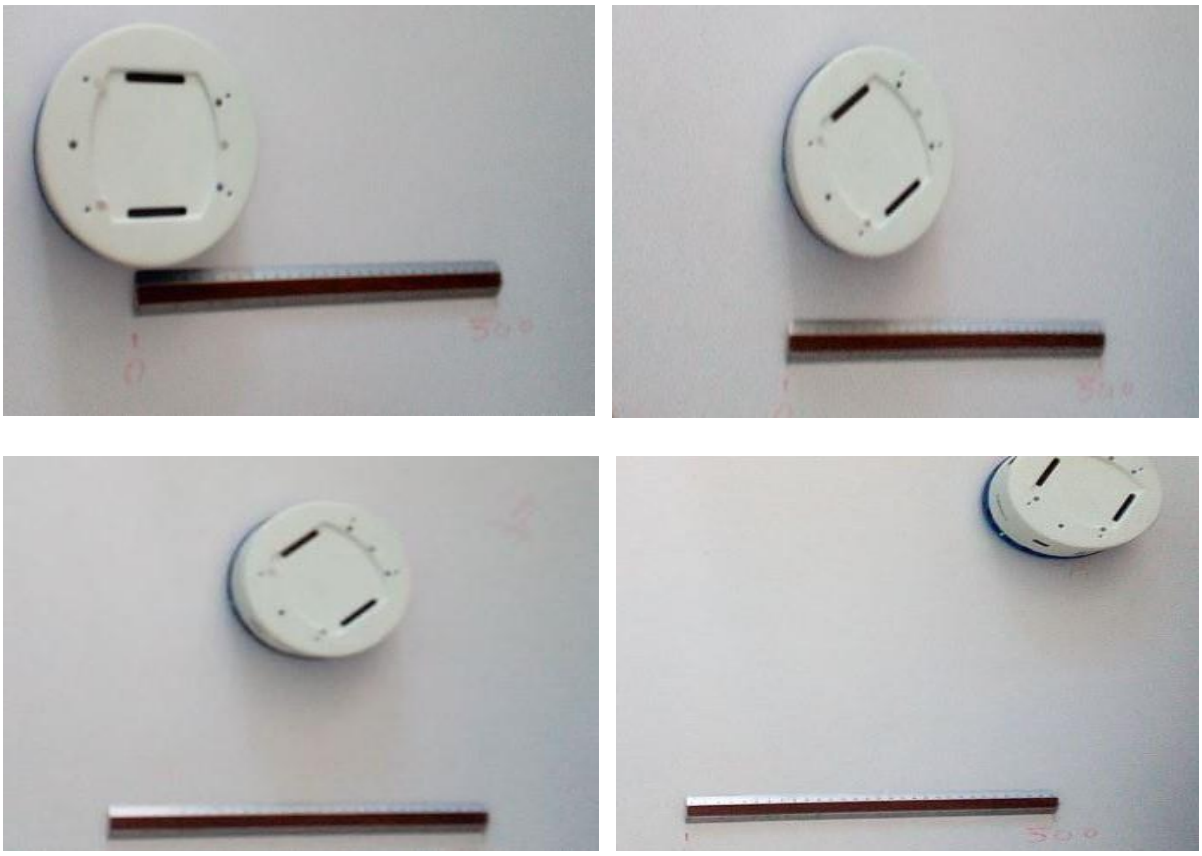


Figure IV.6. Illustration du mouvement de translation avec calibrage de l'orientation.

Test 6 : Amélioration de la vitesse

Lors de ce test, on a fixé Kv=10. La vitesse est devenue très grande. Ce qui a mis le robot en danger. Puisqu'il a dépassé le la cible prévue à grande vitesse. Alors, on a forcé l'arrêt du programme.

Pour bien contrôler la vitesse, on a déminué la valeur de Kv et au même temps on a augmenté la valeur de Txy pour ralentir les équations d'intégration. On a pris : Txy= 8.5 et kv=2. Comme illustré par Figure IV.7, le robot a bien démarré. Mais, en arrivant au milieu du trajet, il perd la route et oscille.

