REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE AKLI MOHEND OULHEDJ BOUIRA

FACULTE DES SCIENCES ET DES SCIENCES APPLIQUEE

DEPARTEMENT GENIE ELECTRIQUE

Filière : Electronique

Spécialité : Systèmes Electroniques Complexes

Mémoire de fin d'études

Pour l'obtention du diplôme de

MASTER

Réalisé par :

BOULILA Abdel Ouahab

Thème

Guidage d'un robot mobile autonome à travers le langage de programmation graphique : LabVIEW.

Soutenu le : 24/10/2017 devant les jurys composé de

Mr Kihal	President
Mr Hammouch	Examinateur
Mr Touafek	Examinateur
Mr.Messai Adnane	Encadreur
Mr.MoudacheSaid	Encadreur

Année universitaire : 2016_2017

Remerciements

Je remercie, avant tout, le **DIEU** le tout-puissant pour la volonté, la santé et la patience qu'il m'a donné durant toutes ces longues années d'études afin que je puisse arriver à ce stade.

Je tiens d'abord à remercier le Dr. MESSAI ADNANE et Mr. MOUDACHE SAID, qui m'ont offert l'opportunité de travailler sur ce thème. Je les remercie pour la grande confiance qu'ils m'ont accordée tout au long de ce projet. Leurs rigueurs scientifiques, leurs esprits visionnaires et leurs qualités humaines ont beaucoup contribué à la réalisation de ce travail.

Je remercie mes très chers parents qui ont toujours été là pour moi<<Vous avez tous sacrifiés pour vos enfants n'épargnant ni santé ni efforts, vous nous avez donné un magnifique modèle de labeur et de persévérance. Je suis redevable d'une éducation dont je suis fier>>.

Enfin je remercie tous mes amis que j'aime et toutes personnes qu'on connaisse de loin et du près pour leur sincère amitié et confiance et à qui je dois reconnaissances et nos attachements.

A tous ces intervenants, je présente mon remerciementmon respect et ma gratitude.

J'adresse mes sincères remerciements à tous les professeurs intervenants et toutes les personnes qui par leurs paroles leurs écrits leurs conseils et leurs critiques m'ont guidé, et ont accepté ma rencontre et ont répondu à mes questions durant ma recherche.

sommaire

Introduction Générale

INTRODUCTION GENERALE

Au cours de ces dernières années la robotique est devenue un domaine très important dans l'industrie à cause de sa grande utilité, l'utilisation des robots manipulateurs, mobiles ou fixes, au sein des installations nucléaires remontées à des décennies bien avant aujourd'hui. Plus particulièrement, en cas d'incidents graves caractérisés par une contamination massive, là où une installation nucléaire devient sans conteste un des lieux les plus hostiles qui soient pour l'être humain. L'endroit devient ainsi très dangereux pour les personnes chargées d'intervenir dans des situations pareilles. Et pour remplacer celles-ci, on met souvent au point des robots mobiles capables de se déplacer à l'intérieur d'une installation nucléaire donnée et d'y effectuer des tâches bien déterminées, certes simples mais pouvant s'avérer salvatrices dans la plupart des cas.

L'objectif du travail décrit dans ce manuscrit est de téléguider le robot mobile 'DaNi'2.0 de la société National Instruments dans un environnement inconnu et ce en utilisant essentiellement le langage de programmation graphique LabVIEW. En plus d'autres modules et utilitaires de programmation, LabVIEW sera installé dans un ordinateur de bureau qui sera amener à communiquer avec le robot via une connexion Ethernet WiFi. Le robot en question sera par la suite utilisé pour porter des détecteurs nucléaires et ce pour inspecter des zones susceptibles d'être touchées par la contamination radioactive.

Et de ce fait, le présent mémoire sera structuré de la manière suivante :

Dans le premier chapitre, on essayera de s'introduire au domaine de la robotique et on mettra l'accent sur les robots mobiles et les techniques de guidage utilisées pour contrôler ces derniers.

Lors du deuxième chapitre, on mettra en relief les aspects hardwares, plus ou moins détaillés du robot mobile DaNI 2.0. Il question de donner un aperçu sur la carte de contrôle embarquée sbRIO9632 et les principaux composants électronique qu'elle renferme. En passera également en revu les capteurs et les actionneurs utilisés lors de l'opération du guidage du robot. A la fin de ce chapitre, on établira un modèle cinématique simplifié du robot. Celui-ci va être utilisé lors des programmes de guidage développés dans ce travail.

Quant au troisième chapitre, celui-ci va être consacré à l'aspect software de ce travail. On introduira quelques notions de programmation graphique en utilisant le langage LabVIEW.

Par la suite, on utilisera les notions ainsi acquises pour connecter le robot à l'ordinateur du bureau et le préparer à recevoir le programme du guidage proprement dit.

Au cours du quatrième chapitre, on va expliquer, en pas-à-pas, le déroulement du programme élaboré et qui vise le contrôle et le téléguidage du robot mobile DaNI.

On terminera notre travail par une conclusion générale, là où on dressera un bilan sur l'ensemble des tâches réalisées et les extensions futures qui peuvent être ajouté à cette première contribution.

Chapitre I

Généralités sur les robots mobiles

I.1.Introduction

L'utilisation des robots manipulateurs, mobiles ou fixes, au sein des installations nucléaires remonte à des décennies bien avant aujourd'hui. Plus particulièrement, en cas d'incidents graves caractérisés par une contamination massive, là où une installation nucléaire devient sans conteste un des lieux les plus hostiles qui soient pour l'être humain. L'endroit devient ainsi très dangereux pour les personnes chargées d'intervenir dans des situations pareilles. Et pour remplacer celles-ci, on met souvent au point des robots mobiles capables de se déplacer à l'intérieur d'une installation nucléaire donnée et d'y effectuer des tâches bien déterminées, certes simples mais pouvant s'avérer salvatrices dans la plupart des cas.

Dans un premier temps, et dans le cadre de la robotique de mesure et d'intervention en milieux hostiles, on se propose d'explorer ce domaine en exploitant (*commandes du mouvement et algorithmes associés*) un robot mobile déjà préconstruit par NI : National Instrument, en tant que prototype pédagogique. Les techniques acquises à travers cette étape nous permettrons par la suite de réaliser notre premier prototype de robot mobile, se déplaçant sur roues dans un terrain plat non accidenté, prévu pour mesurer des doses de la radioactivité dans une installation nucléaire réelle (*au site de Birine par exemple*). Celui-ci aura la prétention d'être fortement blindés et **capables de résister (que ça soit du point de vue mécanique et/ou électrique) à des taux de radioactivité extrêmes.**

Et puisque le domaine de la robotique implique des nombreuses thématiques telles que la mécanique, l'électronique, l'automatique, l'informatique ou l'intelligence artificielle. Nous entamons ce domaine, i. e. celui des robots mobiles à roues, par un rapport à caractère introductif là où on va présenter quelques caractéristiques techniques sur ce type de robots mobiles. C'est en quelque sorte un résumé restreint de la recherche bibliographique qu'on a déjà entamé sur ce sujet.

I.2.Définitions

Un robot mobile est une machine automatique capable de se déplacer dans un environnement donné. Il est généralement constitué d'un châssis et d'un ensemble des roues, des chenilles, des pattes ou encore les ailes pour les robots areine ou marine en plus de l'électronique de commande et les moteurs associés **(Figure I.1).** Dans ce rapport on s'intéresse plutôt aux robots terrestres à roue.

Figure I.1 : Les différents types des robots mobiles.

I.3.Domaine d'application

Aujourd'hui, le domaine d'application des robots mobiles est très vastes, quelques exemples peuvent être résumés dans le tableau suivant : (Figure. I.2)

Domaines	Applications
Industrie nucléaire	- surveillance de sites - manipulation de matériaux radioactifs - démantèlement de centrales
Sécurité civile	- neutralisation d'activité terroriste - déminage - pose d'explosif - surveillance de munitions

Figure. I.2 : Domaine d'applications des robots mobiles.

I.4.Classification de robots à roues

Les robots mobiles à roues peuvent être classés selon leur degré d'autonomie, système de locomotion, leur domaine d'application, leur système de localisation, l'énergie utilisée…, etc. Pour nous, on s'intéressera plutôt à la classification des robots à roues par la position et le nombre des roues qu'ils contiennent [11][13].

I.4.1.Robot unicycle

Un robot de type unicycle est actionné par deux roues indépendantes, il possède éventuellement une ou plusieurs roues folles pour assurer sa stabilité. Son centre de rotation est situé sur l'axe reliant les deux roues motrices. (Figure I.3). C'est ce type de robot qui nous intéresse le plus, parce qu'il est similaire au robot qu'on possède dans le laboratoire en l'occurrence DANI II construit par National Instruments.

On note aussi que c'est l'une des configurations les plus utilisées en raison de la simplicité de commande, de construction et propriétés cinématiques intéressantes. C'est un robot non-holonome. On signale également qu'avec un unicycle à plateforme non-holonome, et qui de ce fait ne peut pas effectuer certains mouvements, on est obligé de faire plusieurs manœuvres pour suivre une certaine trajectoire [4][14].

Figure I.3 : robot mobile unicycle « Pioneer P3-DX ».

I.4.2.Robot tricycle

Un robot de type tricycle est constitué de deux roues fixes placées sur le même axe et d'une roue centrée orientable placée sur l'axe longitudinal. Le mouvement du robot est donné par la vitesse des deux roues fixes et par l'orientation de la roue orientable. Son centre de rotation est situé à l'intersection de l'axe contenant les roues fixes et de l'axe de la roue orientable. C'est aussi un robot non-holonome. En effet, il est impossible de le déplacer dans une direction perpendiculaire aux roues fixes. Sa commande est plus compliquée. Il est en général impossible d'effectuer des rotations simples à cause d'un rayon de braquage limité par la roue orientable. (Figure I.4) [13][14].

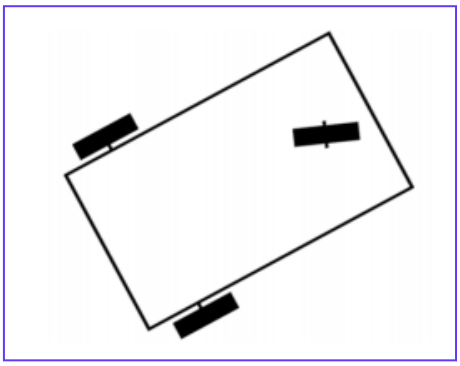


Figure I.4 : robot mobile tricycle.

I.4.3.Robot voiture

Un robot de type voiture constitué de quatre roues, deux roues en arrière sur le même axe pour la stabilité de robot et deux roues en avant sur le même axe orientable. **(Figure I.5).** Ce type de robot peut être considéré comme un unicycle et à plateforme non-holonome. Il est difficile à contrôler car il ne peut pas tourner sur place surtout dans des environnements encombrés.

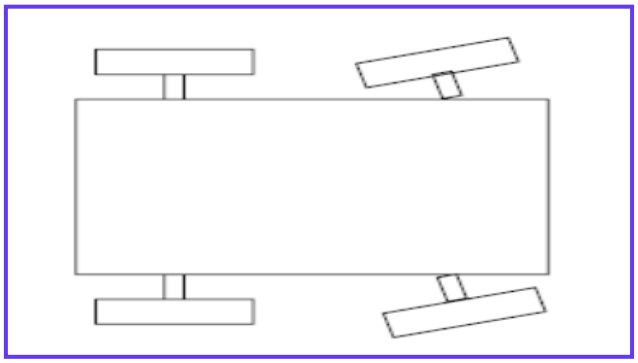


Figure I.5 : robot mobile de type voiture.

I.4.4.Robot omnidirectionnel

Le robot omnidirectionnel permet un déplacement libre dans toutes les directions en rotation et en translation. Ce type de robot est donc considéré comme une plateforme holonome. On distingue deux types des robots omnidirectionnels : le premier possède trois ou quatre roues **(*Figure I.6*)** qui tourne à la même vitesse pour fournir une translation à la direction souhaitée. Ce type de commande qui est d'ailleurs très simple est effectué uniquement dans le cas des translations.

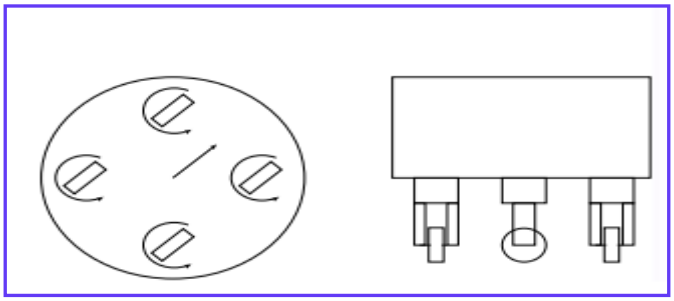


Figure I.6 : robot omnidirectionnel quatre ou trois roues orientables.

Le deuxième type est constitué de trois roues fixes de différente orientation et qu'on appel parfois roues suédoises ou roues holonomes **(Figure I.7).** Ce type de robot est commandé en

agissant sur la vitesse de rotation des trois roues. Si on veut par exemple aller tout droite, on fixe une roue et on fait tourner les deux autres roues pour aller dans la direction de la roue fixe ou dans le sens opposé. Si on veut le faire tourner, on actionne les trois roues en même temps et de même vitesse.

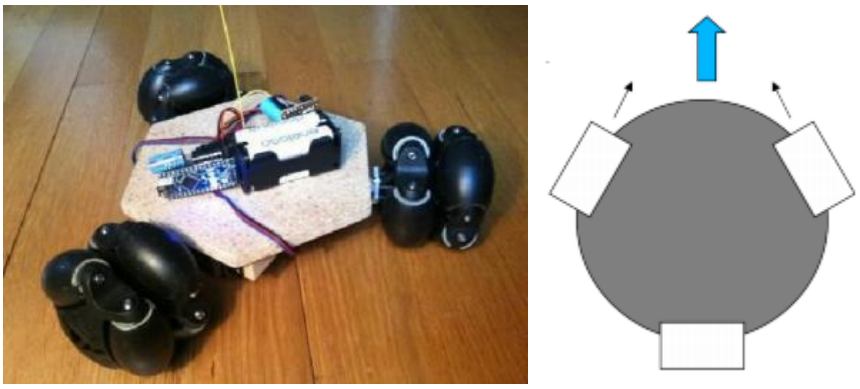


Figure I.7 : robot omnidirectionnel à roue suédoises.

I.5.Localisation

La localisation dans le domaine de la robotique mobile a une importance particulière, car un robot mobile, qui évolue dans un environnement donné, doit parfaitement connaître sa position ainsi que la position des autres objets présents dans le même environnement. L'opération de localisation peut être classée en deux classes différentes :

-***La localisation relative*** : consiste à intégrer des informations sur les vitesses ou les accélérations fournies par des capteurs proprioceptifs (odomètres, centrales inertielles, …etc.).

-***La localisation absolue*** : Cette méthode utilise des éléments repérables par le robot dans l'environnement de navigation, de position Connue, pour permettre au robot de se repérer relativement à ceux-ci. (Les balises, GPS, …etc.).

Pour l'implémentation de l'opération de la localisation, différentes méthodes sont utilisées, parmi elles on peut citer :

I.5.1.Odométrie

Le principe de l'odométrie est de déterminer la position du robot dans son environnement considéré. Pour cela, la méthode nécessite une initialisation dans son environnement. On utilise des capteurs proprioceptifs qui mesurent l'état de robot lui-même. Ces mesures sont

utilisées pour calculer la position(X, Y) et l'orientation (θ) de robot et d'où son positionnement par rapport au repère initiale.

Un certain nombre d'erreurs viennent entacher la précision de l'odométrie :

–Des erreurs systématiques : erreur sur le diamètre des roues par rapport à la valeur nominale attendue, diamètres différents, erreurs sur la disposition des roues, résolution des codeurs ;

– Des erreurs non systématiques : Dans le cas d'un sol non plan ou irrégulier, glissements divers (dus à la nature du sol, à une accélération trop brutale, à un obstacle, un défaut mécanique, etc.) contact au sol non ponctuel [1][6][14].

I.5.2.Système de balise

Ce système utilise deux méthodes pour localiser le robot: télémétrique et trigonométrique, la première méthode consiste à utiliser deux balises par contre la deuxième de trois balise. Les balise réfléchissant un signal émis (laser ou infrarouge) par un appareil de mesure de robot est pour cela on peut estimer l'angle et la distance robot-balise. Si l'on dispose de l'angle de gisement absolu de deux balises ou des angles de gisement relatifs de trois balises, on peut déterminer la position du robot. Il est toutefois plus robuste d'utiliser trois balises, la position de l'intersection des rayons réfléchis étant très sensible a une faible erreur angulaire si les angles de gisement sont faibles.

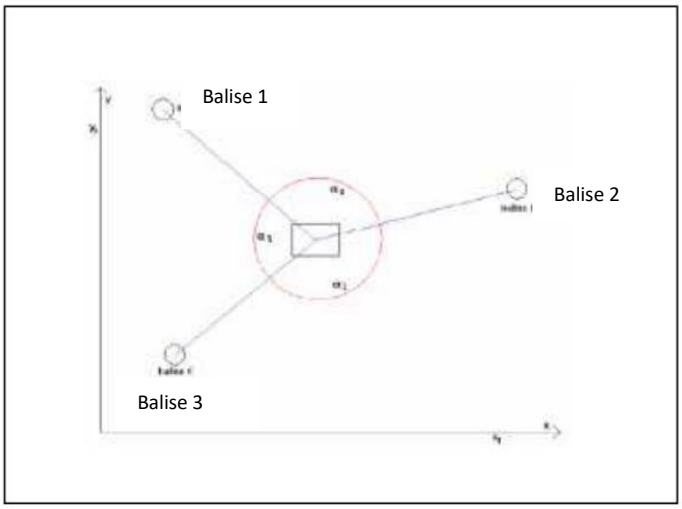


Figure I.8 : localisation d'un robot par la méthode trigonométrie

I.5.3 Global positioning system (GPS)

Global positioning system en français système de localisation mondial : un système de géolocalisation fonctionnant au niveau mondial.

Le système GPS, développé par l'armé américain dans les années 80, est utilisé pour obtenir des positions avec une précision de l'ordre du mètre à cause des capteurs GPS qui ne disposent pas d'un code qui correspondant à la précision maximale. A noter que l'utilisation de ces capteurs nécessite une puissance de calcul et une horloge de grande précision surtout si on utilise des capteurs à utilisation civile (qui ne disposent pas d'accès à la précision maximale des satellites). Une erreur d'un millionième de seconde dans la synchronisation entre l'horloge du capteur et celle du satellite provoque une erreur de 300 mètres sur la position. Pour améliorer cette précision, nous pouvons utiliser les GPS différentiels avec un second récepteur GPS sur une base fixe et de position connue. Il devient possible de mesurer l'erreur et d'en déduire la correction à apporter pour la zone environnante. Pour que ce système fonctionne, il faut que la base mobile reste à une certaine portée de la base fixe. Cette distance varie suivant la gamme de fréquence utilisée pour l'envoi de corrections, et peut atteindre quelques dizaines de kilomètres pour les besoins de la navigation maritime.[3]

I.6.Planification de trajectoire

La planification de trajectoire est un problème clé en robotique. Il a été très étudié ces dernières années et permet au robot de se déplacer pour rejoindre un but sans perdre son chemin. Pour atteindre ce but, on trouve différente classification que nous allons présenter [2].

I.6.1.Approche d'un but

Cette capacité de base permet de se diriger vers un objet visible depuis la position courante du robot. Elle est en général réalisée par une remontée de gradient basée sur la perception de l'objet. Ceci est similaire à l'exemple célèbre des véhicules de «*Valentino -Braitenberg*» qui utilisent deux capteurs de lumière pour atteindre ou fuir une source lumineuse. Cette stratégie utilise des actions réflexes, dans lesquelles chaque perception est directement associée à une action. C'est une stratégie locale, c'est-à-dire fonctionnelle uniquement dans la zone de l'environnement pour laquelle le but est visible.

I.6.2.Guidage

Cette capacité permet d'atteindre un but, qui n'est pas un objet matériel directement visible, mais un point de l'espace caractérisé par la configuration spatiale d'un ensemble d'objets remarquables, qui l'entourent ou qui en sont plus proches (voisins). La stratégie de navigation, utilisant souvent une descente de gradient également, consiste alors à se diriger dans la direction qui permet de reproduire cette configuration. Cette capacité utilisée par certains insectes, comme les abeilles, a été appliquée sur divers robots. Elle utilise également des actions réflexes et réalise une navigation locale qui requiert que les amers caractérisant le but soient visibles.

I.6.3.Action associe à un lieu

Action associe à un lieu c'est la première capacité réalisant une navigation globale (le robot connu entièrement l'environnement) c'est-à-dire qu'il peut rejoindre un lieu invisible depuis des repères mémorise de son environnement comme des zones de l'espace. Chaque zone associe une action, l'enchainement de ces actions définit une route qui permet de rejoindre le but. Comme le montré le figure I.9. L'objectif de robot est de rejoindre le lieu A qui est invisible par rapport au lieu D (le départ de robot). Pour cela le robot va utiliser les repères qui sont déjà mémorisés dans sa mémoire, le chemine à suive est représenté par les flèches [10].