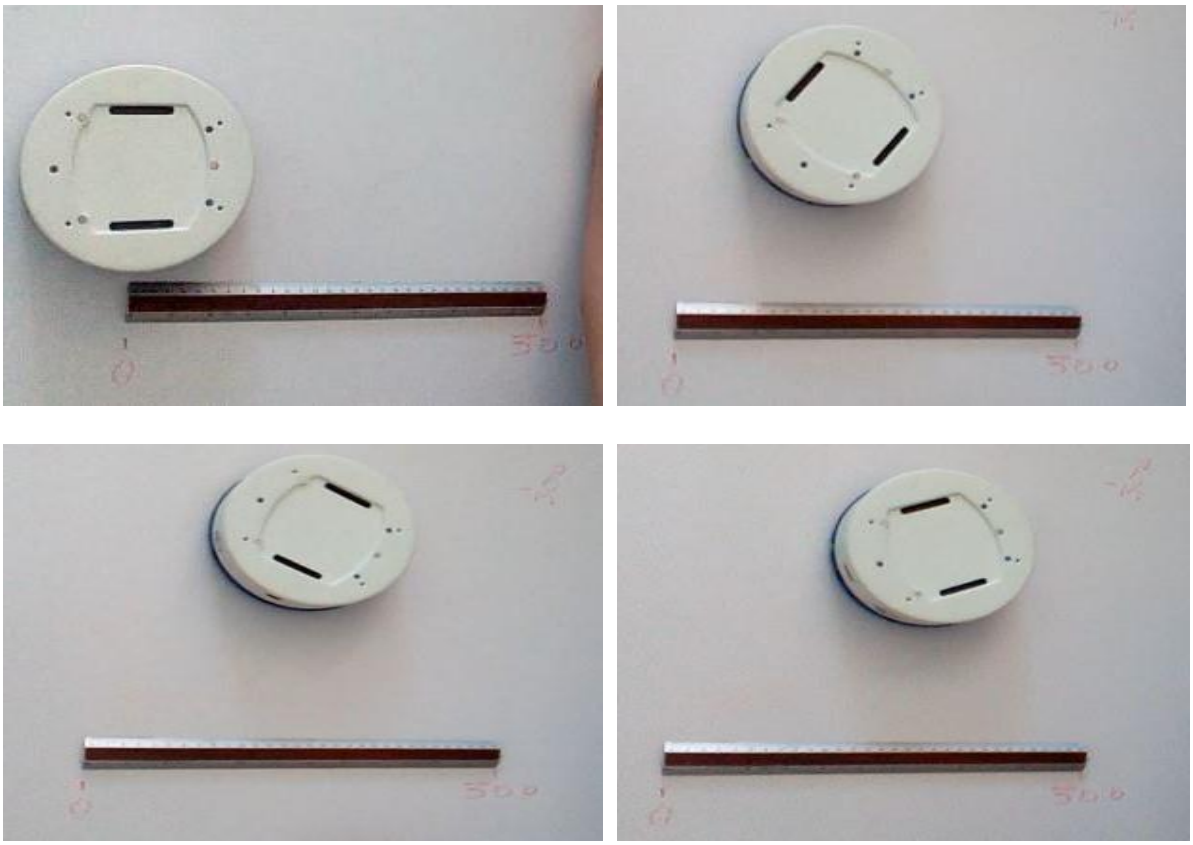


Figure IV.7. Amélioration de la vitesse de translation avant recalibrage de l'orientation et perte de contrôle au milieu de trajet.

En réalité dans ce programme, après augmentation de la vitesse de translation, on constate le paramètre Kw est faible (Kw=10 dans tous les tests passés) et n'est pas assez élevé pour amener le robot où on veut. Pour corriger l'orientation de robot, il faut augmenter la valeur Kw et modifier la paramètre Tw en conséquence. On a établi une relation empirique entre les deux paramètres : Tw= Kw/13.333. Après quelques tests, les valeurs finales des facteurs de calibrages qui permis de rendre le robot rapide et converge dans la bonne orientation sont :

Txy= 8.25 kv= 10 kw= 200 Tw=15.

De Figure IV.8, le robot est arrivé avec succès à la cible.

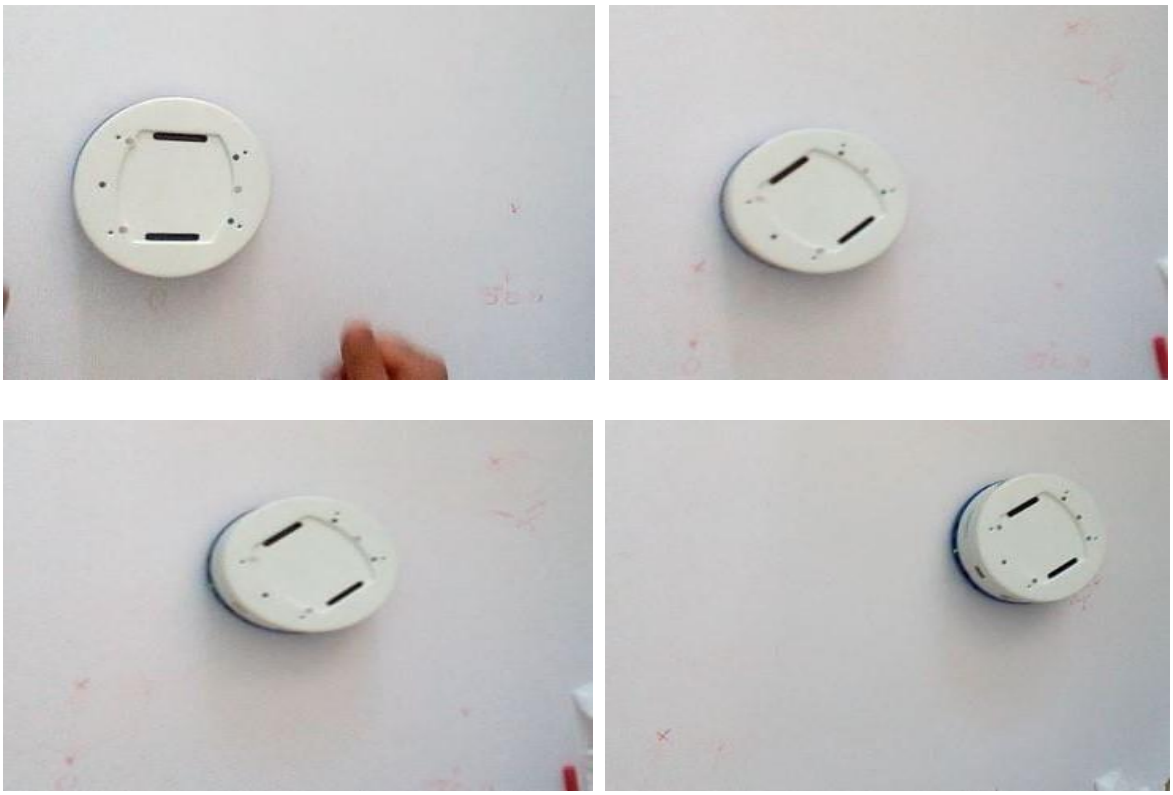


Figure IV.8. Amélioration de la vitesse de translation avec recalibrage de l'orientation.

IV.3. Application 2 : Suivi de trajectoires définie par des points via (way-points)

Comme extension au programme de convergence vers la cible, on a étudié le suivi de trajectoire indiquée par plusieurs nœuds (way-points). Une fois un point cible est atteint, il sert de point de départ pour le point suivant. Le principe consiste à exécuter la procédure de *« convergence vers la cible »* en boucle. Afin de transiter d'un segment au suivant, une décision, concernant l'arrivée au point les reliant, doit être prise. Le programme considère que le point est atteint, lorsque la distance < 0.01. Donc, la recherche du point suivant se fait quand la distance > 0.01. Ceci est illustré par la portion du programme de Table IV.2.