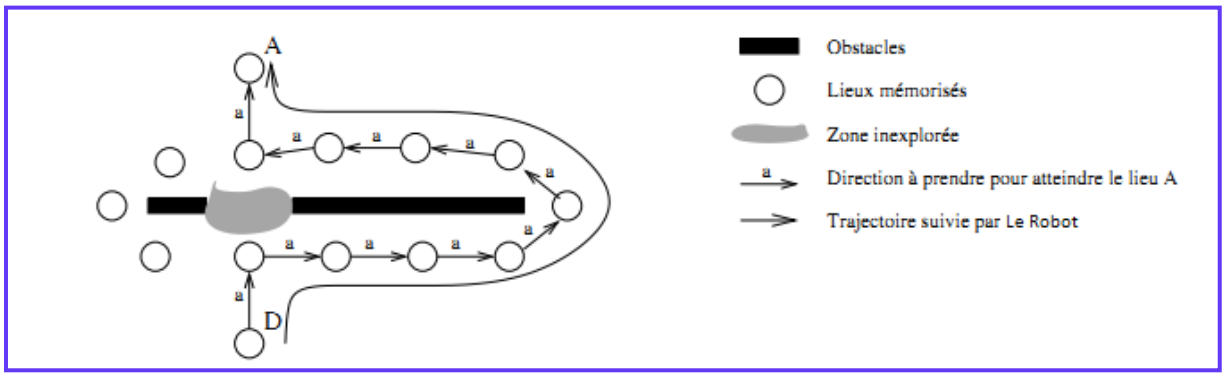


Figure I.9 : Action associée au lieu.

I.6.4.Navigation topologie

La navigation topologie est une capacité plus extensive qu'une action associée au lieu, elle mémorise dans son modèle interne les différents lieux dans sans environnement et les relations entre aux. Pour atteindre son objectif, il calcule les différents chemins et choisie le

chemin le plus court poursuivi comme le montre l'exemple dans la (**Figure I.10**.) On remarque qu'il y'a deux chemins à suivre pour rejoindre le lieu A à partir de la position actuel D. Le chemine le plus court est celui de la gauche, et c'est ce que doit suivre le robot.

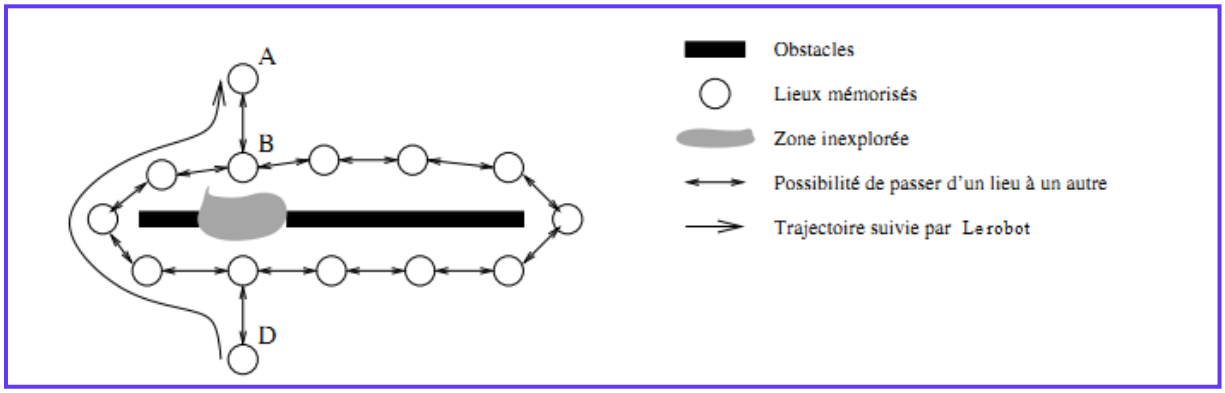


Figure I.10 : Exemple d'un navigateur topologie.

I.6.5.Navigation métrique

La navigation métrique c'est l'extensive de la topologie, car elle permet au robot de planifier son chemine au sein des zones inexplorées de son environnement, pour cela elle mémorise les positions métriques des différents lieux et la possibilité de passe de l'un à l'autre. Par composition de vecteur on peut calculer le chemine à suivre pour atteindre notre but, comme le montre **la Figure I.11**. Par la navigation métrique le robot choisie un chemine inexploré et encore non pas un lieu qui a été déjà mémorisé dans son modèle interne.

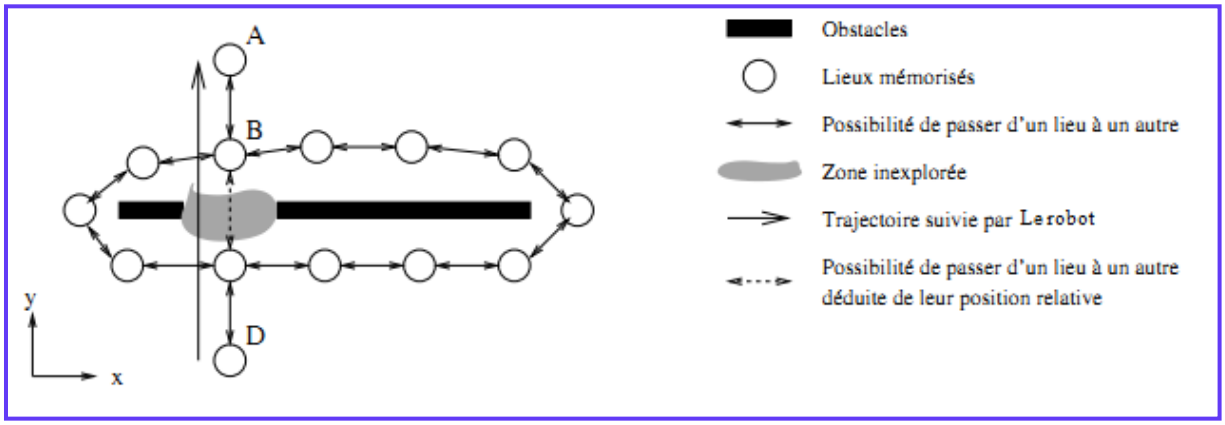


Figure I.11 : Navigation métrique.

I.7 Evitement d'obstacle

L'évitement d'obstacle est un sujet très important pour tous les robots mobiles. Il est indispensable pour permettre au robot de fonctionner dans un environnement dynamique et

pour gérer les écarts entre le modèle interne et le monde réel. Dans ce qui suit, nous allons décrire les méthodes les plus usitées pour implémenter cette caractéristique

I.7.1.Méthode du « The Vector Field Histogram »

Cette méthode consiste dans un premier temps à représenter l'environnement par une grille d'occupation centrée sur le robot, où chaque cellule contient une valeur entière correspondant à la probabilité d'y trouver un obstacle (Figure I.12.). Ensuite, l'environnement est discrétisé en secteurs angulaires pour lesquels la somme des valeurs des cellules est calculée. Un seuil permettant de tolérer un certain bruit est ensuite utilisé pour déterminer les directions possibles pour le robot [14].

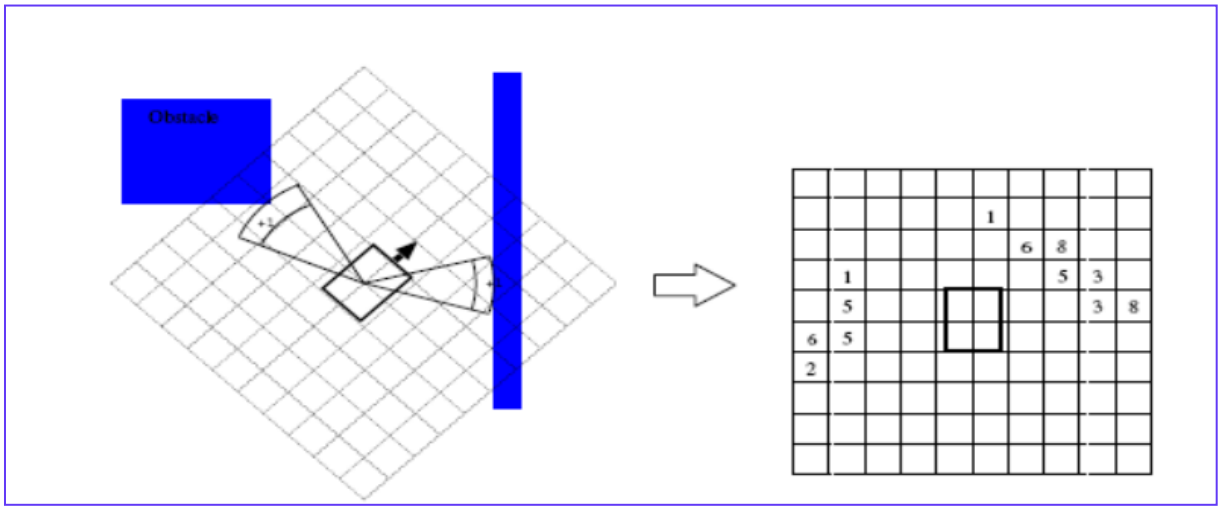


Figure I.12 : Représentation d'un environnement sous forme de grille d'occupation centre sur le robot.

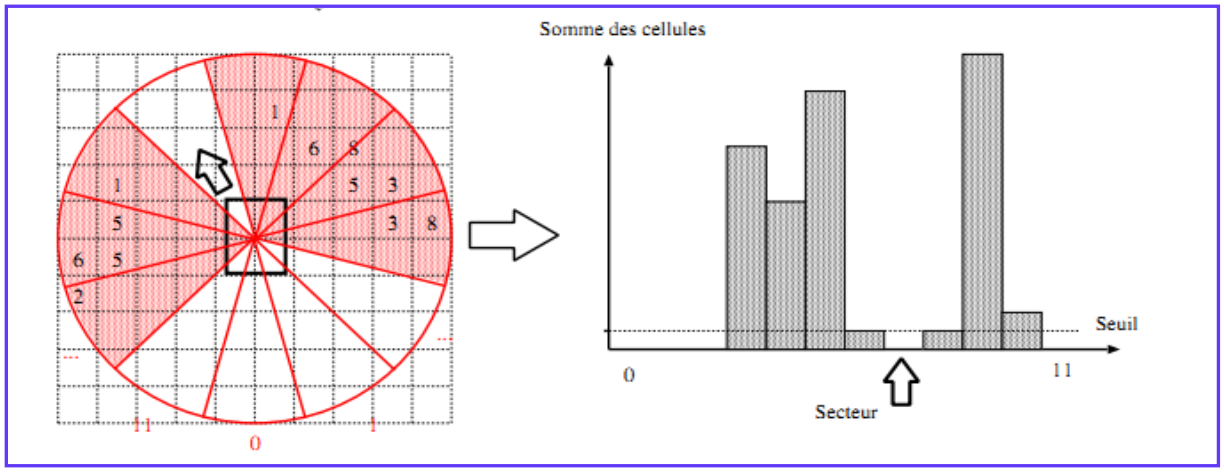


Figure I.13 : Décomposition de la grille sous forme des secteurs.

I.7.2.Méthode de la fenêtre dynamique

La méthode de la fenêtre dynamique permet, à partir de la perception locale de l'environnement, de sélectionner un couple (v , w) de vitesses de translation et de rotation du robot qui répond à différentes contraintes, dont celle d'éviter les obstacles. Un tel couple de vitesses, lorsqu'il est appliqué au robot, produit une trajectoire circulaire, pour laquelle la satisfaction des différentes contraintes peut être évaluée. Pour atteindre ce but, la méthode de la fenêtre dynamique est évaluée pour chacune des trajectoires possibles à partir de la perception locale de l'environnement à un instant donné et de la position estimée du robot à un pas de temps fixé dans le futur pour la trajectoire courante. Si le robot n'a pas rencontré d'obstacles à cet horizon, la contrainte est respectée dans le cas contraire elle ne l'est pas (Figure I.14). Le respect ou le non-respect de cette contrainte est reporté dans un graphe des vitesses qui indique, pour chaque couple de vitesses possible (donc chaque trajectoire), si le robot va ou ne va pas rencontrer un obstacle (Figure I.15). Au sein de cette fenêtre, un couple de vitesses choisir qui ne conduise pas à percuter un obstacle pour garantir un déplacement sûr du robot.

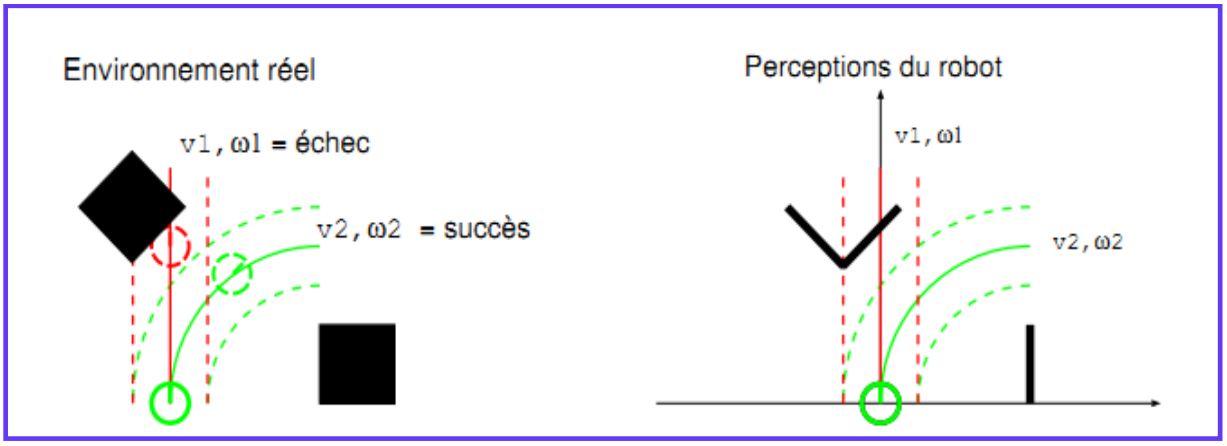


Figure I.14 : Contrainte d'évitement d'obstacle pour la méthode de la fenêtre dynamique.

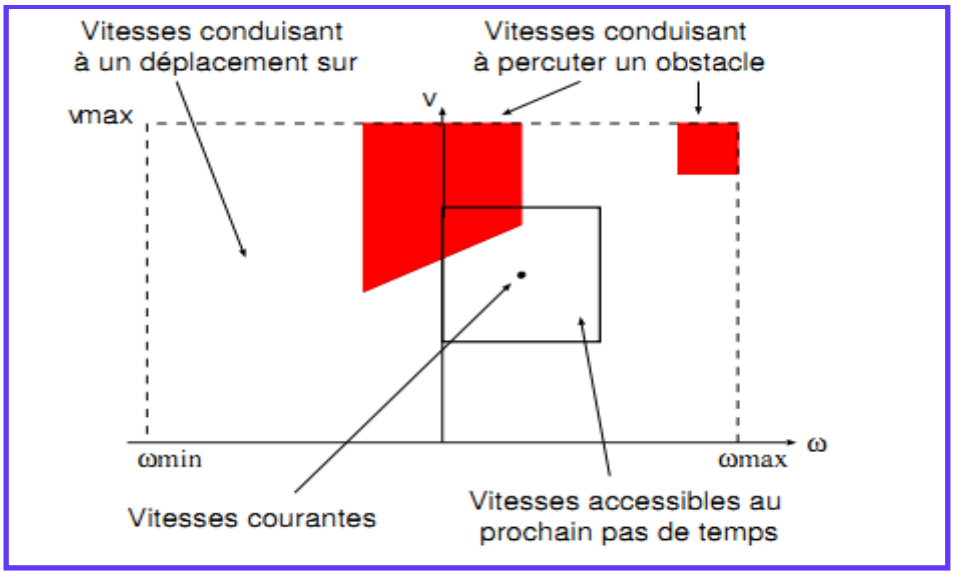


Figure I.15 : Fenêtre de choix d'une vitesse sans collusion.

I.8.Conclusion

Ce rapport introductif est un résumé succinct de la bibliographie qu'on vient de consulter sur le sujet des robots mobiles et plus particulièrement sur les robots type unicycle. Nous avons surtout abordé les notions de base liées au jargon de cette discipline qui invoque à son tour d'autres sous-disciplines tels : La cinématique, La mécanique, l'instrumentation, etc. Le baguage ainsi acquis, nous permettra pour un premier temps de manipuler le Robot DANI-II. C'est le type de robot dont est muni notre laboratoire et qu'on entend intégrer dans nos activités futures telles que l'utilisation des robots pour la mesure des radiations nucléaires dans les zones suspectes et dangereuse, la prospection et prise de photos dans endroits à accès difficiles, …etc.

Chapitre II

Description du robot DaNi 2.0

II.1.Introduction

Dans ce chapitre on va présenter la partie matérielle (Hardware) sur laquelle repose notre travail. En effet, le robot Dani 2.0 développé par la société Américaine : National Instruments Corporation, communément appelée NI. Celle-ci est une entreprise américaine dont le siège social est basé à Austin, Texas. Et c'est un fournisseur de matériels et de logiciels permettant aux scientifiques et aux ingénieurs de concevoir, prototyper et déployer des systèmes aux applications de test et mesure, de contrôle/commande et embarqués.

II.2.Description du Robot DaNi 2.0

Le robot ***DaNi 2.0*** est un robot de type unicycle qui contient deux roues motorisées sur un même axe et une roue holonome, à galets tangentiels, additionnelle utilisée pour assurer la stabilité (**Figure II.1).** Il est essentiellement constitué par :

- Une *carte* d'acquisition et de contrôle embarqué *sbRIO*-9632.

-Un capteur ultrasonore muni d'un servomoteur.

-Deux interrupteurs, le premier s'appelle Master et celui qui contrôle l'alimentation du robot et le deuxième s'appelle Motor, il commande l'alimentation de deux moteurs.

- Deux moteurs à courant continue. (**Figure II.2)**.

Dans notre application, notre robot est relié à un ordinateur, dans lequel on a installé le logiciel LabView, via une connexion par un réseau wifi.

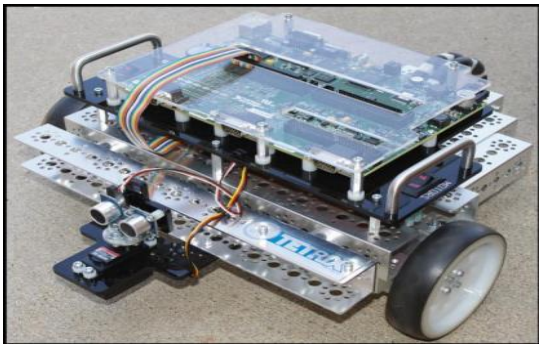


Figure II.1 : Robot DaNi 2.0

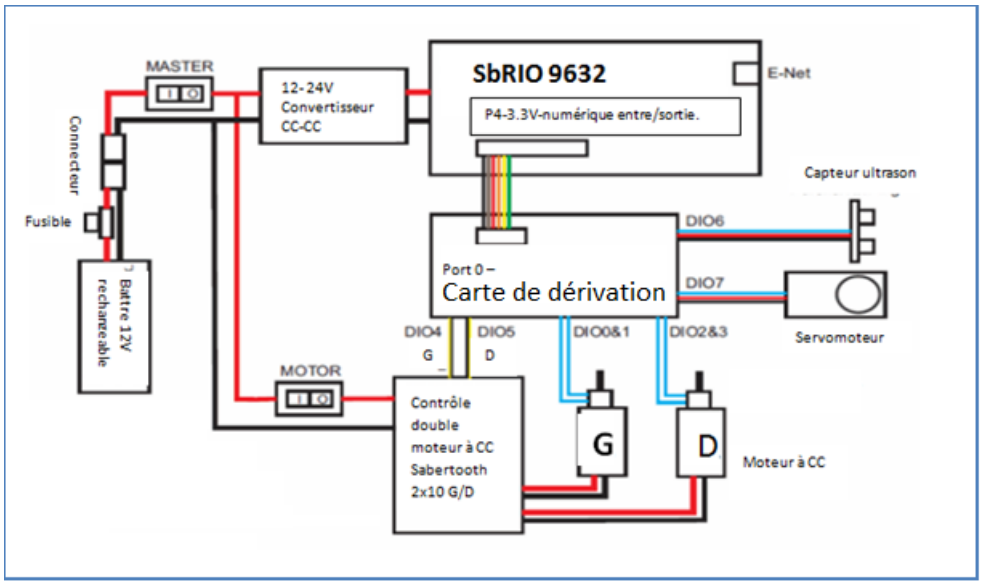


Figure II.2 : Schéma de robot DaNi 2.0

II.3.La carte SBRIO-9632

La carte « Single-Board RIO-9632 » voir **(Figure II.3)**est une carte électronique programmable qu'on aura à utiliser pour la commande et le contrôle des différents éléments du robot mobil, telle que la vitesse de rotation des deux moteurs, l'angle de rotation du servomoteur et l'acquisition des données des capteurs…etc.

On tient également à préciser que la carte contient, entre autres :

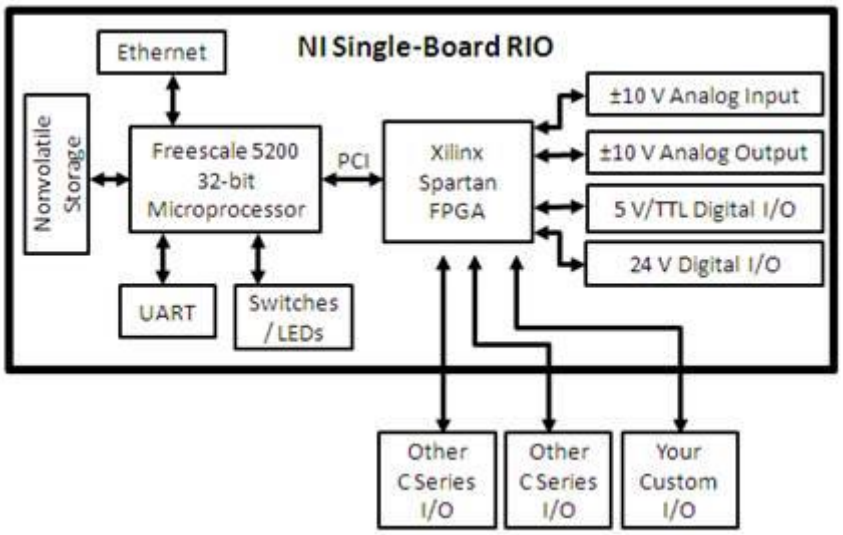


Figure II.3 : La carte SbRIO-9632.

II.3.1.Un circuit programmable de type FPGA

Sur la carte SbRIO, les connexions entrées/sorties avec les capteurs (capteur à ultrason, capteur de mesure de vitesse ou de position, ...) et les actionneurs (moteurs, servomoteur…) transitent tous par un circuit programmable FPGA de type Xilinx Spartan-3 FPGA. Celui-ci commande les actionneurs et récupère les informations issues des capteurs en relation.

En fait, un FPGA, qui un acronyme de Field-Programmable-Gate-Array ou en français un réseau des portes logique programmables. C'est un circuit intégré numérique composé d'un réseau des cellules programmables **(Figure II.4),** CLB (Configurable Logic Block). Chaque cellule est capable de réaliser une fonction choisie parmi plusieurs possibles.

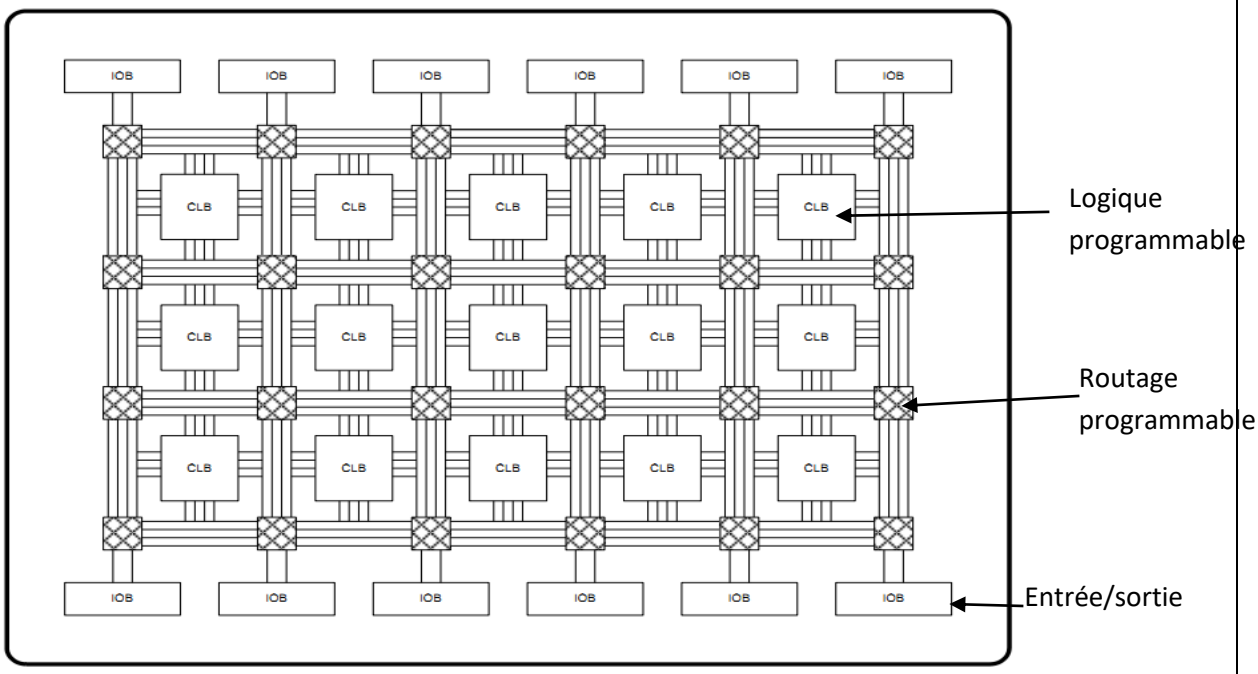


Figure II.4 : Architecture interne d'un circuit FPGA type.

On note aussi que les CLB basés sur les tables de conversion utilisent des petites mémoires programmables. Elle se compose de quatre tables de conversion (LUT - Look Up Table) **(Figure II.5)** et chaque LUT contient une RAM. On signale aussi la présence d'une bascule Flip-Flop et un multiplexeur logique contrôlée par bascule Latch.

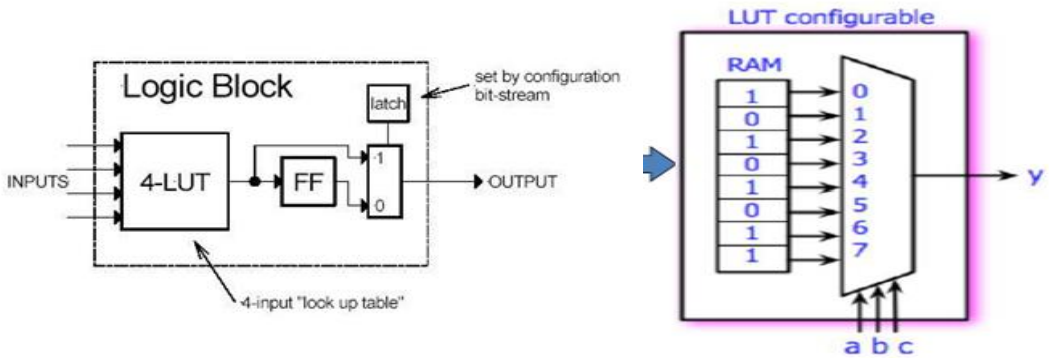


Figure II.5 : L'architecteur de CLB et la LUT.

Les interconnexions sont également programmables. La reconfiguration dynamique du circuit FPGA consiste à changer la programmation de ces circuits logiques programmables alors qu'ils sont en activité.

II.3.2.Processor Real-time (*Industrial Freescale MPC5200 real-time processor*):

Le processeur temps réel qui équipe la carte NI SbRIO-9632 assure la connexion robot-ordinateur et le contrôle du circuit FPGA.

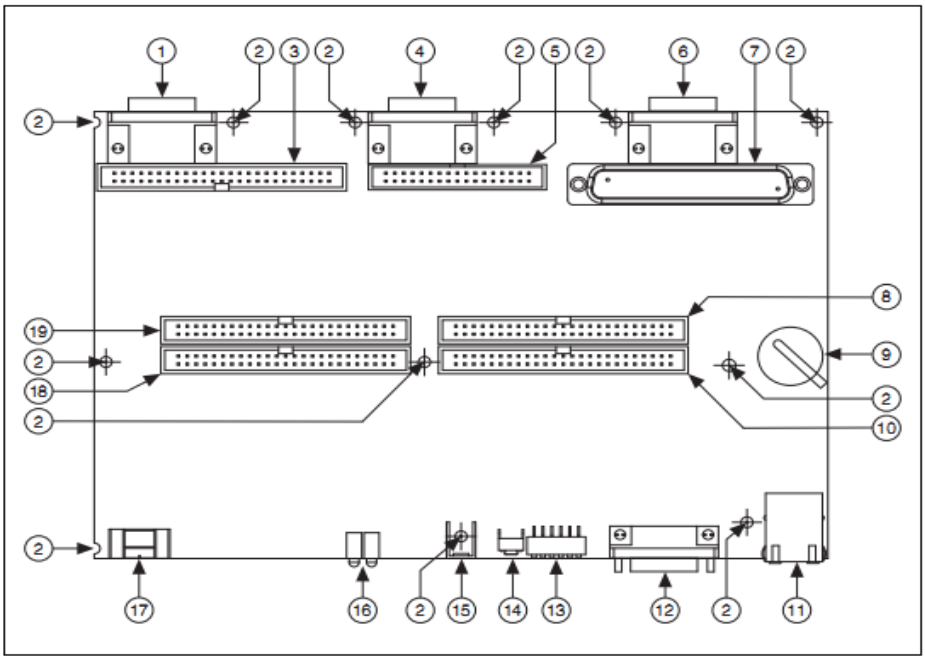
La plupart des applications LabVIEW s'exécutent sur un système d'exploitation général (OS) comme Windows, Mac OS ou Linux. Mais, certaines applications nécessitent des performances en temps réel que les systèmes d'exploitation à usage général ne peuvent pas en garantir.

Les systèmes en temps réel sont déterministes. Le déterminisme est la caractéristique d'un système qui décrit à quelle fréquence le système répond à l'extérieur. Des événements donnés sont traités via des opérations dans un délai bien déterminé. La plupart des applications en temps réel exigent un comportement temporel strict et s'exécute régulièrement dans une petite fenêtre de temps acceptable.

II.3.3.Connecteurs et autre composant

Sur la carte, on trouve également quatre connecteurs entrées/sorties numérique (Digital Input/output) P2, P3, P4, P5. Ceux-ci sont connectés directement au circuit FPGA. Chaque connecteur contient 50 pins qui sont à leur tour divisés en trois ports distincts. Notons ici que le connecteur P4 est utilisé par le robot DaNi.

En plus de ça, la carte renferme un connecteur analogique de 34 pins, quatre ports RS-232, un indicateur à quatre LEDs utilisé pour définir l'état du système de contrôle du robot DaNi et un port Ethernet RJ-45 **(Figure II.6).**



1	J10, Connector for C Series Module 3	11	J2, RJ-45 Ethernet Port
2	Plated Mounting Holes	12	J1, RS-232 Serial Port
3	J7, Analog I/O Connector	13	DIP Switches
4	J9, Connector for C Series Module 2	14	Reset Button
5	J6, 24 V Digital Input (sbRIO-964x/9642XT Only)	15	P1, Ground Lug
6	J8, Connector for C Series Module 1	16	LEDs
7	J5, 24 V Digital Output (sbRIO-964x/9642XT Only)	17	J3, Power Connector
8	P4, 3.3 V Digital I/O	18	P3, 3.3 V Digital I/O
9	Backup Battery	19	P5, 3.3 V Digital I/O
10	P2, 3.3 V Digital I/O		

Figure II.6 : Les connecteurs de la carte Sb-RIO.

II.4.Commande des Moteurs Electriques

Le déplacement du robot DaNi est assuré par deux moteurs à courant continu physiquement reliés aux deux roues. La rotation de ces deux moteurs de la même vitesse et dans et dans des sens différents provoque un déplacement du robot soit en avant où en arrière. Alors que la différence de vitesse ou le sens de rotation établira la rotation de robot.

Le contrôle du sens et de la vitesse rotationnelle des deux moteurs se fait par la commande PWM (Pulse Width Modulation) et un régulateur PID. Une récupération des informations sur l'état des deux roues (Vitesse et position) se fait par le biais d'un capteur optique solidairement collé aux deux axes des moteurs.

II.4.1.Contrôleur (Driver) des moteurs à courant continu Sabertooth 210

Chacun des deux moteurs du robot a besoin une alimentation de 4.6A et que malheureusement la carte Sb-RIO ne peut pas délivrer par les deux files de commande DI04 (contrôle le moteur gauche) et DI05 (contrôle le moteur droite) **(Figure II.2).** Ces pins délivre uniquement 3mA chacune comme débit maximale. Pour résoudre ce problème on a utilisé

Contrôleur de moteur régénératif à double canal connu sous l'appellation de Sabertooth 2x10**(Figure II.7).** Celui-ci assemble les deux files de commande et la ligne de l'alimentation électrique (la batterie) en entrée pour fournir 10A à chaque moteur avec un contrôle de vitesse et sens de rotation.

Le module « Sabertooth 2x10 » est accessible via dix broches de connexion. Plus de détails sur ces broches et leurs modes de connexions sont indiqués sur le tableau II.1. Le module contient aussi un SWITH DIP qui permet de choisir l'un des quatre modes de fonctionnement sélectionnables grâce aux switches intégrés dans le boîtier du circuit intégré. Le mode utilisé dans le cas du robot DaNi 2.0 est le mode : Micocontroller Pulses, Independant linear control. Ce mode permet de contrôler les deux moteurs séparément et les arrêter si le signal de commande est perdu.

	Branche
Moteur 1	M1A, M1B
Moteur 2	M2A, M2B
Batterie	B+ avec le + de la batterie, B- avec le – de la batterie.
Commande PWM de premier moteur	Deux files d'alimentation vers + et – de la carte. La troisième file de commande vers S1.
Commande PWM de seconde moteur	Deux files d'alimentation vers + et – de la carte. La troisième file de commande vers S2.

Tableau II.1 : Les branches de Sabertooth.

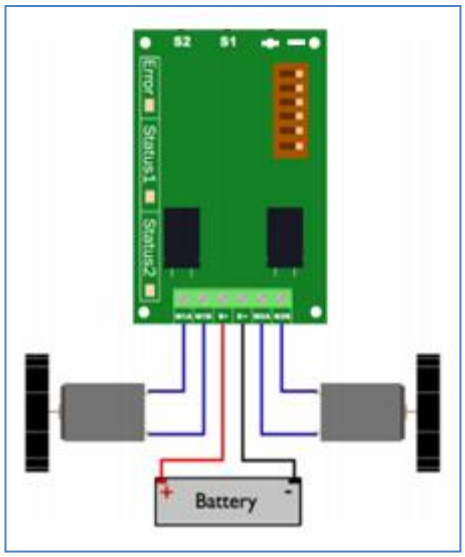


Figure II.7 : Sabertooth.

II.4.2.Sens de rotation des deux moteurs DC

Le robot DaNI est équipé de deux moteurs à courant continu qui peuvent aller soit vers l'avant ou vers l'arrière en fonction de leur polarité de la tension d'entrée. Pour modifier le sens de rotation d'un moteur à courant continue il suffit d'inverser l'alimentation à ses bornes, la structure permettant de réaliser cette inversion s'appelle pont « H » **Figure II.8.** Le module de puissance sabertooth 2x10 RC contient deux ponts en H à son étage de sortie. Chaque étage contient quatre transistors MOSFETs divisés diagonalement en deux couples de deux transistors, chaque couple est activé à l'inverse de l'autre à l'aide d'une porte logique non. Si on active le premier couple le courant allé droit vers la gauche du moteur qui tourne dans le sens positive. Si l'on inverse la commande, le courant allé de gauche vers droite du moteur qui va tourner dans le sens négatif.

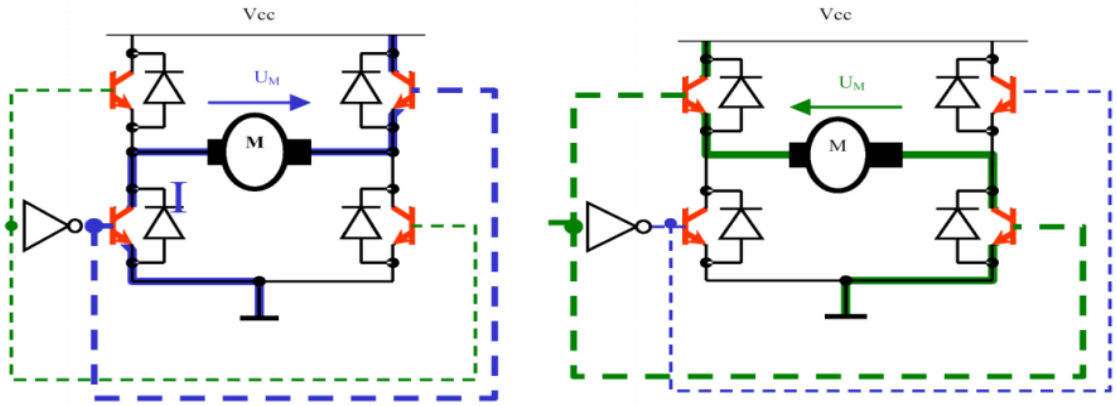


Figure II.8 : Pont H.

II.4.3.Vitesse de rotation des deux moteurs DC

Le signal de commande numérique pont en H cité en haut se présente souvent sous la forme d'une onde carré communément appelé signal PWM (*Pulse Width Modulation*) **(Figure II.9)**.Pour contrôler la vitesse de deux moteurs on varie le rapport cyclique de ce signal d'où la valeur moyenne de tension résultante qui alimentera l'un ou l'autre des deux moteurs.

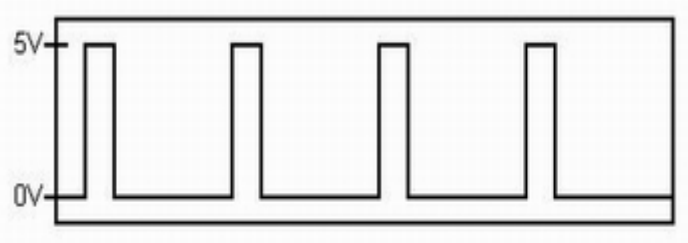


Figure II.9 : signale PMW.

Le signal PWM est utilisé pour générer les niveaux de tension souhaités pour les moteurs, mais le courant nécessaire pour conduire les moteurs doit provenir de la batterie. Le driver sabertooth 2x10 RC combine la puissance de la batterie avec les signaux de commande PWM pour produire des signaux analogiques avec la tension désirée et un courant suffisant pour alimenter les moteurs **(Figure II.10)**. [15]

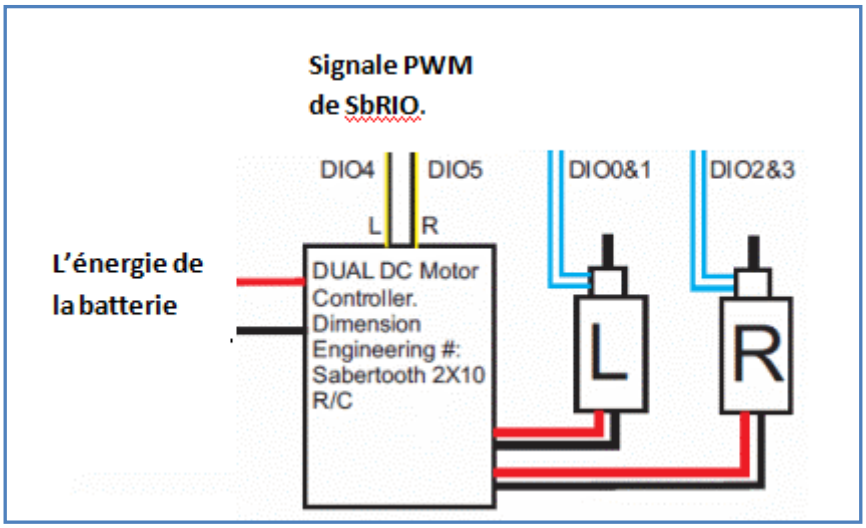


Figure II.10 : Circuit de contrôle les deux moteurs.

II.4.4.Capteur de vitesse/position à codeurs optiques

Les capteurs de vitesse appelés codeurs optiques sont connectés à la carte de contrôle via les fils bleus s'étendant sur DIO 0 et 1 et DIO 2&3 dans la Figure II-2.

Un codeur est un dispositif électromécanique qui convertit le déplacement linéaire ou rotatif en signaux numériques ou impulsionnels. Un codeur optique utilise un disque rotatif, une source lumineuse et un photodétecteur (capteur de lumière).

Le disque est monté sur l'arbre rotatif du moteur et présente des motifs de secteurs opaques et transparents. La source de lumière du codeur, dirigée vers un photodétecteur, traverse les secteurs du disque. À mesure que le disque tourne, ces motifs interrompent la lumière émise sur le photodétecteur, générant une sortie de signal numérique ou impulsionnel. **(Figure II.11)**. Le type d'encodeur incrémental le plus courant utilise deux canaux de sortie (A et B) et deux pistes de code avec des secteurs positionnés à 90 ° hors phase, comme le montre la Figure II-12. Les deux canaux de sortie indiquent la position et le sens de rotation. Si A mène B, par exemple, le disque tourne dans le sens des aiguilles d'une montre. Si B conduit A, alors le disque tourne dans le sens inverse des aiguilles d'une montre. Par conséquent, en surveillant à la fois le nombre d'impulsions et la phase relative des signaux A et B, vous pouvez suivre à la fois la position et le sens de rotation. Certains détecteurs en quadrature comprennent un troisième canal de sortie, appelé zéro ou signal de référence, qui fournit une seule impulsion par tour qui peut être utilisée pour la détermination précise d'une position de référence.

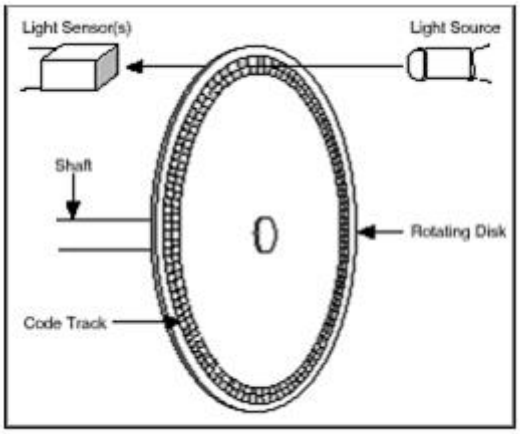


Figure II.11 : La structeur de capteur optique.

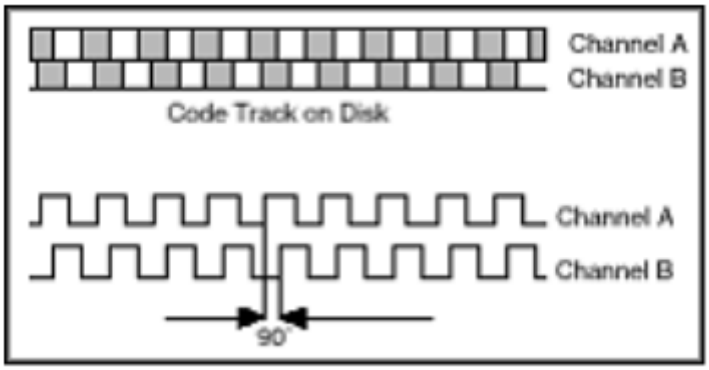


Figure II.12 : Les deux encodeurs A et B.