Table IV.2. Portion de programme concernant le contrôle des way-points.

```
        for (np=1;np<4;np++)
        {
                xt=mp[np][0];   yt=mp[np][1];
                        dx=xt-x;        dy=yt-y;        dist= sqrt(dx*dx+dy*dy);

printf("np= %d\n", np);
                while(dist>0.01)
```

Les points qui formes la trajectoire sont introduits et stockés dans une matrice « mp » de 2 colonnes (x,y). Dans cette première version du programme, on a réservé l'espace pour 10 points au maximum (dim(mp)=10x2). A titre d'exemple, on a introduit les points suivants :

(0 0) , (0 20) , (30 40) et (40 40).

Dans un premier test, on a utilisé les paramètres : Txy= 8.25, kv= 10, kw= 200, Tw=15.

Avec ces paramètres, le robot parcourt le chemin très bien, sauf que quand il arrive à la fin il commence à osciller.

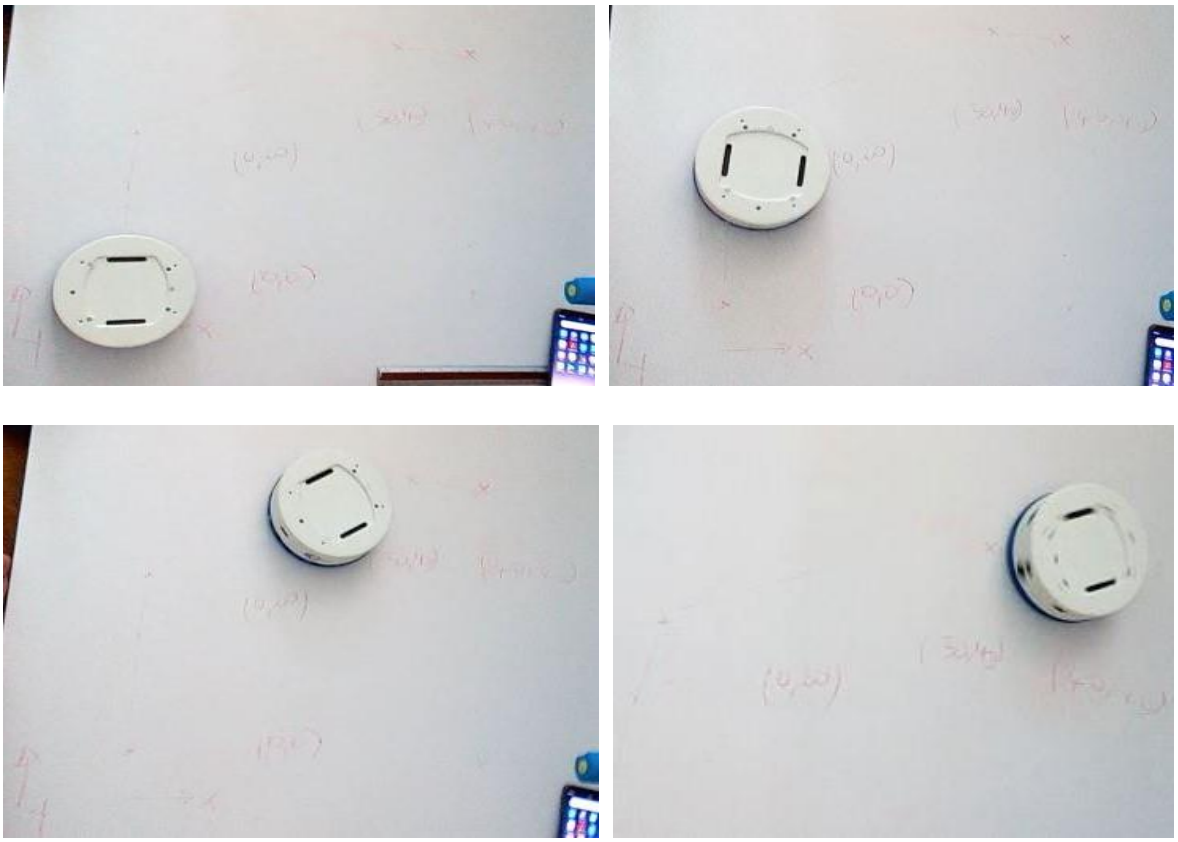


Figure IV.9. Exécution une trajectoire avec way-points avant recalibrage de l'orientation.

Alors on a pensé à revoir le paramètre Kw une autre fois. On a pris : kw=150 et Tw=11.25.

Le robot exécute la trajectoire mais avec une fausse orientation à la fin.

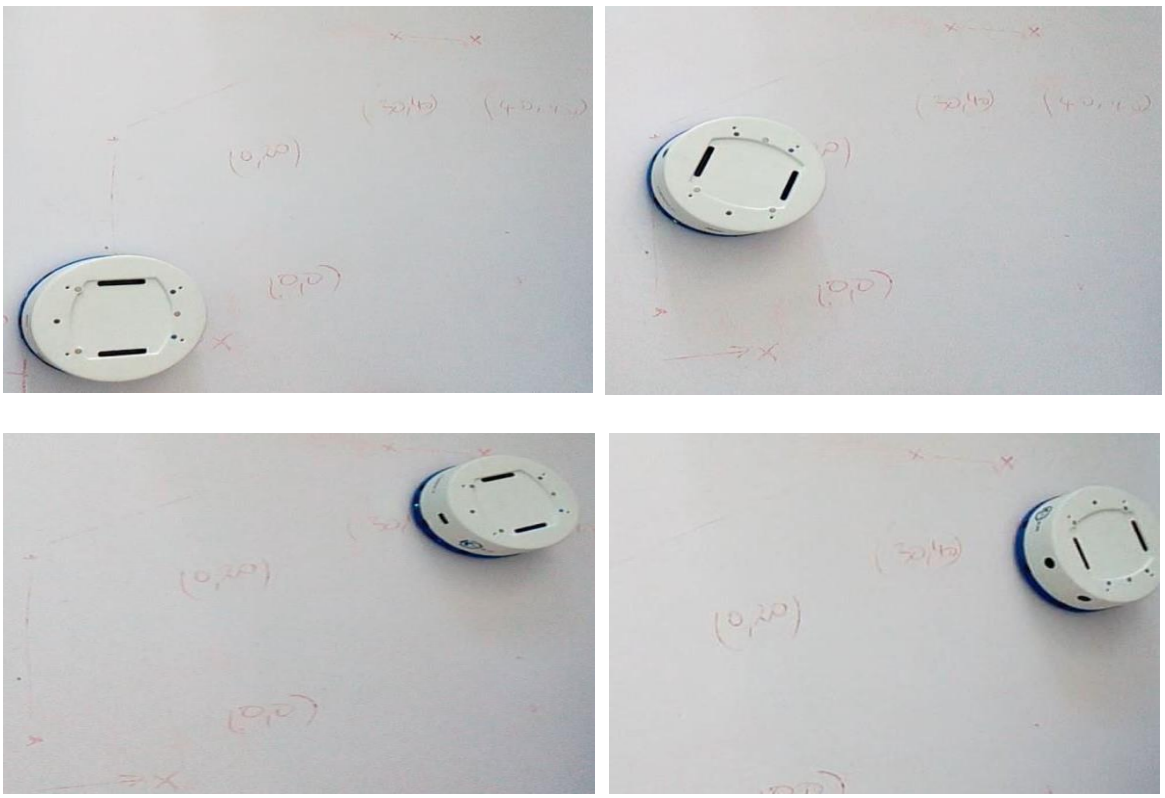


Figure IV.10. Exécution une trajectoire avec way-points avec recalibrage de l'orientation.

Donc, on était contraints de revoir notre relation de proportionnalité entre les deux facteurs Tw et kw. On a ajusté l'équation : Tw=Kw/13. Ce qui donne : Tw= 11.5.

Après cet ajustement, le robot fait le trajet et il arrive à la destination finale.

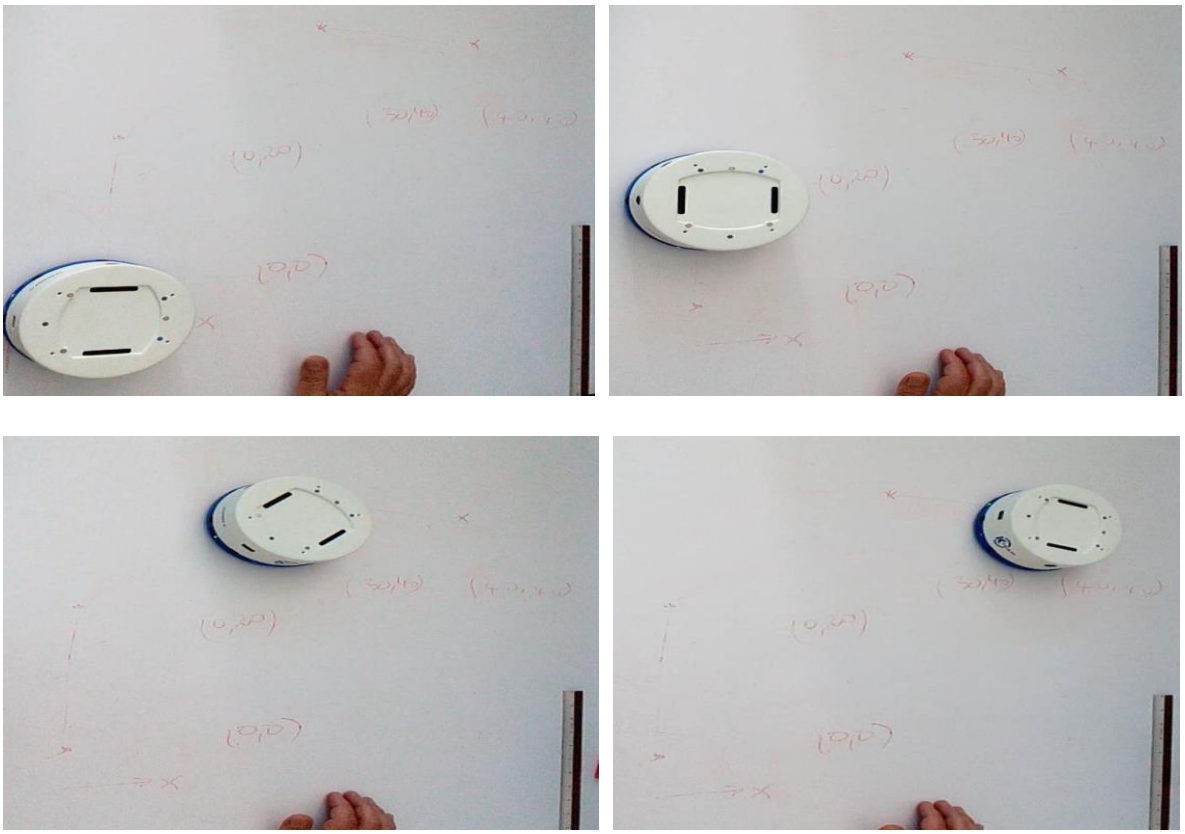


Figure IV.11. Trajectoire avec way-points après ajustement de la relation entre Tw et kw.

IV.4. Conclusion

Ce chapitre présente deux applications concernant le contrôle des déplacements du robot en absence des obstacles. On peut constater que quand on passe à l'implémentation pratique, le faussé se creuse entre les modèles mathématiques et la réalité. Le calibrage et l'ajustement des différents paramètres, introduits pour surmonter cette difficulté, constitue un travail fondamental. Le constat est que les résultats sont sensibles aux paramètres de conception. La robustification de l'algorithme et la prise en compte des obstacles sur le trajet constitue une bonne perspective pour ce travail.

Conclusion Générale

Notre projet consiste à étudier le robot mobile Khepera IV. L'objectif était d'abord de le configurer et de le mettre en marche, puis de se familiariser avec ce dernier et de développer quelques applications. Il s'agit principalement des tâches : aller vers une position désirée ou de suivre une trajectoire constituée de plusieurs points (way-points).