Pour déterminer la vitesse de la roue à partie de l'encodeur on compter le nombre des impulsions dans un intervalle de temp bien déterminer et par application de la loi suivante :

$$\text{Vitesse (rotation par minute)} = \frac{\frac{nombredespulsion\ dansunintervalledetemp * 60\ s}{nombretotaledespulsions * 1\ min}}{intervaldetemp\ (\text{S})}$$

II.4.5.Le détecteur d'obstacle à ultrason

Le capteur ultrasonore (détecteur d'obstacle) utilisé dans le Robot DaNI est monté sur l'axe d'un servomoteur. Ce dernier confère au détecteur l'angle d'orientation voulue. Le détecteur est lié avec la carte SbRIO par un connecteur nommée DIO6 **Figure II.2.** Comme principe de fonctionnement, celui-ci utilise des vibrations sonores dont les fréquences ne sont pas perceptibles par l'oreille humaine. Les fréquences couramment utilisées dans ce type vont de 20 kHz à 200 kHz. Les ondes à ultrasons sont émises pendant 200 µs dans l'air à une vitesse de 331.5 m/s. Après l'émission, le détecteur guette l'écho pendant une durée de 18.5 ms. Les ultrasons sont réfléchis partiellement lorsqu'ils heurtent un corps solide. Distance maximale détectable est limitée à quelque mètres « 3m » et l'angle de détection maximale est de 22.5 deg **(Figure II.13)**. Le calcule distance entre le capteur et l'obstacle se fait à l'intérieur du circuit FPGA en utilisant la loi suivante : [11][10].

$$\text{Distance (m)} = \frac{331.5\left(\frac{m}{s}\right) * T\ (s)}{2}\ . \qquad \text{(II.1)}$$

T (s) : le temps de réponse.

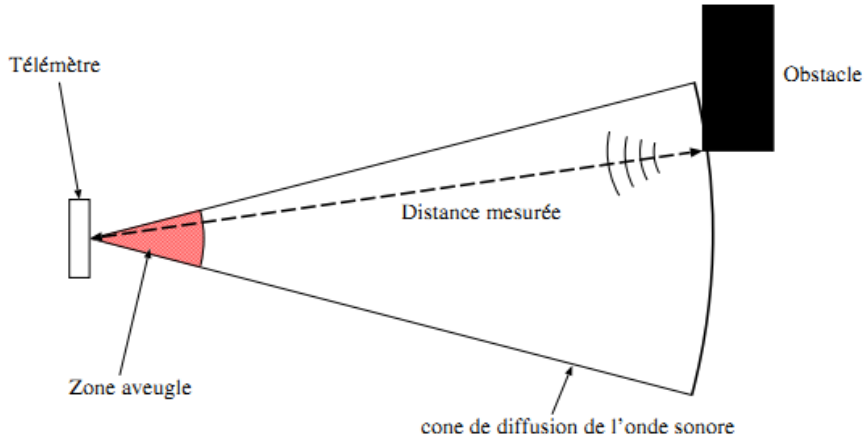


Figure II.13 : La zone détecte par ultrason.

Les inconvénients de ce type de détecteurs sont qu'ils possèdent une "zone aveugle", de quelques centimètres au-dessous de là ils ne peuvent pas détecter les obstacles. De plus, l'onde repêchée est très sensible aux conditions environnementales locales. Ainsi que, si l'angle entre l'obstacle et la direction de l'onde sonore est trop faible il n'y aura pas de retour de l'onde sonore et l'obstacle ne sera pas perçu. L'onde de retour dépend également de la texture de l'obstacle un mur couvert de moquette ne pourra pas être détecté. Ces détecteurs ne détectent pas également les fins d'obstacles comme les pieds de table ou de chaise.

II.4.6.Servomoteur

Le servomoteur utilisé pour faire orienter le détecteur ultrason joue le rôle d'un actionneur. Il est connecté à la carte de contrôle SbRIO(FPGA) à travers le connecteur DIO7 **Figure II.2** par trois fils, un fil rouge est relié à l'alimentation positive (+5V), un fil noir vers la masse (GND) et le fil jaune est utilisé pour la commande.

En fait, le pilotage d'un servomoteur ne demande pas d'interface de puissance est se fait par signal de commande PWM (Pulse Width Modulation), il suffit d'envoyer une impulsion de quelque milliseconde dans un intervalle de 20 ms, la largeur de ces impulsions déterminer l'angle de rotation comme montre le **Figure II.14**

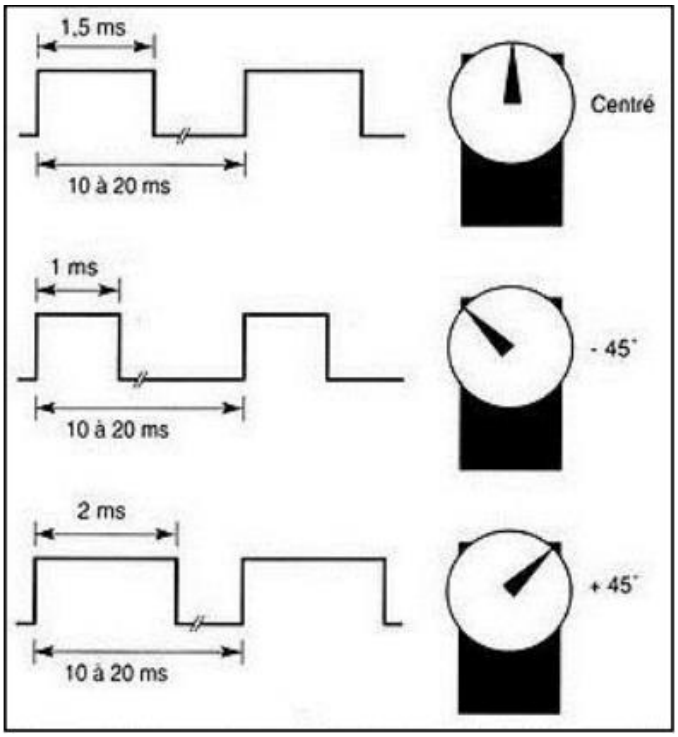


Figure II.14 : Les différents angles de rotation selon longueur d'impulsion.

Le servomoteur compose d'un dispositif électronique d'asservissement, un réducteur mécanique et un moteur à courant continue commandé via un circuit de puissance contenant un pont H à son étage de sortie.

II.5.Modélisation cinématique du robot

Pour pouvoir commander le robot avec une précision plus ou moins acceptable, on doit va faire une modélisation cinématique (l'hypothèse de roulement sans glissement) de notre robot sur le plan *x*, *y* et θ. Le modèle ainsi obtenu sera par la suite utilisé dans la partie commande du robot.

Les roues motrices ayant même axe de rotation qu'on appelle CIR (Centre Instantané de Rotation) **Figure II.15**. Soit « ***r*** » le diamètre de roue qui vaut pour nôtre cas : ***0.1 m***. Et « $\dot{\varphi}$ » la vitesse angulaire de roue (radian/s). On exprime la distance « ***d*** » parcourue par la roue qui par la relation suivante :

$$d = \pi\, r\, \dot{\varphi}\, t.\ (m). \qquad \text{(II.2)}$$

D'où la vitesse linéaire de robot « ***v*** » qui est donnée par :

Si les deux roues tournent de même sens et même vitesse le robot se déplace en ligne droit, on calcule leur vitesse par application de cette relation :

$v = d/t = \pi . r.\dot{\varphi}. t / t = \pi . r.\dot{\varphi}.$ (II.3)

Et si on veut exprimer la vitesse de rotation de robot :

Soit « ρ » le rayon de courbure de la trajectoire du robot, c'est-à-dire la distance du CIR au point « ó ». Soit « L »la distance entre le CIR et la position d'une roue. Et « w » la vitesse de rotation du robot autour du CIR. Alors les vitesses des roues droite et gauche, respectivement notév_d et v_g , vérifient les relations :

$v_d = -r.\dot{\varphi}_d = (\rho + L)\, w.$ (II.4)

$v_g = r.\dot{\varphi}_g = (\rho - L)\, w.$ (II.5)

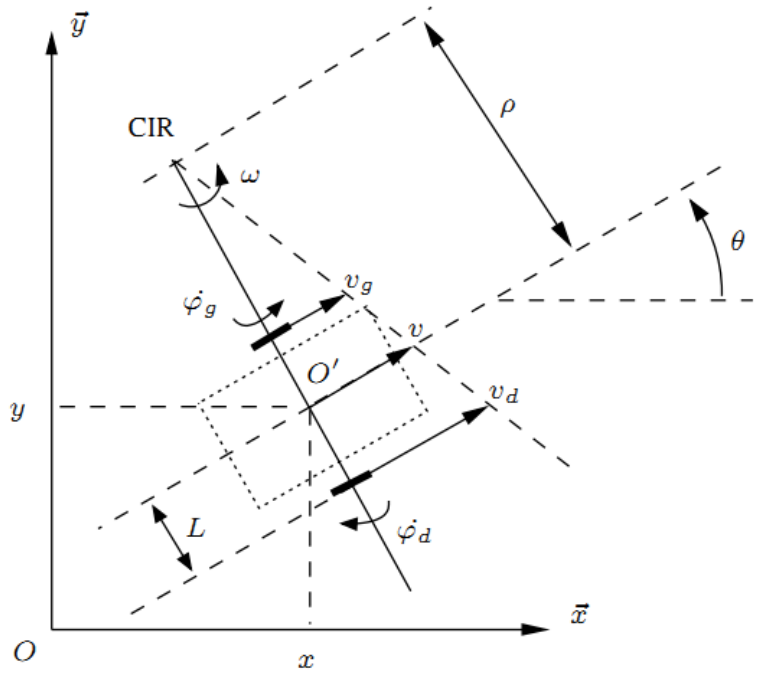


Figure II.15 : Modélisation de robot unicycle.

Ce qui permet déterminer la vitesse de rotation de robot : [1]

$$w = -\frac{r(\dot{\varphi}_d + \dot{\varphi}_g)}{2L}.$$ (II.6)

II.6.Conclusion

Au cours de ce chapitre, nous avons brièvement décrit les différentes parties, électroniques, électriques et électromécaniques qui composent le robot DaNi 2.0. Nous avons également donnée le modèle cinématique du robot, un modèle qu'on va utiliser lors de la commande effective du robot.

Chapitre III

Outil Logiciel de programmation : LabVIEW

III.1.Introduction

Au cours du chapitre précédent, nous sommes intéressés aux différentes parties constituantes du robot DaNi2.0. Le chapitre présent sera consacré à la description de l'outil logiciel utilisé lors de notre travail pour la programmation et le guidage du robot. Il s'agit bien du logiciel « LabVIEW »crée par la firme Américaine National Instrument.

III.2.Brève introduction au langage de programmation : LabVIEW

Le mot « LabVIEW » est un diminutif de **LAB**oratory **V**irtual **I**nstrumentation **E**ngineering **W**orkbench. C'est un logiciel de programmation graphique (haut niveau programmation graphique) **(figure III.1).** Il a été créé en 1986 par Dr.JAMES TRUCHARD, JEFFREY KODOSKY et WILLIAM NOWLIN pour les applications qui nécessitent un test, une mesure et un contrôle avec un accès rapide aux informations sur le matériel et les données. Il est actuellement utilisé dans plusieurs domaines à travers le monde et aide les ingénieurs et les scientifiques de prendre des mesures, des contrôles processus, la collecte et l'analyse des données. [10]

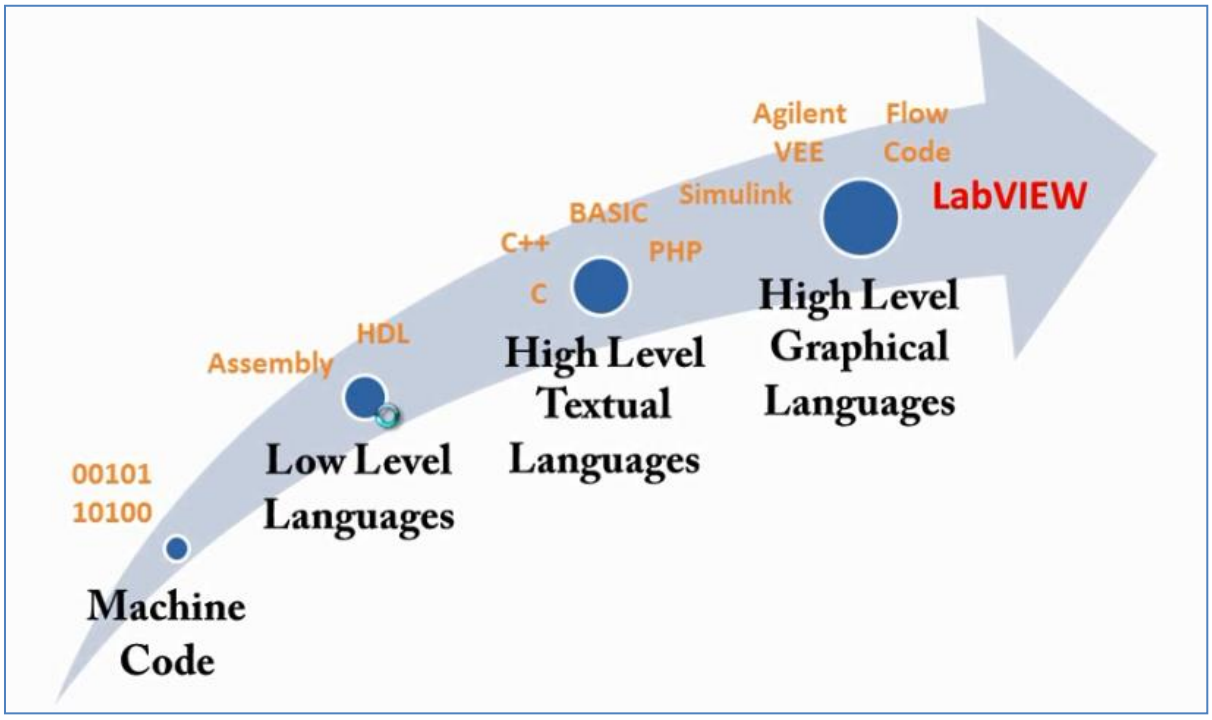


Figure III.1: Les différents niveaux de programmation.

Le diagramme de LabVIEW est lié à une interface utilisateur graphique nommée face-avant. Les programmes et sous-programmes sous forme d'icônes sont appelés des **I**nstruments

Virtuels (***VI***) et les fichiers source enregistrés sur disque ont l'extension de nom de fichier « *.VI »

Chaque ***VI*** possède trois composants : un diagramme qui intègre le code graphique, une face-avant personnalisable par l'utilisateur et un panneau de connexions pour les icônes qui sert d'entrées/sorties pour les variables sous forme de fils.

Une fois un VI écrit, il devient une icône pour un programme de plus haut niveau et intégré dans le nouveau diagramme. Il devient alors un sous-VI appelé donc un sous-programme. Chaque icône de sous-VI créé peut-être personnalisée par un dessin représentant au plus près sa fonction.

III.3.Lancement du programme

La fenêtre de démarrage **(Figure III.2)** s'ouvre lorsque on lance Labview, sur cette fenêtre on peut créer un nouveau projet, créer un fichier ou ouvrir des fichiers existants. La fenêtre de démarrage disparaît lorsqu'on ouvre un fichier existant ou qu'on crée un nouveau fichier, et réapparaît lorsque on ferme toutes les faces-avant et tous les diagrammes ouverts.

III.4.Ouverture d'un nouveau VI

Après lancement Labview et louverteur la fenetre de démarage, on click sur « *Create project* », puis on sélectionne« *Blank VI* » dans la nouvelle fenetre.

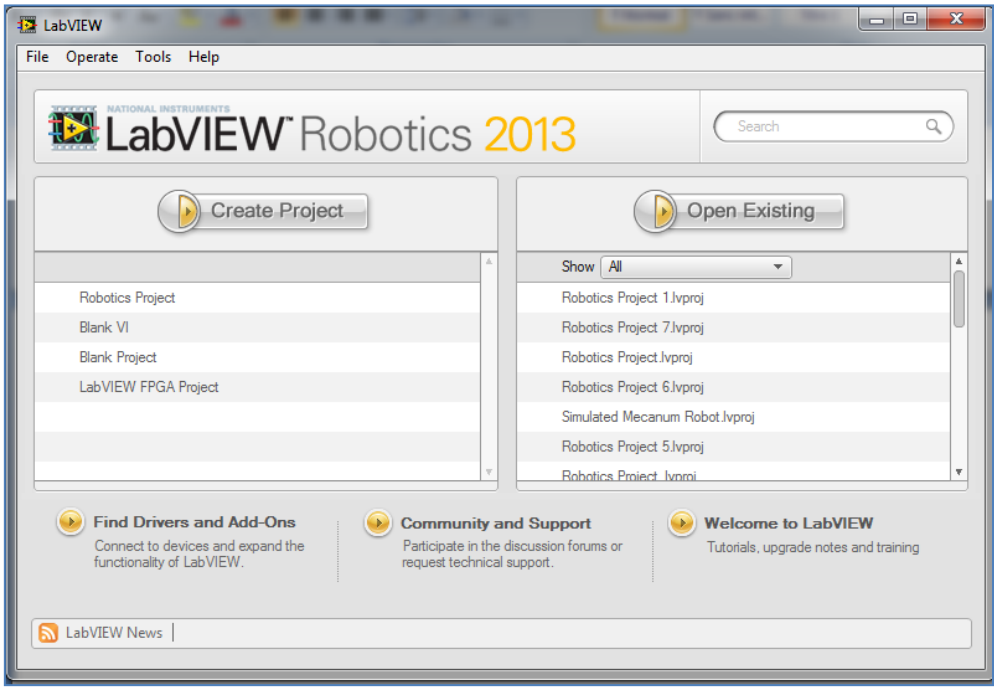


Figure III.2 : Fenêtre de démarrage.

Deux fenêtres sont par la suite automatiquement ouvertes : « Front Panel » et « Block Diagram ».Les deux fenêtres sont systématiquement reliées entre elles. **(Figure III.3)**.

-La fenêtre « Front Panel » contient les entrées du programme comme les buttons ou les contrôles numériques, ...etc. Elle contient également les champs de sorties du notre programme, par exemple : les afficheurs numériques ou les graphes et même des photos ou de gestionnaires de son…etc.

- La fenêtre « Block Diagram », sur cette fenêtre on trouve le code source du programme sous une forme graphique.

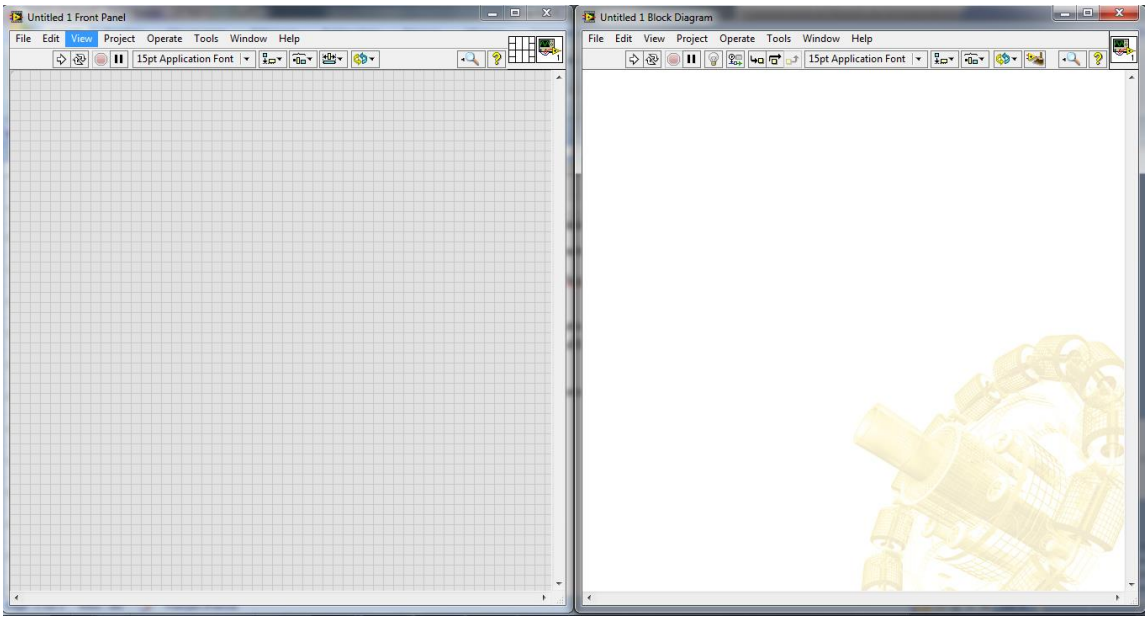


Figure III.3 : Frant Panel et Blok Diagram

III.5.Création d'un programme

Dans ce qui suit, On va essayer de créer un simple programme qui fait l'addition de deux chiffre (S=A+B). Après l'ouverture de deux fenêtres « Front Panel » et « Block diagram », on peut commencer notre programme par un clic droit de la souris sur le « Front Panel » là où s'ouvre une palette de control **(Figure III.4)**. Sur cette palette, on trouvera des sous-palettes (modern, silver, system…) et qu'on pourra d'ailleurs étendre en cliquant sur la flèche au-dessous de palette. Chaque sous-palette contient une gamme des variables, sur la **Figure III.4** on peut voir un tas de variables relatifs à la sous-palette : Modern.