Après plusieurs essais et plusieurs réglages, le robot arrive à exécuter les tâches demandées. La réalisation de cette étude nous a pris tant de temps et d'efforts consacrés avant de voir le fruit de notre travail. Grace à cette étude nous avons enrichis nos connaissances en robotique mobile et nous avons appris de nouveaux savoirs dans le domaine de la programmation des systèmes embarqués et l'interaction avec le hardware.

Cependant, la mise en fonction du robot n'a pas été facile. L'accès au robot et la communication avec les méthodes sans fil expliquées dans le manuel n'ont pas été de grande utilité. Citant aussi que le robot Khepera IV exige un sol bien plat pour qu'il exécute les tâches commandées.

Nos expériences menées sur la convergence vers la cible et suivi d'une trajectoire ont été qualifié de succès :

- Dans la première application (aller vers un point cible), on a vue deux méthodes de programmation, la première a échoué à cause de la distance exprimé en nombre de tours des roues. Ce qui nous a poussé à adapter un algorithme qui utilise l'odométrie basée sur la distance calculée dans le plan (X,Y) du repère associé au plan de travail.
- La deuxième application (suivi de trajectoires), a été une continuation de l'application « aller vers une cible » qui consiste à exécuter la procédure de cette dernière en boucle, en allant d'un point au suivant.

Nous espérons qu'à travers cette étude, on a pu présenter d'abord un document utile pour l'utilisation future du robot mobile Khepera IV, avec d'avantages d'illustrations. On espère aussi que les applications exposées dans le dernier chapitre et les problèmes qui en découlent forment un substrat pour de nouvelles investigations. On cite en particulier :

- Etude plus approfondie de l'odométrie et son amélioration par sa robustification vis-à-vis des paramètres de conception et de calibrage et l'intégration de l'évitement des obstacles.
- Réessayer la commande du robot à distance via Wi-Fi.
- Commander le robot via l'interface Windows.

Cette étude reste une bonne expérience en termes d'apprentissage et de découverte sur le plan pédagogique, scientifique et personnel.

Bibliographie

Bibliographie

[1] Bali Chaher, *« Réalisation d'un robot mobile avec évitement d'obstacle et trajectoire programmée »*.Mémoire de master, Université Mohamed Khider Biskra 2012.

[2] David Filliat, *« Robotique mobile »*. Support du cours, École Nationale Supérieure de Techniques Avancées ParisTech, 2012.

[3] Jorge M. Soares, Inaki Navarro et Alcherio Martinoli , *« The Khepera IV Mobile Robot: Performance Evaluation, Sensory Data and Software Toolbox »* Proceedings of Robot 2015: Second Iberian Robotics Conference , pages767-781, 2015.

[4] Khireddine Med el Amine et Drihem Nadia, *« Contrôle d'un robot mobile. Thèse master académique »*. Mémoire de master, Université Abou Bekr Belkaïd de Tlemcen 2015.

[5] Laetitia Matignon, *« Introduction à la robotique »*.Support du cours, Université de Caen France, 2012.

[6] Mami Mohammed Amine**,** *« Coopération multi -robot pour le maintien de la connectivité »*. Thèse doctorat, université d'oran Ahmed Ben bella 2018.

[7] Manuel du robot Khepera IV, 2018.

[8] Rita Baddoura-Gaugler, *« L'homme et le robot humanoïde : Transmission, Résistance et Subjectivation. Psychologie »*. Support du cours, Université Paul Valéry - Montpellier III, 2013.

[9] Site official du robot Khepera IV : https://www.k-team.com/extensions-khepera-iv

[10] Slimane Noureddine, *« systèmes de localisation pour robots mobiles »*. Mémoire de master, université de Batna 2005.

Annexes

Annexe A1 : Création et compilation du premier programme :

Pour la création et la compilation du premier programme, on a suivi les étapes, indiquées dans le manuel d'utilisation du robot, pour intégrer les bibliothèques de fonctions, notamment celles concernant le fonctionnement du robot, :

1- Télécharger la version Linux de "IDE Eclipse pour les développeurs C / C ++"
2- Extraire le fichier programme Eclipse par la commande suivante :

tar -xzf eclipse-cpp-mars-2-linux-gtk.tar.gz –C ~/

3- La dernière version de Khepera toolchain (light ou full) doit être déjà installée. On peut vérifier en exécutant sur le terminal la commande suivante sur une seule ligne:

/usr/local/khepera4-yocto/build/tmp/sysroots/i686- linux/usr/bin/armv7a-vfp-neon-poky-linux-gnueabi/arm-pokylinux-gnueabi-gdb –version

qui devrait retourner : **GNU gdb (GDB) 7.6.2**

4- Exécuter Eclipse et dans la fenêtre " Workspace launcher ". choisir le dossier pour la sauvegarde du projet. Fermer ensuite la fenêtre de bienvenue.
5- Aller dans le menu "File => Projet C" ou C ++, et choisir un nom du projet (ex: premiertest). Puis appuyer sur le bouton suivant.
6- Dans la fenêtre qui suit "C Projet", cliquer sur le bouton " Advanced Settings ", et sur le "C / C ++ Build => Settings". Dans " Cross Settings " entrer :

Prefix : ***arm-poky-linux-gnueabi***

Path : ***/opt/poky/1.8/sysroots/i686-pokysdk-linux/usr/bin/arm-poky-linuxgnueabi/***

7- Sur "Cross GCC Compiler" Includes => Include paths, ajouter :

/opt/poky/1.8/sysroots/cortexa8hf-vfp-neon-poky-linux-gnueabi/usr/include

8- Sur « Miscellaneous », replacer "Other flags" par :

-c -march=armv7 -a -mfloat-abi=hard -mfpu=neon -mtune=cortex-a8 --sysroot=/opt/poky/1.8/sysroots/cortexa8hf-vfp-neon-poky-linux-gnueabi

9- Sur "the Cross GCC Linker", sur « Miscellaneous », replacer "Linkers flags" avec:

-march=armv7-a -mfloat-abi=hard -mfpu=neon -mtune=cortex-a8 --sysroot=/opt/poky/1.8/sysroots/cortexa8hf-vfp-neon-poky-linux-gnueabi

10- Sur “Cross GCC Linker => Libraries” dans "Libraries (-l), ajouter « khepera » avec le bouton « + » en haut à droite.