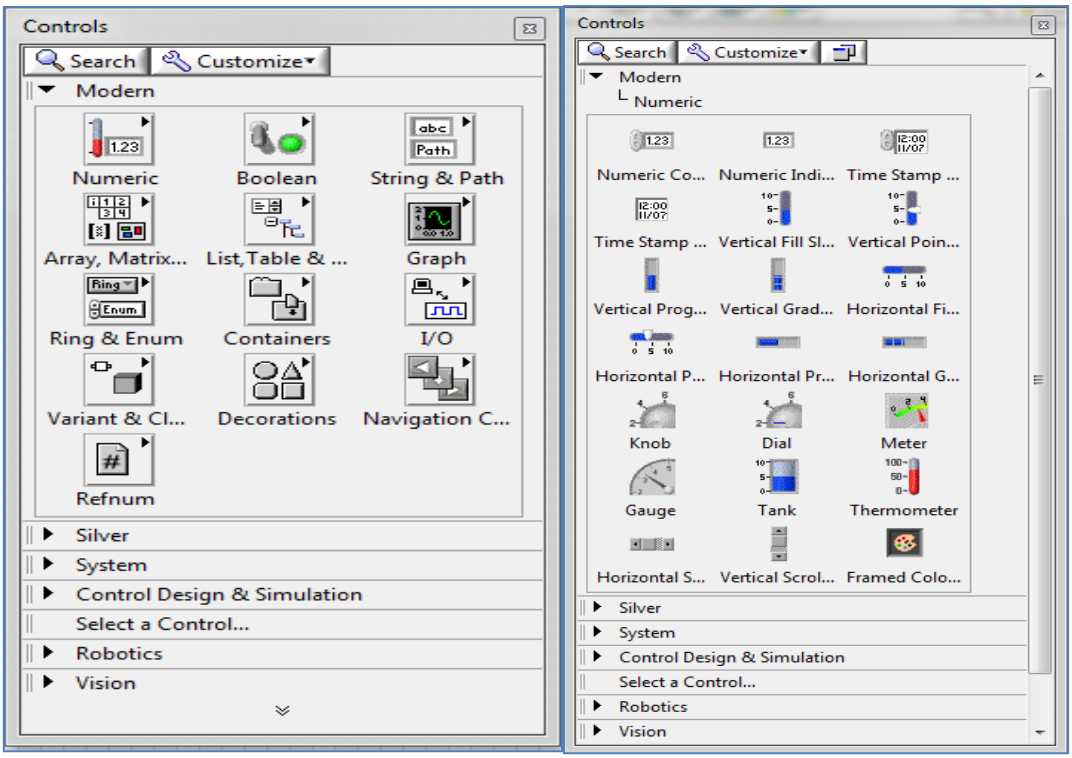


Figure III.4 : Control palette.

A Chaque variable, on peut affecter un type donné (numérique, booléen, string….) à laquelle correspond une couleur spécifique sur la fenêtre« block diagram » (numérique en orange, booléen en vert…etc.).

Par un simple clic sur une gamme de variable, un ensemble des contrôles (entré) s'affiche ainsi que des indicateurs (sortie).Pour commencer notre programme, On clique sur '*numeric control*'et on fait tirer le composant choisi sur la fenêtre « front panel »**(Figure III.5).**Après, on change le nom d'origine du composant par un double clic sur 'numeric', et on le mettra : A.

Par un simple clic sur le bouton droit de la souris, tout en restant sur le nouveau composant introduit sur le « Front Panel », une liste s'affichera, sur cette liste on peut changer le type du composant par '*representation*'. Pour notre cas, on changera le type de control vers indicateur en utilisant le champ 'change to indicateur', ….etc. Ceci peut également être fait en cliquant sur le champ 'properties', une nouvelle fenêtre apparait, sur celle-ci on peut changer le nom, le type, …etc.

En ce qui concerne la variable B, on suivra les mêmes étapes que celles faites pour la variable A.

Pour introduire la variable du résultat de l'opération « *S* », i.e. la somme*A* + *B*, on suivra également les mêmes étapes que celles faites pour les variables A, B.

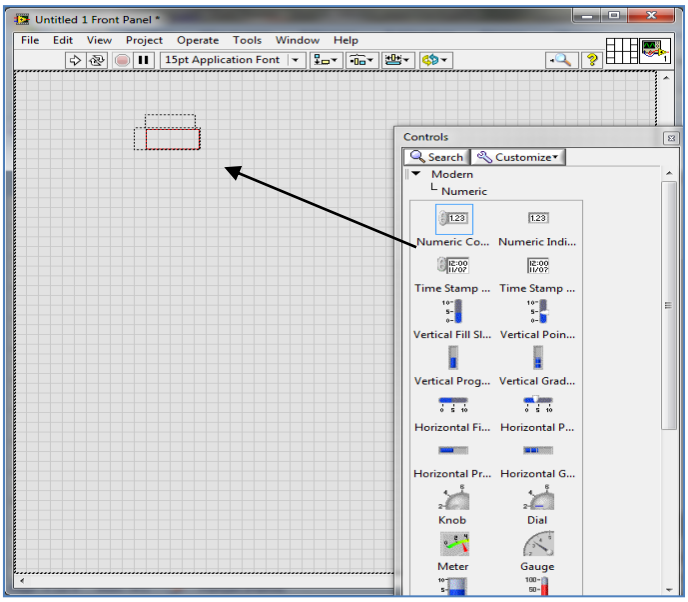


Figure III.5 : Tire « numeric control » sur le « front panel ».

Donc, les variables d'entrée/sortie du programme sont introduites au niveau de la fenêtre « Front Pane ». Pour implémenter le traitement qu'on va effectuer sur ces variables, on passera à la fenêtre « block diagram ». Ceci-est fait en passant par les commandes : **windows >> show block diagram.** Sur la fenêtre liée à ce block, on remarque que les trois variables, A, B et S, qu'on a créé dans le front panel *A, B et S* sont déjà là. Après ça, on clique sur le bouton droit de la souris, une action qui va ouvrir la palette des fonctions. Sur cette palette on trouve les différents fonctions (programme, measurerment I/O, robotics….), **(Figure III.6)**

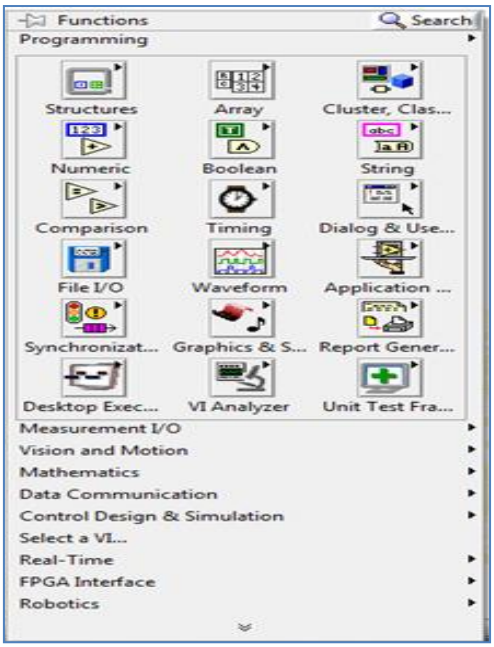


Figure III.6 : palette des fonctions.

On choisira l'objet 'numeric' dans la palette des fonctions et on sélectionne l'opérateur : 'add' et on le place sur le « block diagram » (**Figure III.7)**. Cet élément contient deux entrées et une sorte. Les deux entrées sont affectées aux variables *A*, *B*, tandis que la sortie est reliée à la variable *S*.

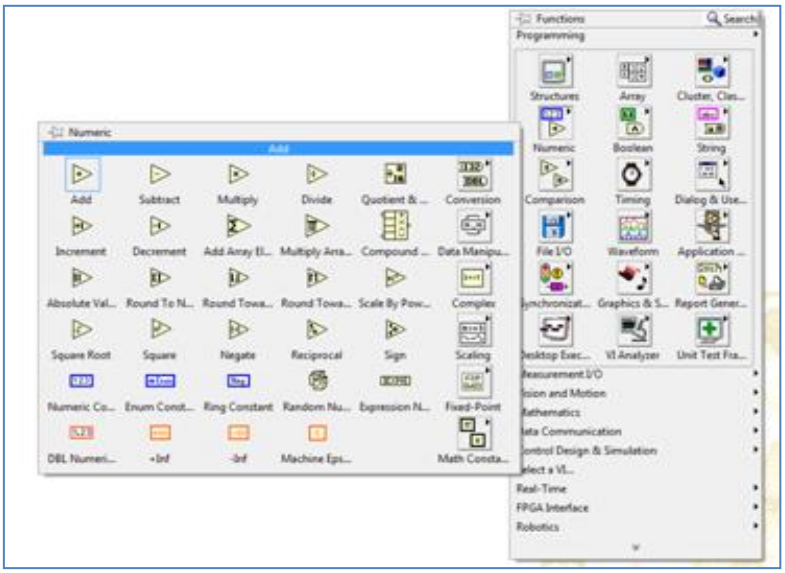


Figure III.7: la fonction « add ».

Notre programme est maintenant prêt, pour l'exécuter on clic sur le bouton « ***Run*** » (exécuter), celui-ci est localisé sur la (barre d'outils) en haut de la fenêtre « Front Panel » **(Figure III.8).**Le résultat final est affiché sur la **Figure III.9**.

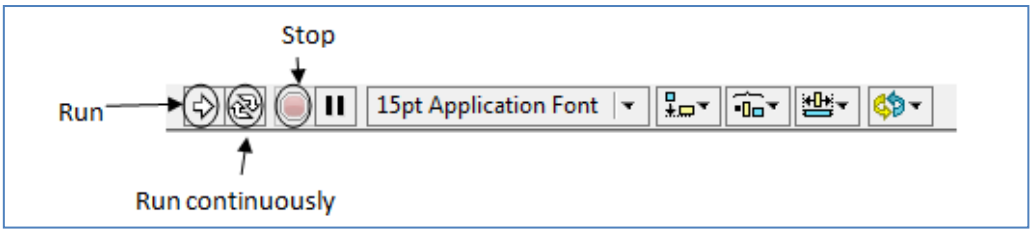


Figure III.8 : barre d'outils et bouton Run.

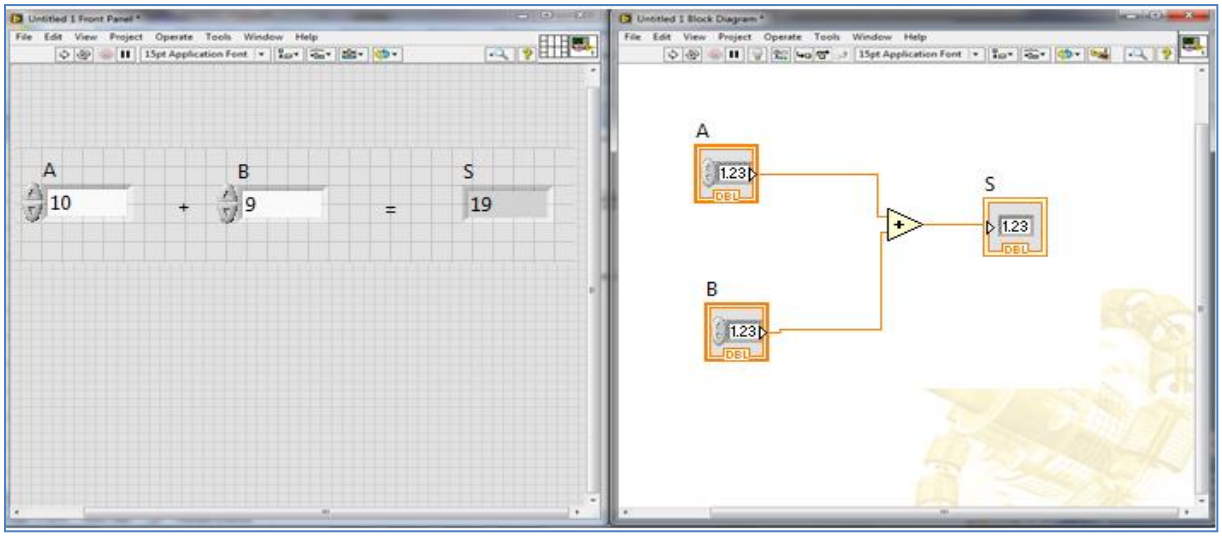


Figure III.9 : Le résultat final de notre programme.

Le programme s'exécute une seule fois et on parallèle. Mais si on clique sur le bouton « *Run Continuously* », le programme s'exécute infiniment jusqu'à on clique sur le bouton « stop » de la barre d'utile. Pour ce qui des applications futures, et pour contrôler l'exécution des programmes, qu'on aura à développer, on utilisera les structures et plus précisément les boucles.

III.6.Utilisation des structures

III.6.1.La boucle « For »

La partie du programme incluse à l'intérieur d'une boucle « For » s'exécute plusieurs fois selon le nombre d'itération spécifié dans une variable *N* représentant le maximum d'itérations à établir **(Figure III.10).** *N* étant la valeur câblée au terminal (***N***) de décompte de la boucle. Le terminal d'itération (**i**) fournit le nombre d'itérations actuel de la boucle, qui varie de 0 à *N*-1. La valeur 0 présente la première itération et 1 la seconde itération. Si on relie un 0 à N la boucle ne s'exécute pas. On peut ajouter une boucle « For » au programme d'addition précédent. La structure de ce programme est affichée dans la **Figure III.10**.

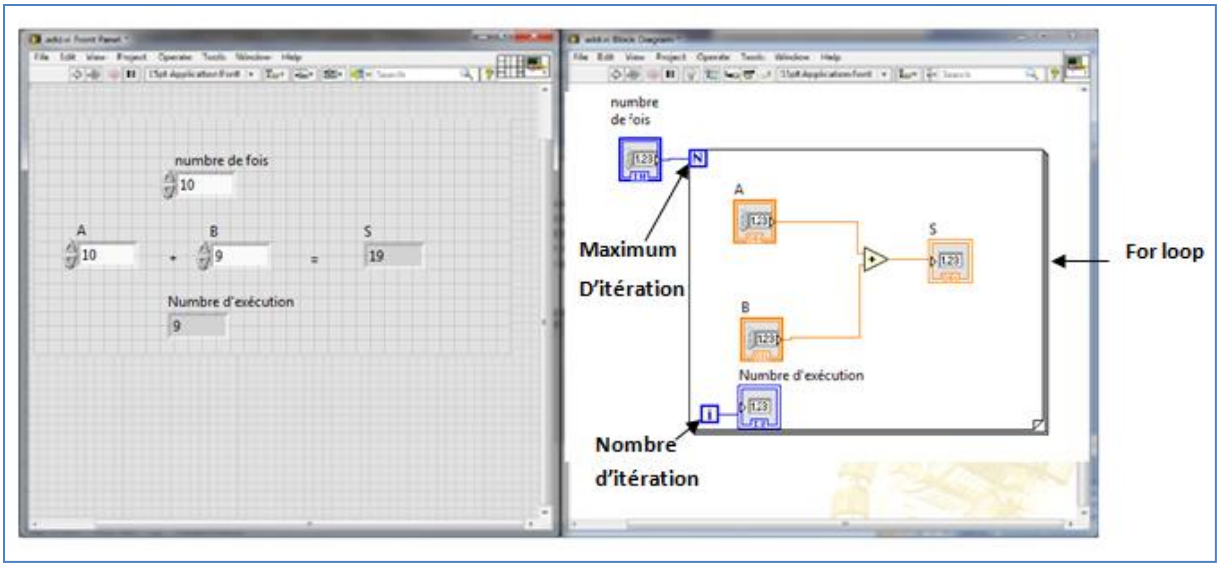


Figure III.10:Ajout d'une boucle "For" au programme d'addition précédent.

III.6.2.La boocle Tant que 'While'

La différence entre une boucle « Tant que » et une boucle « For »réside dans le fait à ce qu'une boucle « While » utilise une condition pour l'arrête l'exécution du programme comme par exemple dans le cas de l'exécution de « Do … while » dans la programmation textuelle. La boucle répète le sous-diagramme situé à l'intérieur de la boucle jusqu'à ce que le terminal de condition d'entrée reçoive une valeur booléenne particulière. La valeur booléenne dépend de la condition d'arrêt définie pour la boucle While. Le terminal d'itération (**i**) fournit le nombre d'itérations actuel de la boucle, qui vaut zéro à la première itération. On peut ajouter une boucle « While » au programme d'addition précédent. La structure de ce programme est affichée dans la **(Figure III.11).**

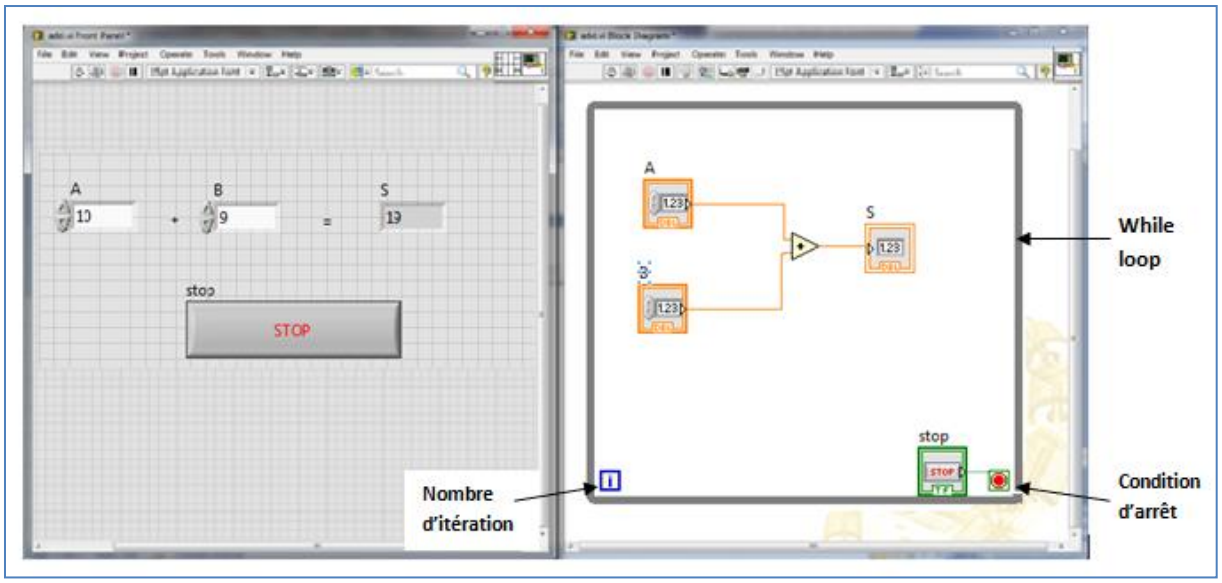


Figure III.11:Ajout d'une boucle "While" au programme d'addition précédent.

III.6.3.La structure "case"

La structure « CASE » comporte deux ou plusieurs sous-diagrammes ou cas. Un seul sous-diagramme est visible à la fois et la structure n'exécute qu'un seul cas à la fois. Une valeur de saisie détermine le sous-diagramme qui s'exécute. La structure « Case » est similaire aux déclarations de case ou si ... alors ... autres déclarations dans les langages de programmation basés sur le texte (**Figure III.12).**

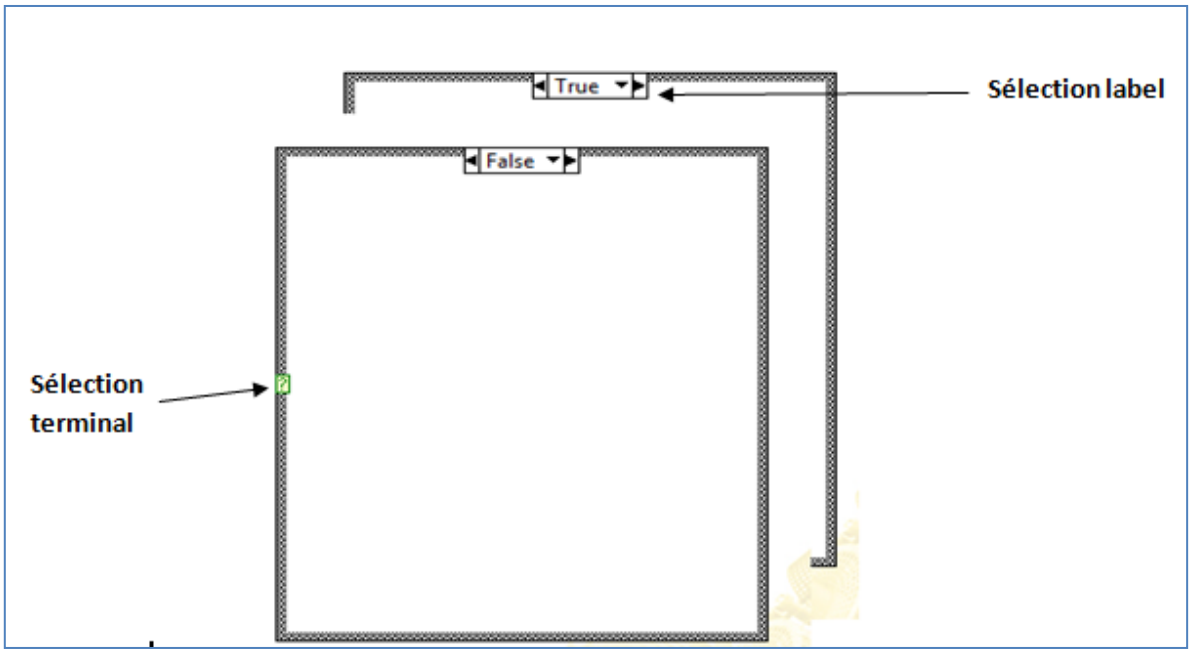
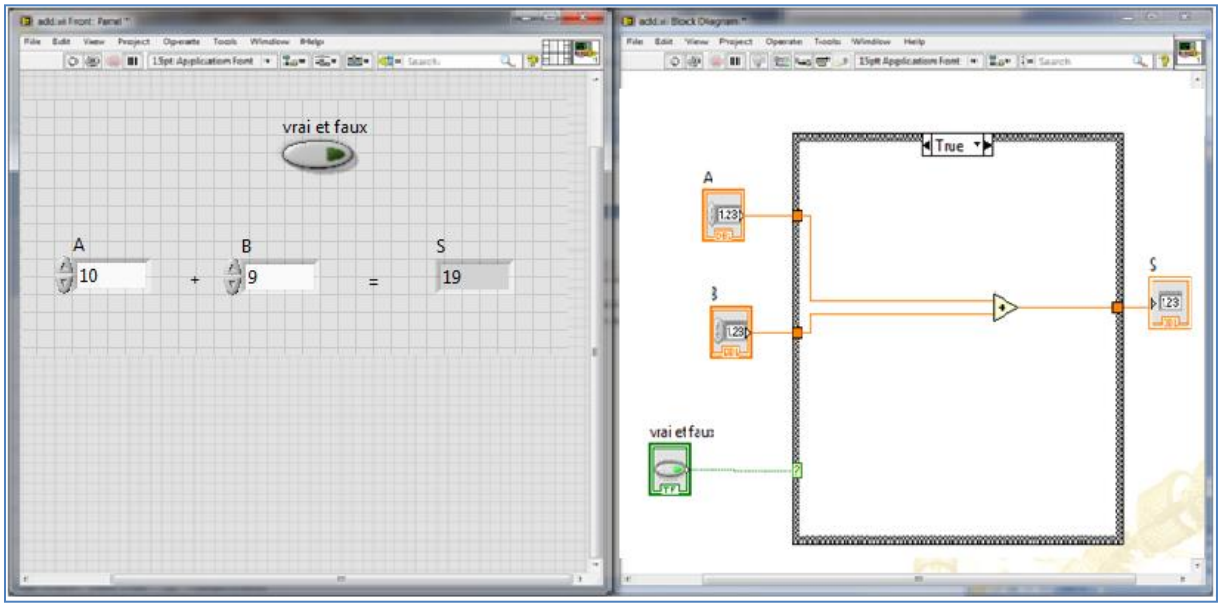


Figure III.12 : Les subdiagrams.

On applique case structure sur notre programme avec deux conditions vrais et faux, si la condition vrais addition les deux variables S=A+B, si faux soustraction les deux variables S=A-B. **(Figure III.13).**



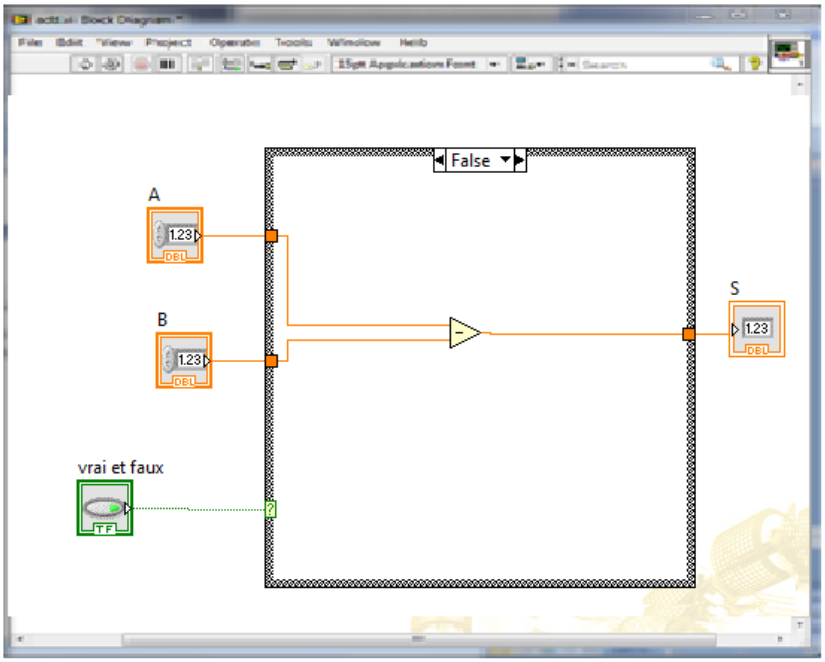


Figure III.13 : Ajout d'une structure "Case" au programme d'addition précédent.

III.7.Les graphes d'affichage

Graphs (graphe) sont des entités de sortie utilisés pour afficher les données sous forme graphique. Dans l'environnement de développement LabVIEW.

LabVIEW comprend les types de graphes et de graphes déroulants suivants :

- **Graphes et graphes déroulants** — Affichent les données acquises à une vitesse constante.
- **Graphes XY** — Affichent les données acquises à une fréquence variable et les données pour les fonctions à valeurs multiples.
- **Graphes et graphes déroulants d'intensité** — Affichent des données 3D sur un tracé 2D en utilisant des couleurs pour afficher les valeurs de la troisième dimension.
- **Graphes numériques** — Affichent les données en tant qu'impulsions ou groupes de lignes numériques.
- **Graphes de signaux mixtes** — Affiche les types de données qu'acceptent les graphes, les graphes numériques et les graphes XY. Accepte aussi les clusters qui contiennent des combinaisons de ces types de données.
- **Graphes 2D** — Affiche des données 2D sur un tracé 2D de face-avant.
- **Graphes 3D** — Affiche des données 3D sur un tracé 3D de face-avant **(Figure III.14)**.

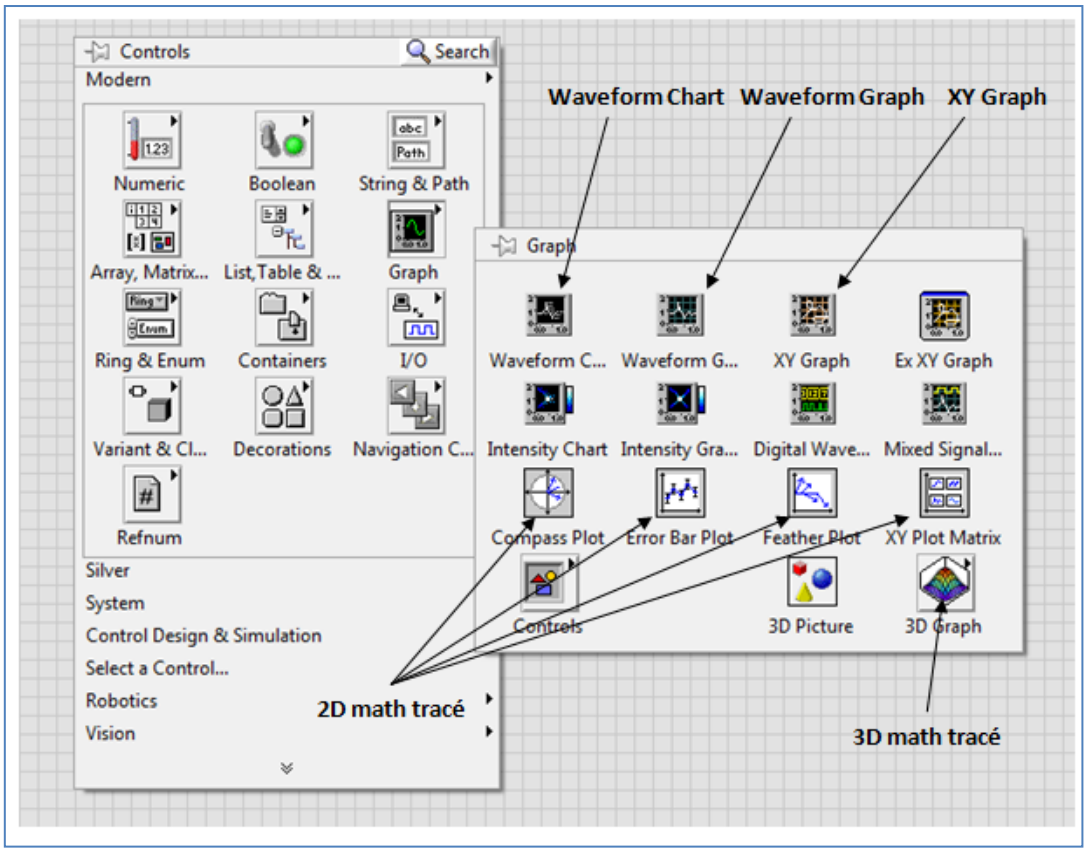
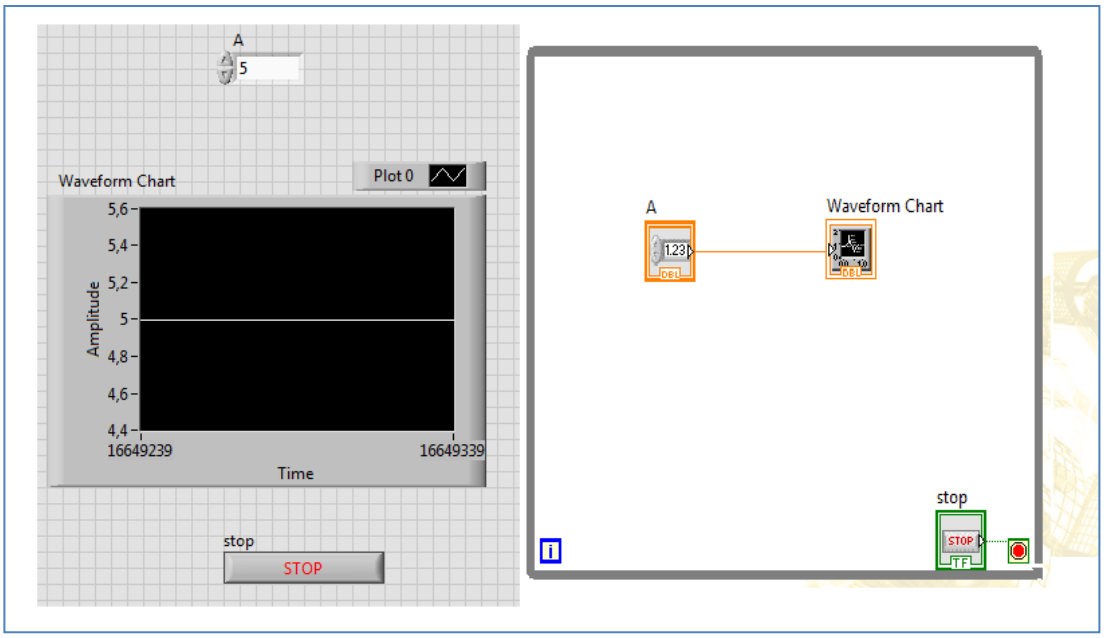


Figure III.14 : Les différents types des graphs en LabVIEW.

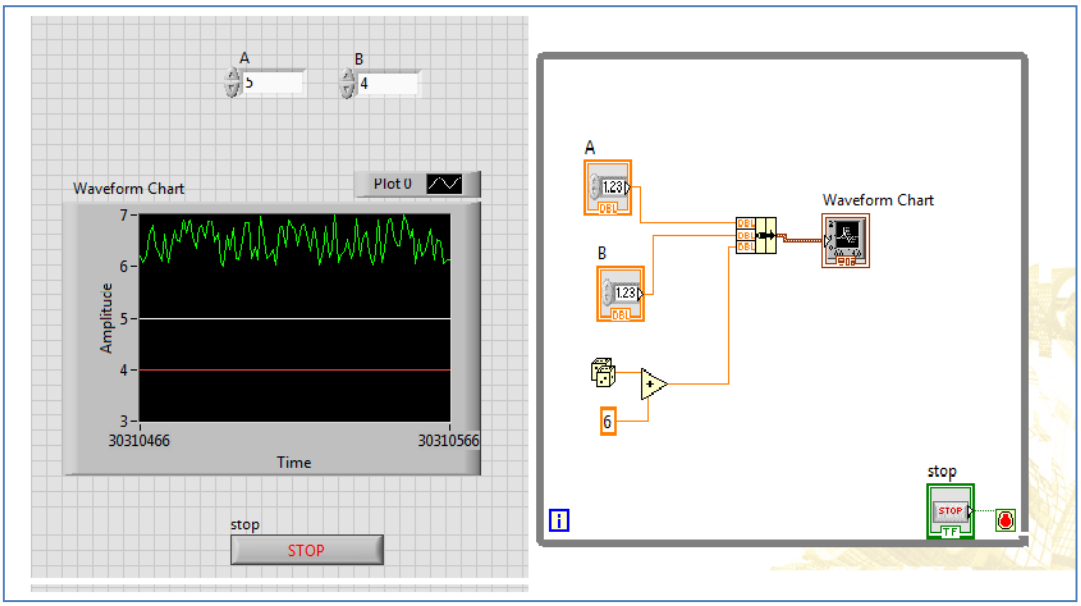
III.7.1.Les Graphes « Waveform Charts »

Il ya un seul type de « Waveform Charts » mais on trouve trois types de traçage ‘stip’, ‘scope’ et ‘sweep’. La différence entre ce trois type est la méthode de traçage le variable. On sélectionné un seul de ces types par click droit de la souris sur le graphe et on choisit Advenced >> Update Mode dans le menu.

« Waveform Charts »peut afficher un ou plusieurs signal, pour afficher uniquement un signale on relai une sortie scalaire vers le « Waveform Chart ». Il affiche la variation de sortie en fonction de temps. Pour afficher plusieurs signux, on utilise l’outil ‘Bundle’ qu’on trouve dans la palette des fonctions >> programming >> cluster. Celui-ci peut relier plusieurs variables à l’entrée de bundle et la sortie vers Waveform Charts. **(Figure III.15)**



(a)



(b)

Figure III.15 : (a) Un seul signal relié à« waveform charts » (b) plusieurs signaux reliés à un seul « waveform charts ».

III.7.2.Les graphes déroulants « Waveform Graphs »

Ce type de graphe affiche un vecteur des variables sur l'axe (Y) et contrôlés le temps sur l'axe (X). Par défaut l'initialisation de temps X_0=0, et le pas ΔX=1, on peut changer l'initialisation et le pas par l'utilisation de l'outil« bundle » **(Figure III.16)**

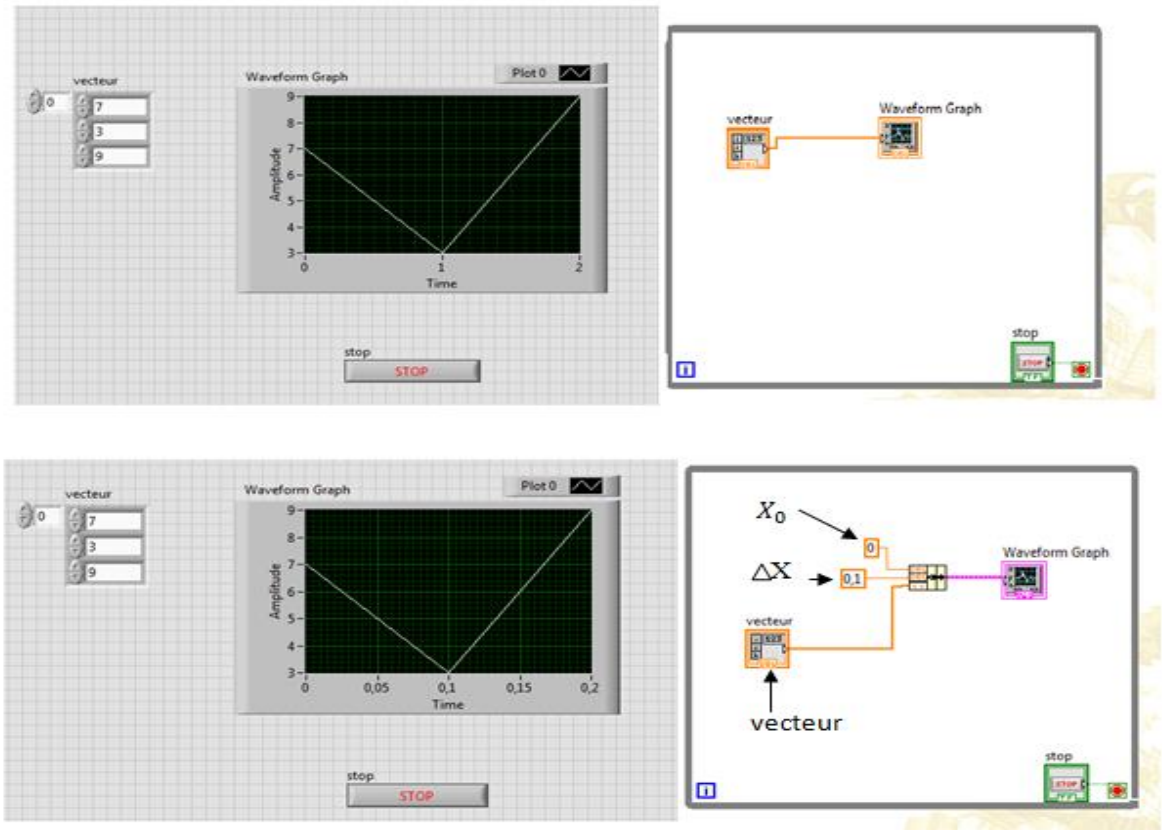


Figure III.16: Waveform graphs.

III.7.3.Les GraphesXY

Le graphe XY est un objet graphique cartésien qui affiche deux vecteurs sur forme d'un graphe **(Figure III.17),** le premier vecteur présent sur l'axe Y et le deuxième par l'axe X. Ce qui est différent par rapport « *Waveform graphs* » est que le pas est irrégulier. Pour relier les deux vecteurs à « *XY graphs* » on utilise l'outil « bundle ». La taille de l'affichage peut être changée en cliquant sur les chiffres d'extrémité supprimer l'ancienne valeur et la replacer par une nouvelle valeur.

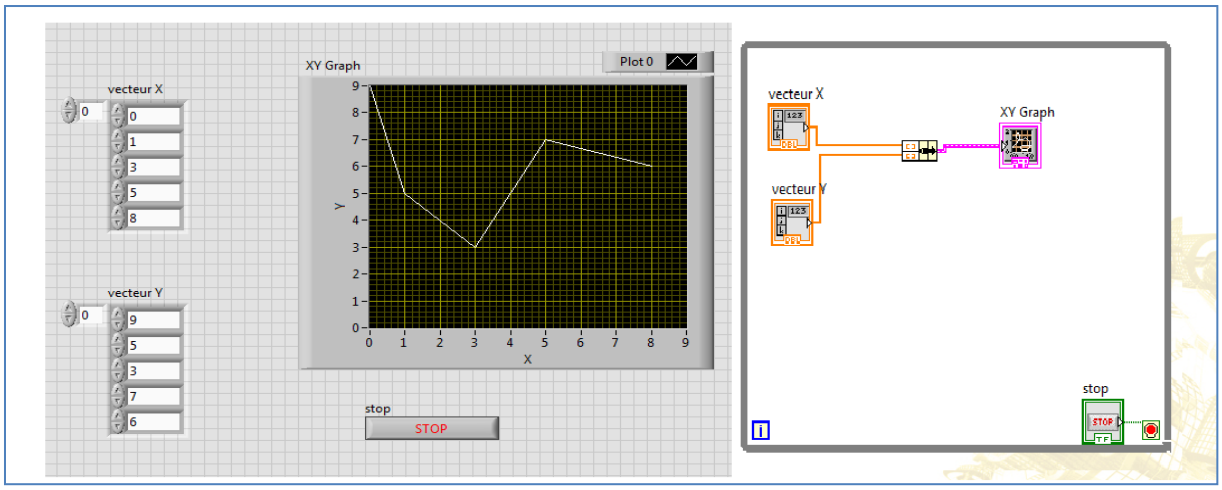


Figure III.17 : XY Graphs.

III.8.Les SubVI

Un « *SubVI* » est un sous-VI, c'est l'équivalent d'un sous-programme dans un langage de programmation textuel comme le C ou le Fortran. On peut appeler un « *SubVI* »à travers la palette des fonctions : *Function palette>>select a VI*, ou par utilisation des fonctions comme « robotic » **(Figure III.18)**, et créer par la suite un nouveau VI.

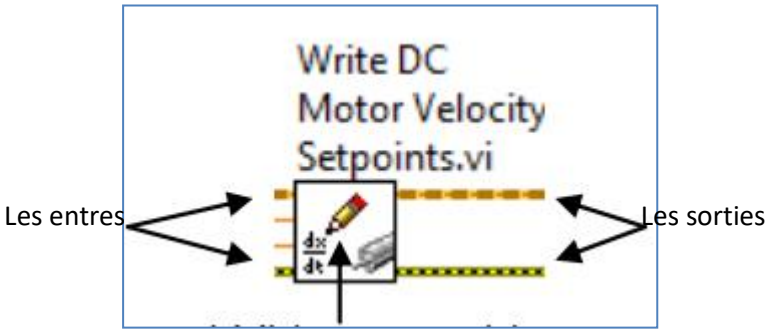


Figure III.18 : Structure d'un « *SubVI* ».