11- Dans "Libraries search path", ajouter:

/opt/poky/1.8/sysroots/cortexa8hf-vfp-neon-poky-linux-gnueabi/usr/lib

12- Choisir le menu Build “C/C++” à gauche. Choisir « Release » sous « configuration list » à droite. Répéter les étapes ci-dessus pour cette configuration à partir de l’étape 6.

13- Appuyer sur le bouton “Terminer”.

14- Aller dans le menu " File => New => Source file " choisir « premiertest.c » comme nom de fichier.

15- Fermer la fenêtre de bienvenue avec sa croix en haut à gauche.

16- Insérer dans le fichier premeirtest.c le code source C suivant:

```
#include <khepera/khepera.h>
int main(int argc, char *argv[]) {
        int rc;

        /* Set the libkhepera debug level - Highly recommended for development. */
        kb_set_debug_level(2);
        printf("LibKhepera Template Program\r\n");

        /* Init the khepera library */
        if((rc = kb_init( argc , argv )) < 0 )
        return 1;

        /* ADD YOUR CODE HERE */

        return 0;
}
```

17- Ensuite, compiler le projet avec le menu " Menu Project => Build-All "

=> Le fichier de sortie sera dans le sous-répertoire « Debug » ou « Release » du projet.

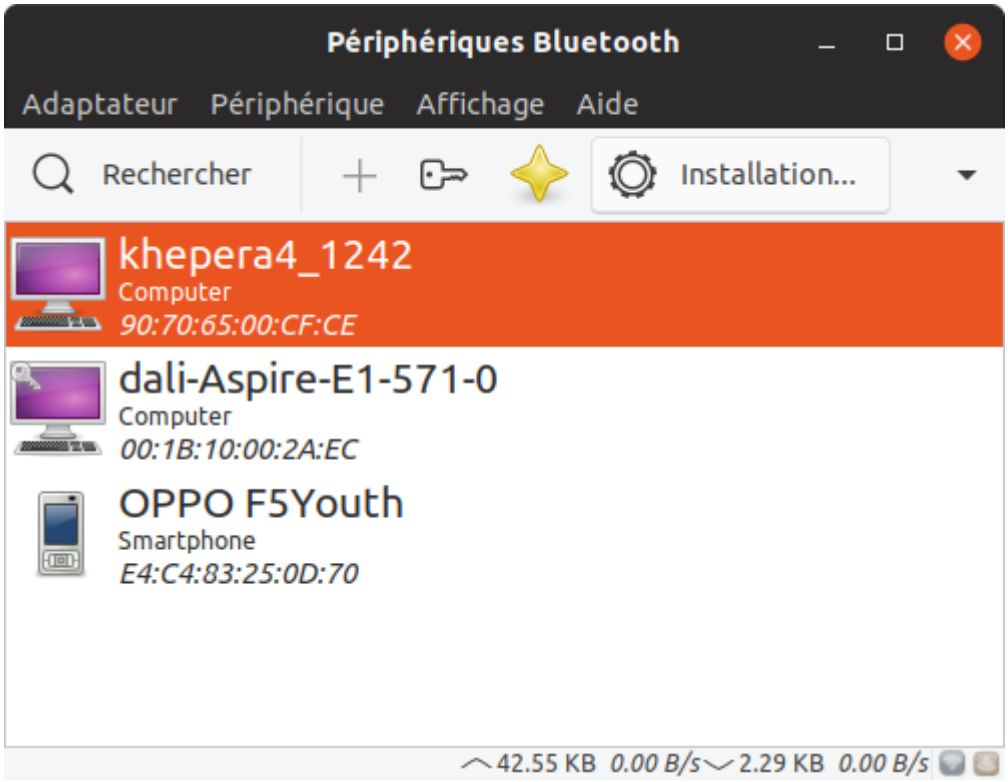
18- Transférer le fichier "premeiertest" à votre robot (voir annexe A3) et exécuter le fichier de programme avec la commande:

./premiertest

Annexe A2 : Connexion PC/Robot via Bluetooth:

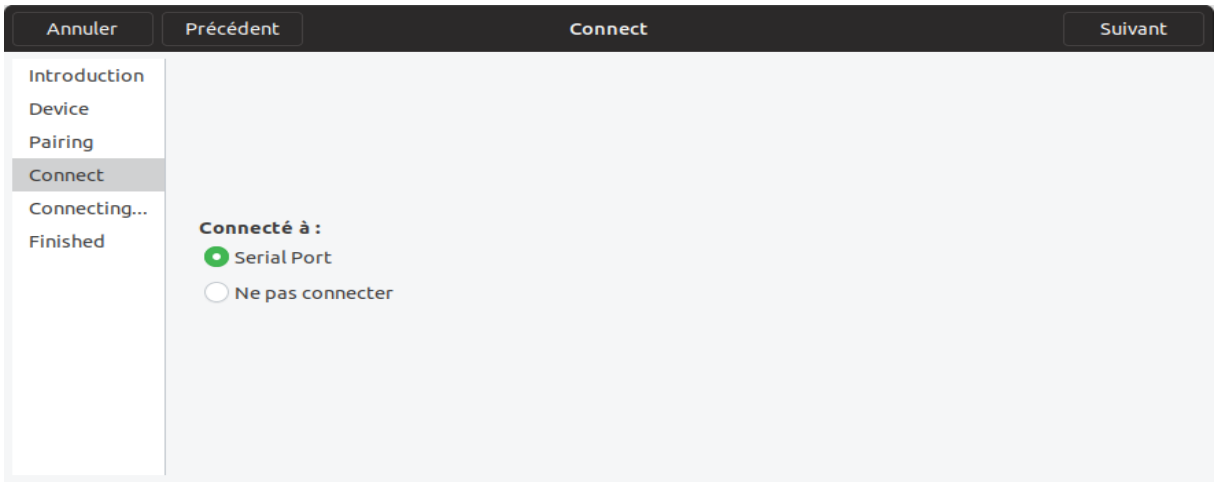
1- Ouvrir le logiciel blueman indiqué dans la figure suivante.