III.8.1.Etapes de création un SubVI

L'une des méthodes utilisée pour la création d'un nouveau « subVI », consiste à ouvrir un nouveau un « VI » et crée un programme dans celui-ci, par exemple un programme qui calculé la circonférence C=2πr et la surface S=πr^2d'un cercle. Après et à l'aide de la souris, on sélectionne tout le programme sauf les entres et les sorties comme s'est indiqué sur la (**Figure III.19).**

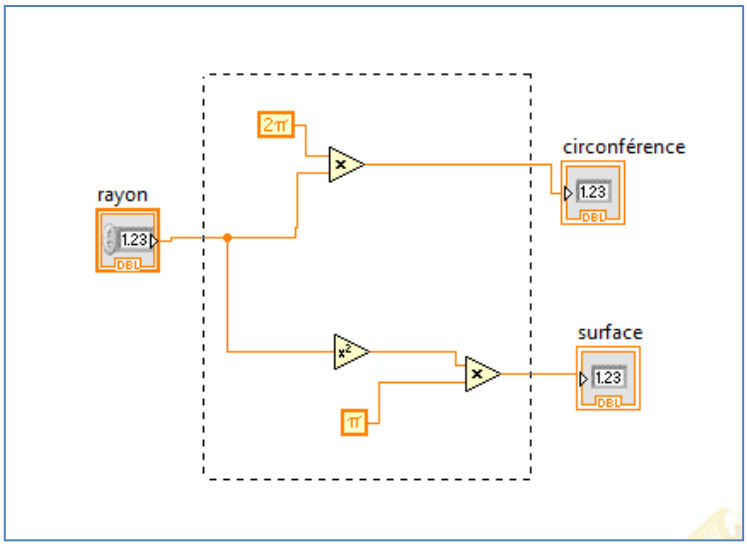


Figure III.19 : Cercle de mesure.

Puis, on clique dans la barre des tâches sur Edit >>*Creat SubVI*. Après on iravers l'outil « *default VI connector* »qui se trouve juste au-dessus de la fenêtre « front panel » **(Figure III.20).** Cet outil sera utilisé pour définir les entrés et les sorties du nouveau « SubVI » et ce en cliquant sur les rectangles gauches pour définir les entrées et puis sur les rectangles à droite pour définir les sorties. Pour changer l'interface graphique du « SubVI », on utilisera l'outil« *IconeEditor* » qui se trouve juste à droite du« *default VI connector* » **(Figure III.20)**

Figure III.20 : Les outils d'édition du SubVI : « *default VI connector* » et «*Icone Editor*»

Arrivés à un tel stade, on peut affirmer qu'on s'est un petit peu familiarisé avec l'outil logiciel de développement i. e. LabVIEW. On a repassé en vue les principaux outils et fonctions qui peuvent nous permettre d'aborder dorénavant les aspects logiciels qui vont nous permettre de configurer et de programmer le robot mobile DaNI 2.0. Il faut seulement noter ici qu'en plus du logiciel LabVIEW qui doit être installé sur l'ordinateur de travail, d'autre modules tels que : LabVIEW Robotics Module, LabVIEW Real-Time Module, LabVIEW FPGA Module doivent également être installés dans la même machine pour permettre le bon fonctionnement des programmes visant à contrôler le robot mobile DaNI 2.0.

Aussi avant de pouvoir programmer le robot mobile, nous devons tout d'abord connecter cet engin à l'ordinateur contenant les outils de développement logiciels utilisés pour configurer le robot. Et c'est ce qu'on va aborder dans la suite de ce chapitre.

III.9.Etablissement une connexion réseau (Ethernet) entre l'ordinateur et le robot DaNi

Dans cette partie, on va établer une connexion entre l'ordinateur qui contient LabVIEW et le robot DaNI 2.0.Cette connexion nous permettra de programmer et de récupérer les différentes informations provenant robot.

Pour cela on a suivi les étapes suivantes.

- On démarre le module LabVIEW Robotics tout en procédant de la manière : cliquer sur démarrer >> tous les programmes >> National Instrument >> Labview robotics >>robotics Hardware setup. A la fin de cette sous-étape, une fenêtre s'affiche comme l'indique la **Figure III.21.**

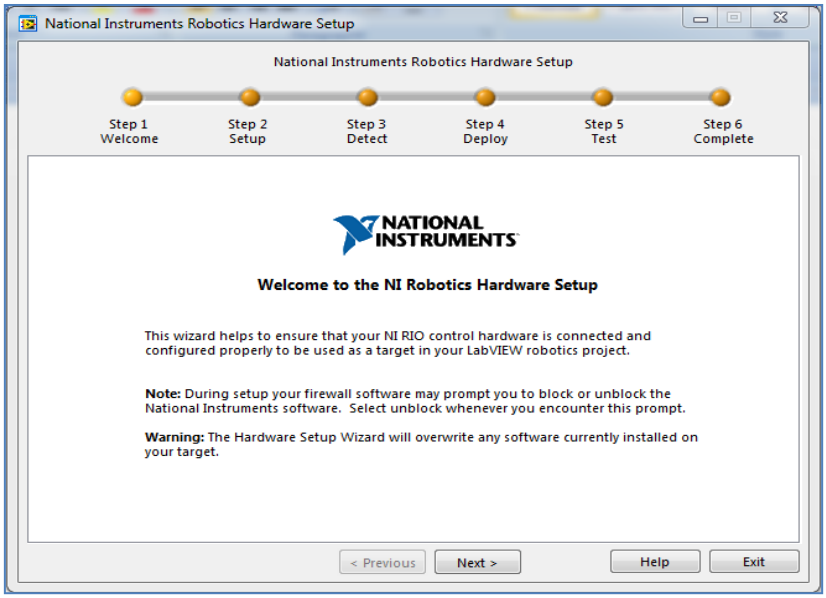


Figure III.21: Fenêtre Hardware setup.

La fenêtre Assistant de configuration du matériel illustré à la Figure III.21 va nous guider à travers plusieurs étapes pour établir une communication entre l'ordinateur hôte et DaNI. Après la lecture du contenu de cette fenêtre on clique sur *next*, une nouvelle étape est franchie, comme le montrée le **Figure III.22**.

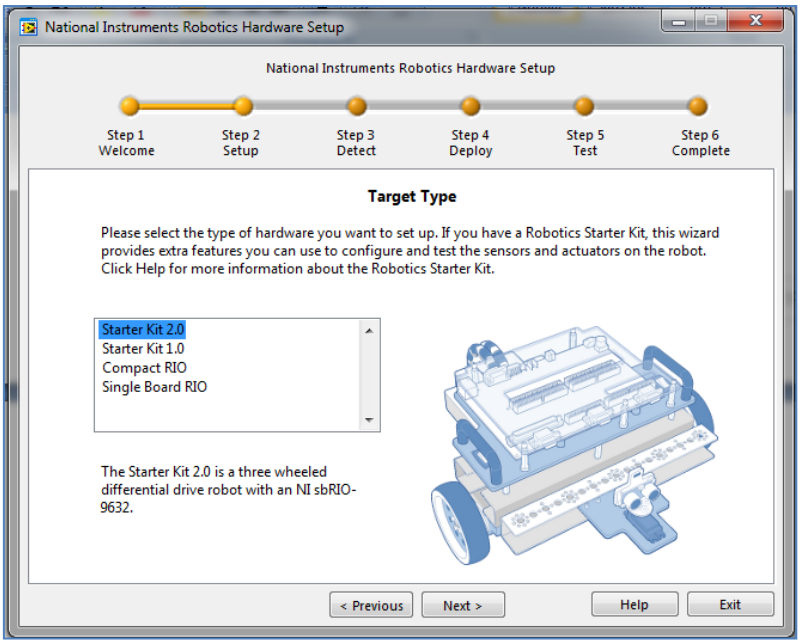


Figure III.22 : Fenêtre de sélection du type de cible.

La fenêtre illustrée à la Figure III.22 nous ouvre la possibilité de configurer du matériel pour différents types de cibles (matériel utilisé dans les systèmes robotiques). On choisira le type de cible, qui est dans notre cas : Starter Kit 2.0 et puis on cliquera sur le bouton Suivant>une nouvelle fenêtre s'affiche similaire à celle de la **Figure III.23**.

Sur cette fenêtre, on nous rappelle de connecter les moteurs et les capteurs au robot et à l'ordinateur du robot avant de continuer, et pour nous assurer que les moteurs sont en position arrêt. Si les moteurs et les capteurs sont connectés à DaNI, on clique sur le bouton Suivant>.

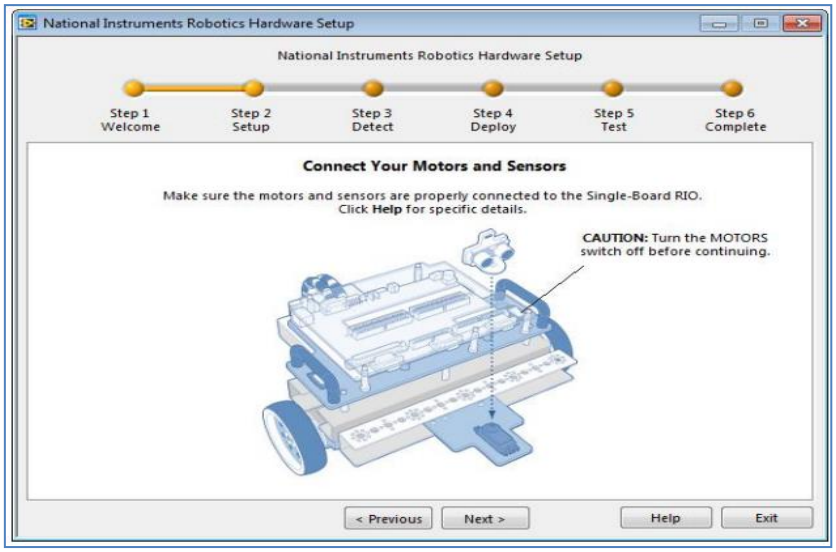


Figure III.23 : Fenêtre de connexion moteurs et capteurs.

La fenêtre suivante, **Figure III.24**, nous donne deux étapes, Ethernet et les connexions électriques. Elle nous invite à Connecter le câble Ethernet croisé avec le kit entre la carte sbRIO sur DaNI et l'ordinateur hôte. L'alimentation est déjà connectée, mais on doit quand même nous assurer que la batterie est chargée et que l'interrupteur principal est en position ON. Les commutateurs DIP et les LED seront expliqués plus tard, ici on vérifie simplement que les commutateurs DIP sont tous en position OFF et qu'une seule LED est allumée. On peut également ignorer la partie CompactRIO de la fenêtre.

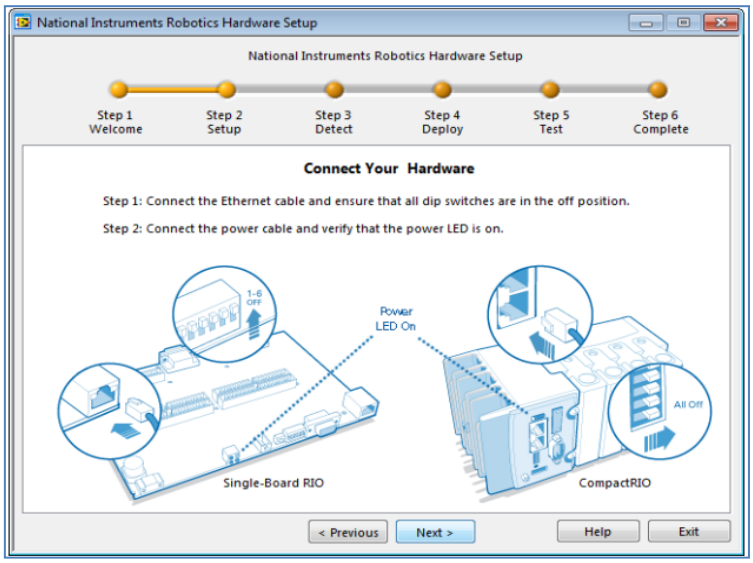


Figure III.24 : Connexion Ethernet et alimentation.

Après avoir cliqué sur le bouton Suivant>, il a fallu attendre quelques secondes pour que la cible ait été reconnue. Une fois le matériel détecté et une adresse IP attribuée comme indiqué dans la Figure ci-dessous, nous avons vérifié que nous avons sélectionné le périphérique avec le numéro de série correct. La série est située sur un petit autocollant vert dans le coin inférieur droit du RIO Single-Board. **(Figure III.25).**

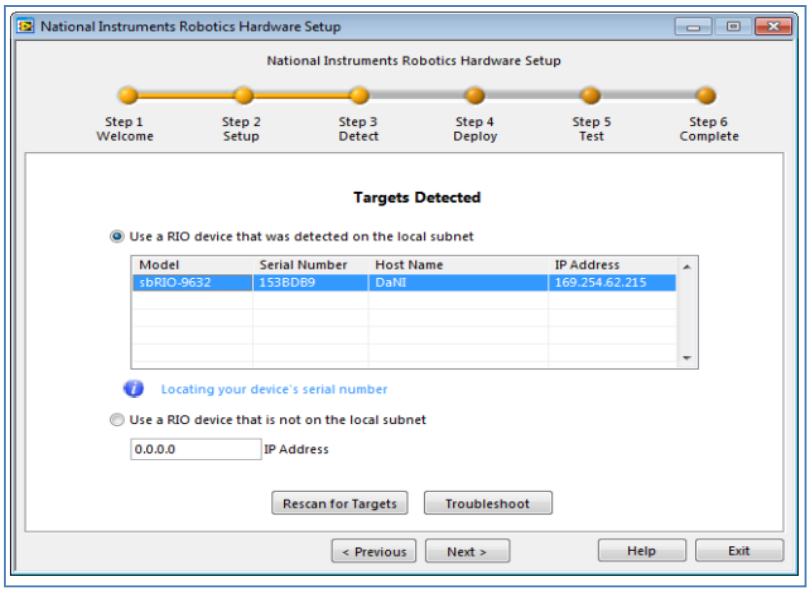


Figure III.25 : Fenêtre des cibles détectée.

Étant donné que le système peut supporter plusieurs cibles RIO CompactRIO ou Single-Board connectées au même sous-réseau que l'ordinateur hôte, l'Assistant de configuration matérielle renvoie le type de modèle et le numéro de série de chaque cible disponible sur la page Détecter le matériel. Cela nécessite que qu'on choisisse le périphérique avec un clic de souris pour qu'il soit en surbrillance, comme indiqué sur la **(Figure III.25)**. A la fin on cliquera sur Suivant> pour continuer avec l'assistant, ce qui lui permet d'installer un logiciel et de copier des fichiers sur la cible RIO unique. Cela prendra quelques minutes et la fenêtre montrée dans la **Figure III.26** sera affichée.

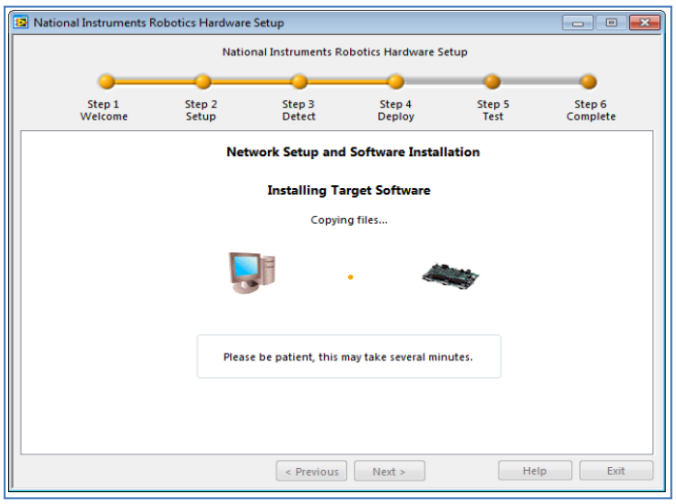


Figure III.26 : L'installation software.

Une fois le logiciel installé, la fenêtre de la Figure III.28 s'affiche, montrant un déploiement réussi. On clique sur *next*.

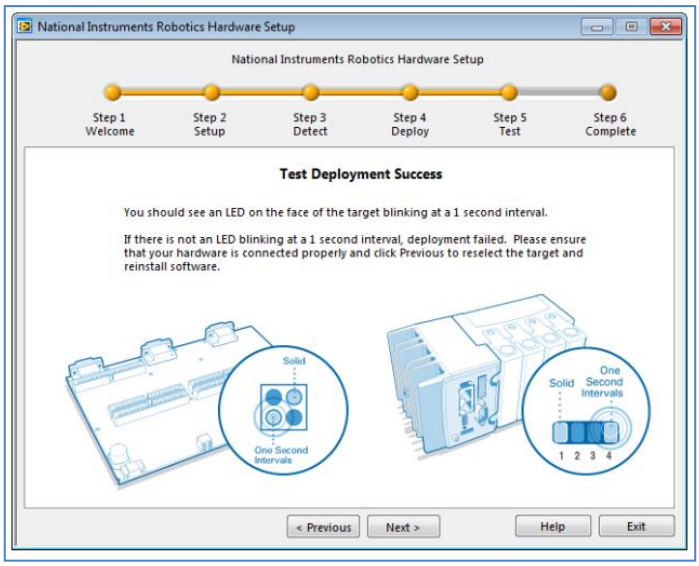


Figure III.28 : Fenêtre de déploiement réussi.

Une nouvelle fenêtre s'affiche sur cette fenêtre on peut orienter et tester le capteur ultrason PING))). On peut orienter le capteur par l'utilisation du curseur montré dans le **Figure III.29**. Après on clique sur *next*.

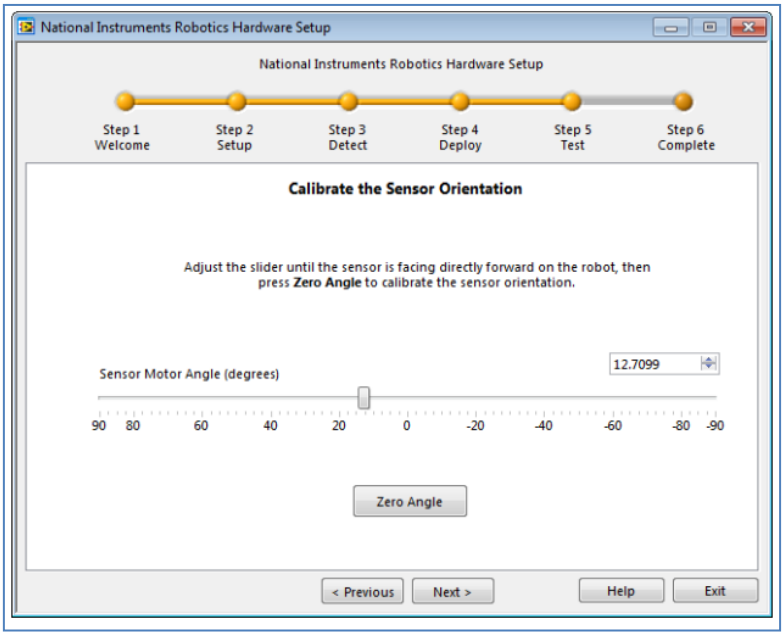


Figure III.29 : Orientation du capteur ultrason PING))).

Un graphique montrant le signal de distance (plage) de PING))) s'affiche comme indiqué sur la **Figure III.30**. Ici on doit s'assurer que le capteur fonctionne correctement en localisant DaNI orthogonal sur une surface plane qui reflète le son, comme un mur ou le côté d'une boîte, à une distance connue (dans la plage de 0,5 à 3 m) devant le capteur. On a souvent attendu quelques minutes après avoir placé un objet dans le champ de vision du transducteur ultrasonique pour observer la distance correcte car les signaux de distance précédents peuvent encore être en mémoire.

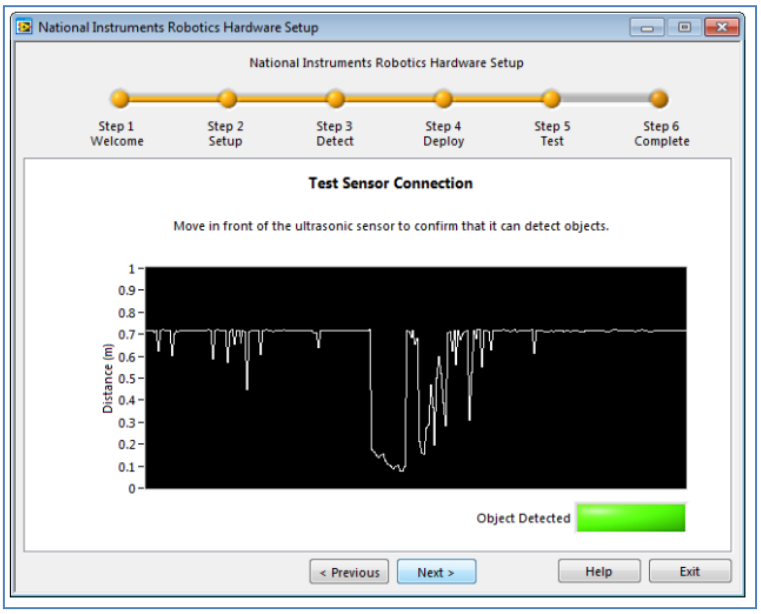


Figure III.30 : Fenêtre de test de la connections du capteur.