2- Chercher les nouveaux appareils. Le robot va apparaitre comme « ***khepera4-ABCD*** », ou ABCD et le numéro de série(1242 dans ce cas).



3- Cliquer sur le bouton droit de la souris, puis choisir configuration. La fenêtre suivante apparaitra ; choisir ***« Continuer sans association »*** puis cliquer sur ***« suivant ».***

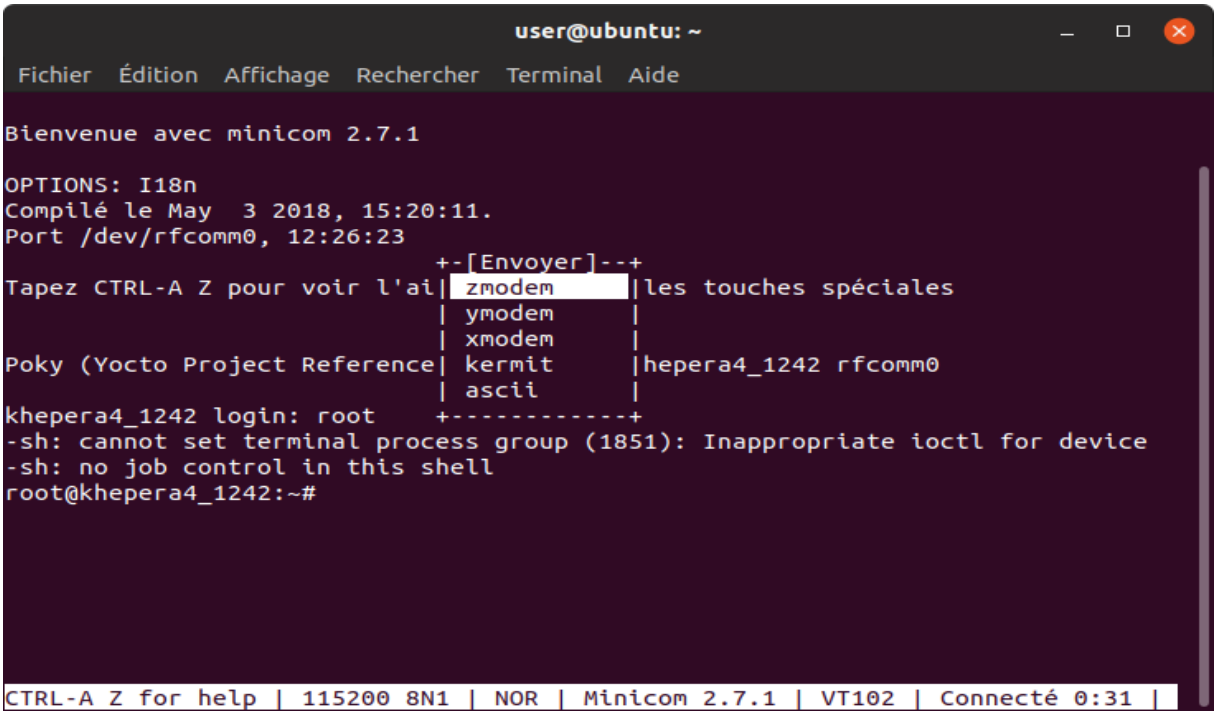
4- Choisir connecté à « Serial port », puis cliquer « suivant ».



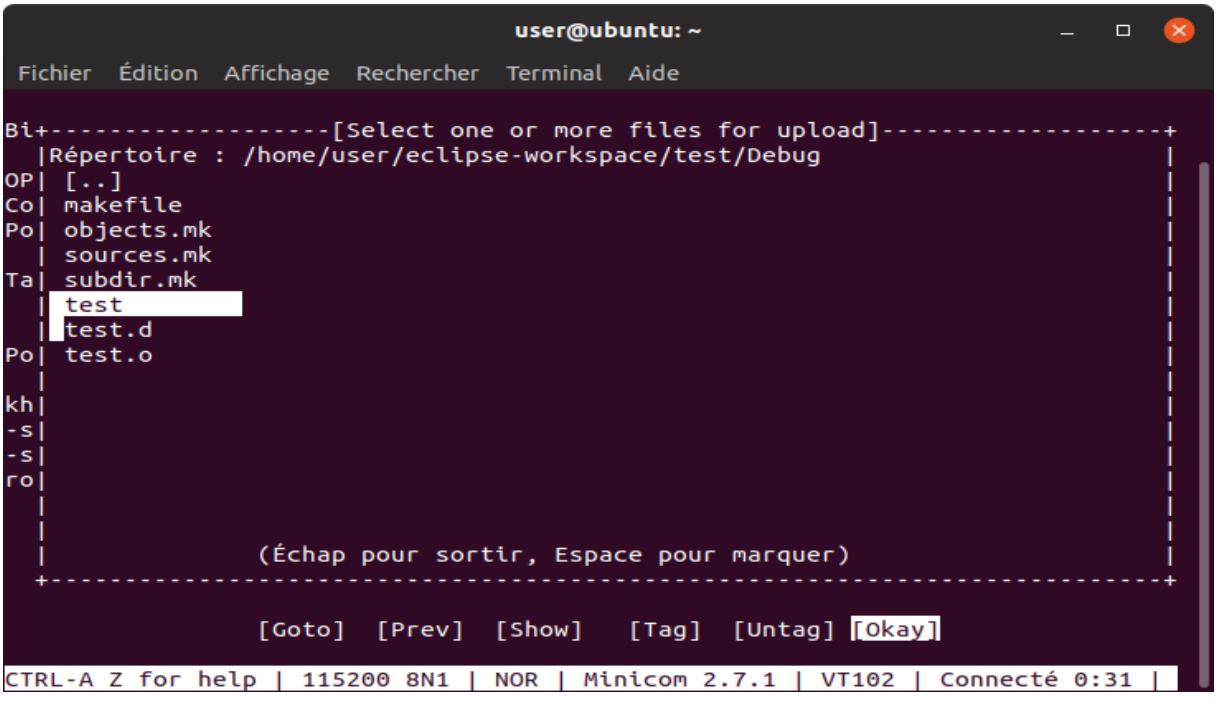
5- En finissant les étapes précédentes, la fenêtre ci-dessus apparaitra indiquant que l'appareil est connecté avec succès.

Annexe A3 : L'envoi d'un fichier du PC vers le robot :

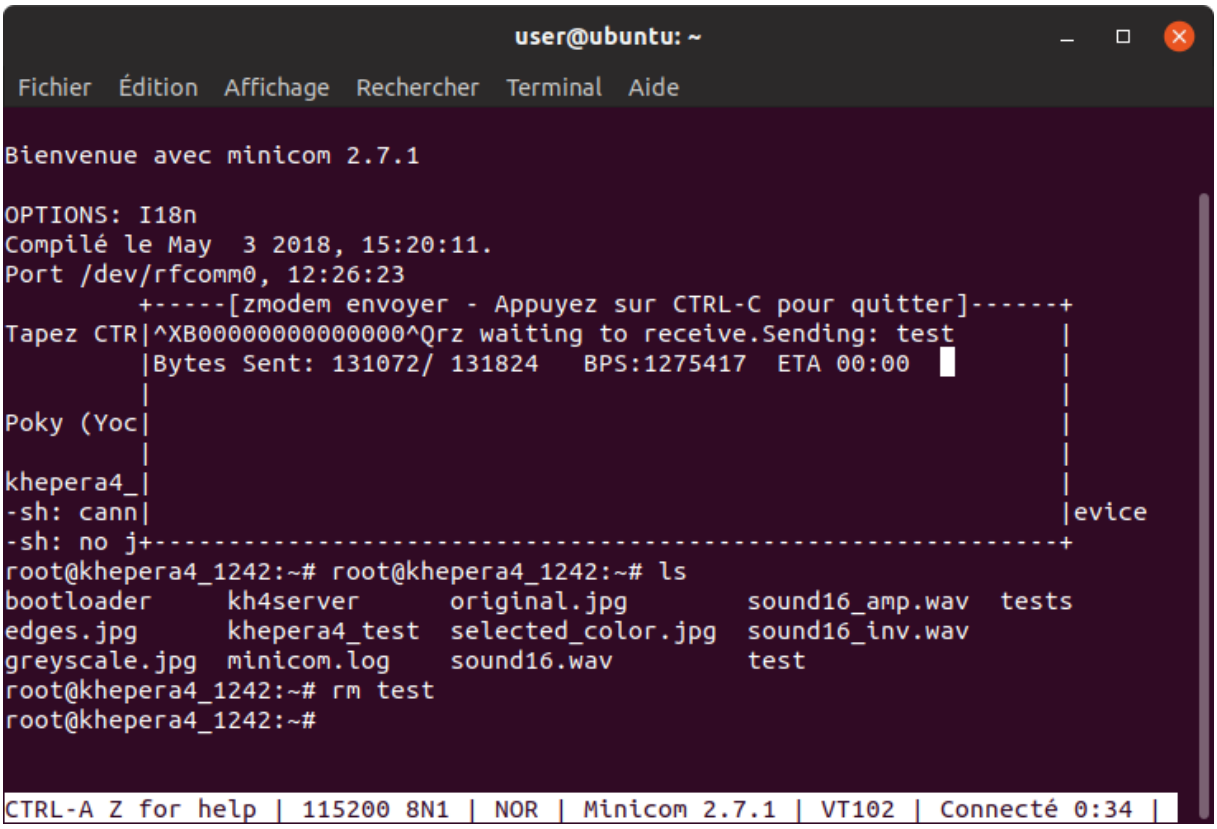
1- Sur la console Minicom, maintener les touches **« Ctrl+a »** et cliquer sur la touche **« s »**. La fenêtre ci-dessus apparaitra, choisir la commande **« zmodem »** et appuyer sur la touche **« entrée ».**



2- Sélectionner le fichier à envoyer en cliquant deux fois sur la touche **« espace »**, et sélectionner **« Okay »** , puis cliquer sur la touche **« entrée»** pour l'envoyer.



3- Le fichier est en cour d'envoi.



Résumé

Ce travail de master vise à étudier le robot mobile *Khepera IV*. Il s'agit d'abord de le mettre en marche et ceci par l'installation de tous les outils software nécessaires et la configuration du robot et en particulier les protocoles de communication et l'accès à ce dernier. Après la phase de familiarisation, on a procédé au développement de quelques applications telles que : aller vers une position désirée et suivre une trajectoire constituée de plusieurs points (way-points). Le travail a été effectué sous Linux en utilisant le langage C et le compilateur Eclipse.

Abstract

This work of master dissertation aims to study the *Khepera IV* mobile robot. The first stage is to prepare the robot and its development environment by installing the requires software and configuring the robot, especially, the communication protocols. After getting started, we developed some applications such as goto a desired goal position and trajectory following using waypoints. The work is performed under Linux using C code and Eclipse compilator.

ملخص

يهدف هذا العمل المعد للحصول على شهادة الماستر الى دراسة الروبوت Khepera IV.
الغرض الاول هو جعل الروبوت يمشي، وهذا عبر تثبيت جميع البرمجيات الضرورية وتكوين الروبوت، خاصة بروتوكولات الاتصال والوصول الى هذا الاخير. بعد مرحلة التعارف، تم تطوير بعض التطبيقات، مثل: الذهاب الى موضع مطلوب واتباع مسار متكون من عدة نقاط. تم العمل على هذا المشروع على نظام Linuxباستخدام لغة Cوالمترجم Eclipse.