Après avoir testé PING)), on cliquera sur Suivant>> pour tester les moteurs et les encodeurs. On a Placé DaNI dans une zone où il peut se déplacer en toute sécurité, On a mis le bouton Moteur sur ON et utiliser les curseurs illustrés à la **Figure III.31**. Après avoir terminé le test des moteurs on a cliqué sur *next*. **(Figure III.31)**

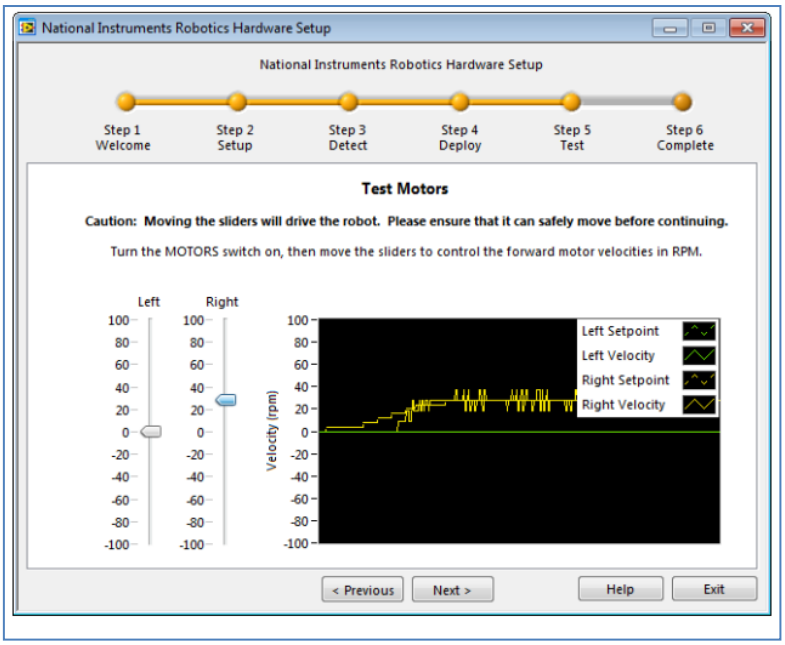


Figure III.31 : Fenêtre du test des moteurs.

Ceci complète la configuration matérielle et la fenêtre illustrée à la **Figure III.32**, s'affiche. On coche la case «Créer un nouveau projet de robotique dans LabVIEW» et on clique sur le bouton Terminer (il n'y a pas de bouton Quitter comme indiqué dans les instructions dans la fenêtre) **(Figure III.31)**

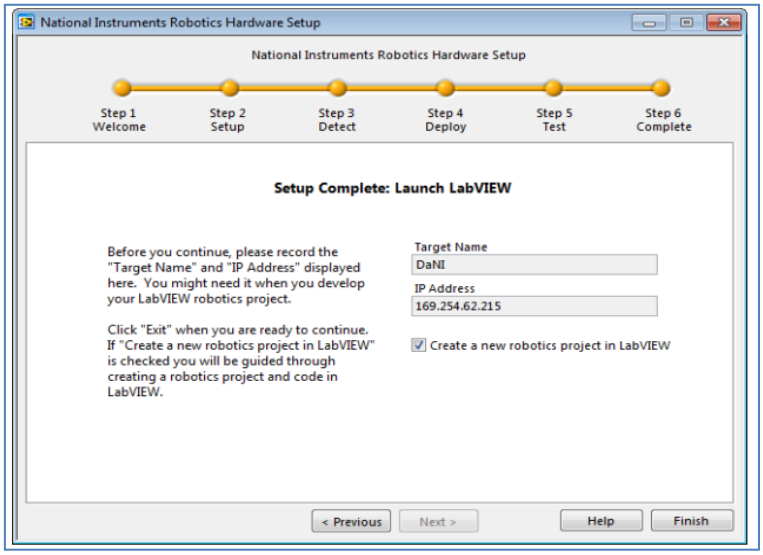


Figure III.32 : Fenêtre de l'étape 6 montrant l'achèvement réussi de la configuration matérielle.

Après avoir pu établir une connection Ethernet entre le robot et le PC contenant les outils de développement logiciel, on peut maintenant passer à l'étape de création d'une nouvelle tâche et qui va concerner la programmation du robot DaNI 2.0.

III.10.Création robot projet

Dans la section précédente, l'assistant de configuration matériel a créé un projet automatiquement pour nous et ce suite au cochage de la case (Create new robotics project in LabVIEW).

Dans cette partie on va créer un projet sans utilisation de l'assistant de configuration matérielle, pour cela, on lance LabVIEW **(Figure III.2)**, on sélectione« Create Project », un nouvelle fenêtre **(Figure III.33)** va s'affichee en conséquence. Sur cette fenêtre**,** on choisira Robotics >> Robotics Project.

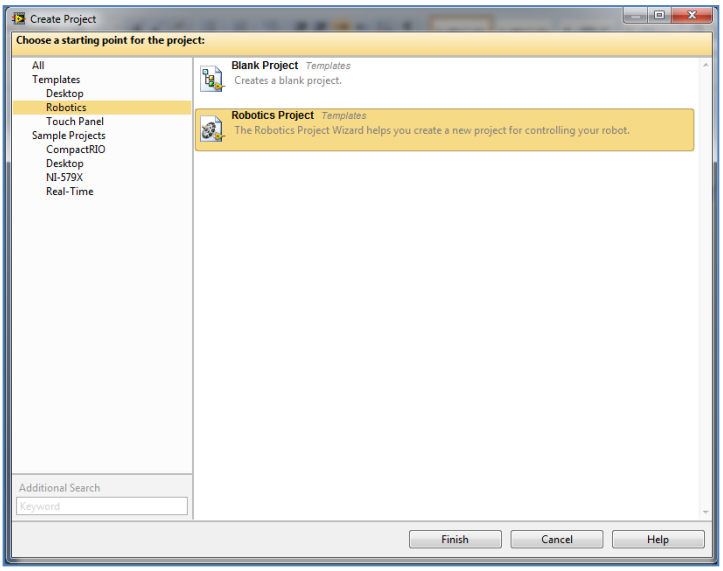


Figure III.33 : Coix du type de projet.

Une autre fenêtre similaire à celle de la figure **(Figure III.33)** va s'afficher, on sélection 'Robotics Start Kit 2.0', ce qui correspond à la référence du robot mis à notre disposition. A la fin, on cliquera sur next.

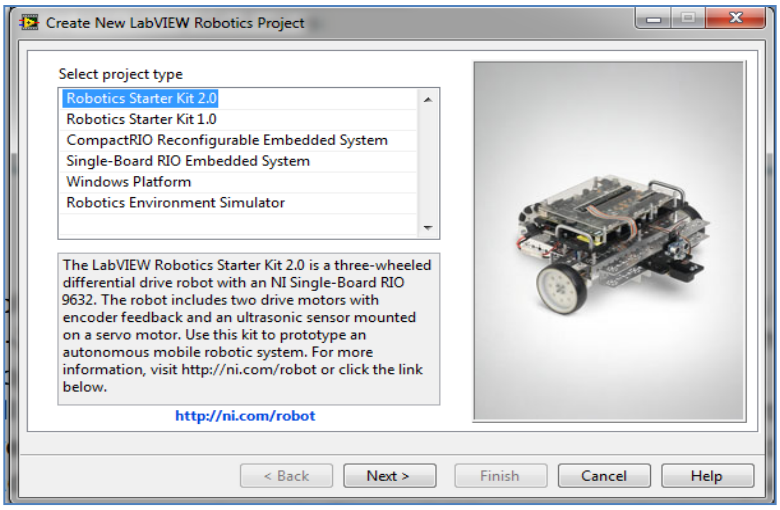


Figure III.33 : sélection type de robot.

Une autre fenêtre s'affichera, sur cette fenêtre, on peut introduire l'adresse IP correspondant à notre robot et on click sur next.**(Figure III.34)**

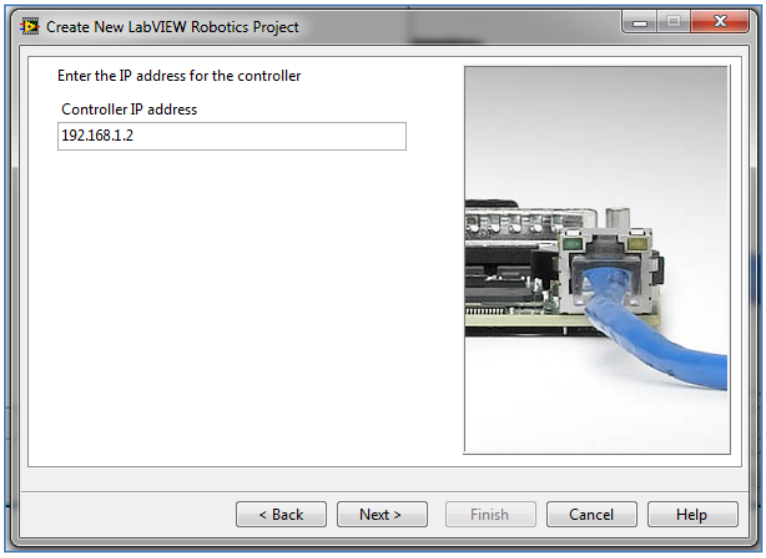


Figure III.34 : Configuration de l'adresse IP.

Lors de l'étape suivante, s'affiche une nouvelle fenêtre contenant une liste, sur cette liste on peut choisir le type de programme à téléverser dans la carte de contrôle du robot. Pour notre cas, on peut par exemple sélectionner 'Empety Application Shell' comme c'est montré sur la **(Figure III.35).**

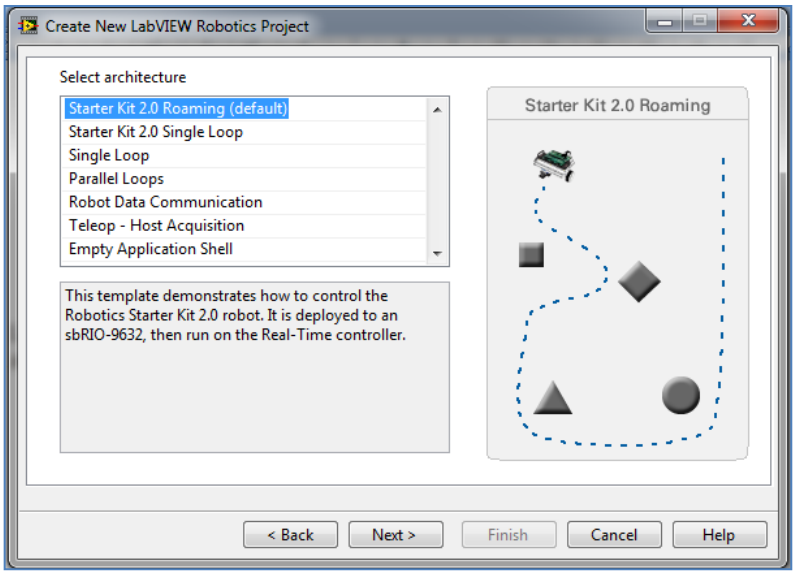


Figure III.35 : Selection architacteur.

La fenêtre suivante est simple fenêtre de confirmation du nom du projet et de son emplacement. Sur celle-ci, on peut soit garder les propositions du logiciels ou chager soit le nom du projet et/ou son emplacement selon la situation. Pour finir cette tâche entrer effectivement dans la phase de configuration du roboton clique sur le booton « Finish ». **(Figure III.36).**

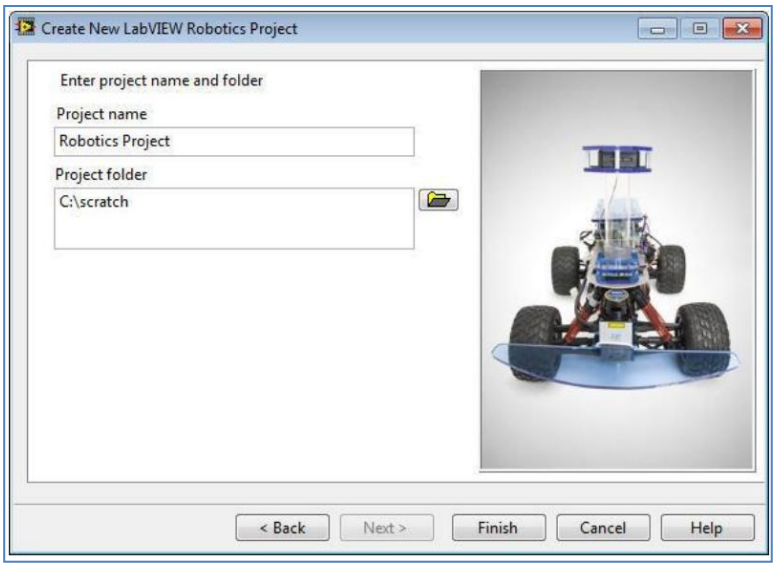


Figure III.36 : L'emplacement et le nom de projet.

III.11.Conclusion

LabVIEW est un logiciel de programmation graphique complet dont les aspects ne peuvent pas être couverts dans seul chapitre tels celui qu'on vient de repasser. On s'est juste contenté du strict nécessaire et ce pour aborder la programmation do robot mobile DaNI 2.0.Lors de ce chapitre, on a également vu comment peut-on connecter notre robot à un PC de bureau via une liaison Ethernet. Arrivés à un tel stade, on peut maintenant passer à la programmation du robot proprement dite. Ceci va faire sujet du prochain chapitre.

Chapitre VI

Programmation le robot DaNi

IV.1.Introduction

Après avoir passé en revue la partie Hardware de notre robot 'DaNI' 2.0 etl'examen des outils Software de programmation, ce chapitre sera consacré à la description plus au moins détaillée des programmes effectivement réalisés sous le Langage de programmation LabVIEW et ce pour le contrôle et le guidage du robot mobile.

IV.2.But du travail à réaliser

Le but essentiel des programme réalisés dans le contexte de ce travail est de rendre le robot « DaNI » autonome dans son déplacement et ce en évitant un tas d'obstacles aléatoire susceptibles d'être trouvés dans un milieu d'exploration relativement inconnue.

IV.3. Description générale du programme

Comme tout programme développé sous le langage LabVIEW, notre programme est composé de deux fenêtres la première fenêtre c'est le« *Front Panel »*montré dans La **Figure IV.1** et la deuxième fenêtre présente le « *Block Diagram »* montré dans la **Figure IV.2**.

- *Le front panel :* est une interface graphique qui contient un graphe qui va afficher en temps réel les déférents obstacles détectés par le robot via son capteur ultrason PING))) et un bouton de stop pour arrêter la navigation du robot le robot.

- *Block diagram* : Cette fenêtre contient le code source du programme, celui-ci est constitué d'un ensemble des « SubVI » qui contribuent ensemble pour réaliser les fonctions requises. Sur ce programme on trouve des « SubVI » pré-élaborés et appartenant au logiciel LabVIEW et surtout au module LabVIEW Robotics, voir **Figure IV.2**, et on trouve également d'autres« SubVI » qu'on a créé nous même au cours du développement de ce programme.

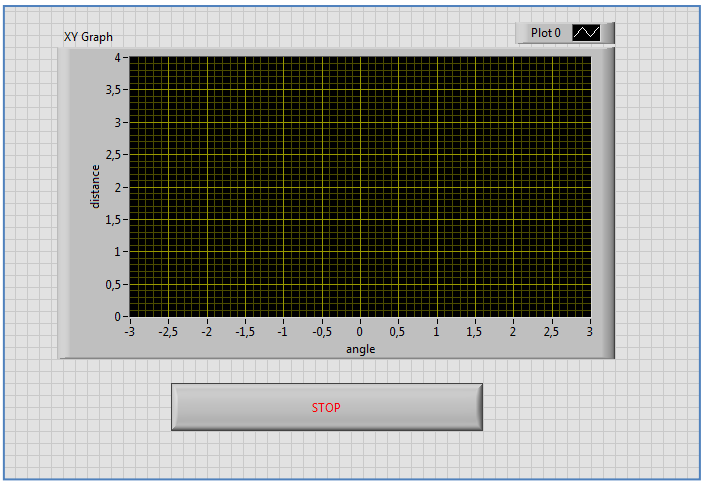


Figure IV.1 : Vue de la fenêtre « Front Panel » du programme principal.

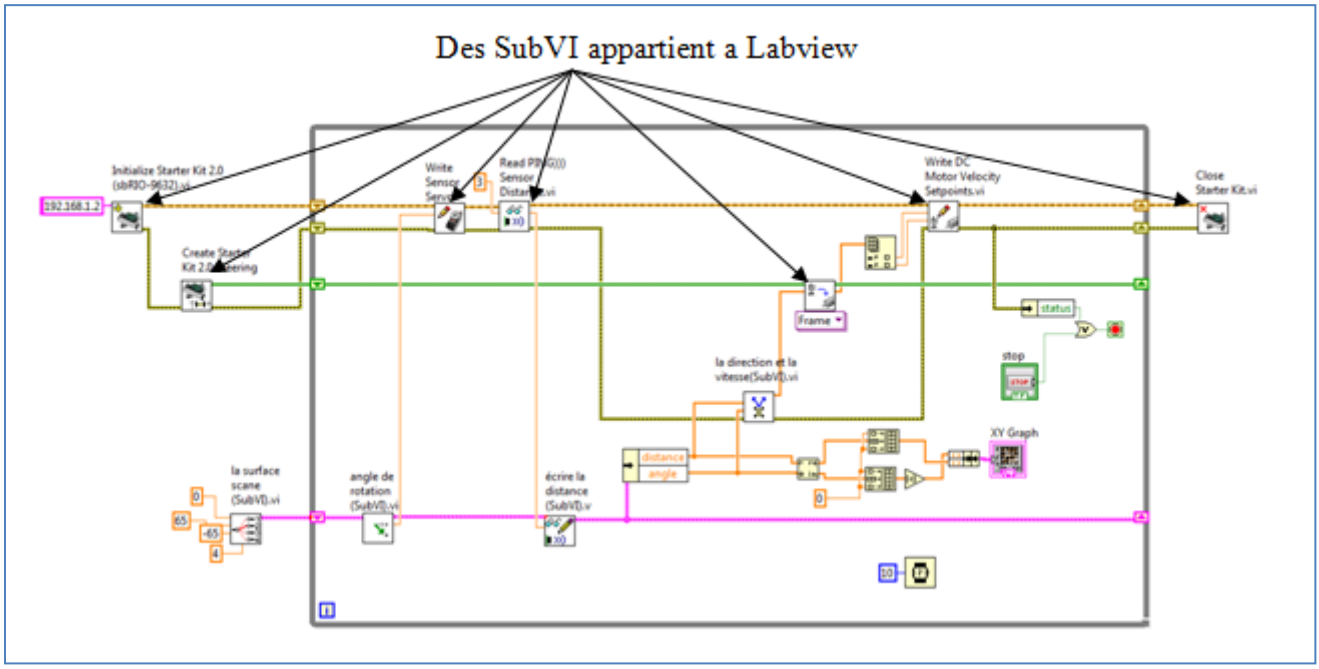


Figure IV.2 : Le code source de programme de guidage du robot.

IV.3.1.Description du fonctionnement du programme

Pour comprendre la manière avec laquelle ce programme fonctionne, on doit tout d'abord fixer le sens de l'évolution et le chemin d'exécution des différentes tâches. Donc premièrement on doit se rendre compte que l'exécution du programme commence de la gauche vers la droite. Le point de départ est le SubVI ***Initialize Starter Kit 2.0*** et le point d'arrêt est le ***Close Starter Kit*** tout à fait à droite.

L'exécution d'une tâche au sein de chaque *SubVI* commence lorsque celui-ci reçoit à son entrée un signal« START » ou s'il y'a un changement dans ses paramètres d'entrées. Une

exception est faitepour le *SubVI* ***Initialize starter kit2.0***à l'intérieur duquel l'exécution de la tâche commence automatiquement lorsqu'on lance le programme.

Dans ce qui suit, nous allons expliquer en détail le rôle et la tâche réalisée dans chaque *SubVI* figurant le *Block diagram* dans ce programme principal :

IV.3.1.1.Le SubVI ***Initialize starter kit 2.0 :***

C'est l'un des composant de la bibliothèque du module LabVIEW Robotics et qu'on peut retrouver dans la palette *Function>> Robotics>>Starter Kit 2.0.*

Il commence une session de communication avec FPGA sur le robot qu'on veut contrôler et renvoie une référence qu'on utilisera pour lire ou écrire sur le circuit FPGA. On doit tout d'abord appeler ce SubVI avant d'accéder aux E/S du circuit FPGA.

Figure IV.3: Le SubVI*: Initialize starter kit2.0.*

IV.3.1.2. Le SubVI ***Create Starter Kit 2.0 Steering Frame :***

C'est l'un des composant de la bibliothèque du module LabVIEW Robotics starter kit et qu'on peut retrouver dans la palette *Function>> Robotics>>Createstarter Kit 2*.0définit le type de roue et leur géométrie par rapport au châssis du robot Dani.

Il définit les positions *x* et *y* de toutes les roues par rapport au centre du cadre de direction et spécifie les angles des roues par rapport à la direction de déplacement. Lorsque ce VI s'exécute, ilPlace toutes les informations dans la mémoire de l'ordinateur pour l'utilisation par d'autres *VIs* dans le programme et crée un numéro de référence pouvant être connecté aux *VIs* descendants afin qu'ils puissent accéder à la mémoire.

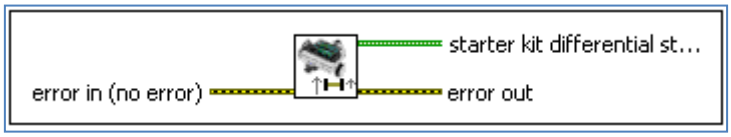


Figure IV.4: Le SubVI*: Create Starter Kit 2.0 Steering Frame.*

IV.3.1.3. Le SubVI ***Write sensor servo angle***

C’est l’un des composant de la bibliothèque du module LabVIEW Robotics starter kit et qu’on peut retrouver dans la palette *Function>> Robotics>>Write sensor servo angle.*

Ce VI permettra d’écrire un angle de rotation au servomoteur pointer surle cadre DaNi qui implique un changement d’orientation du capteur ultrasonore. Le fil supérieur est un fil de référence qui identifie le périphérique initialisé dans le cas où le VI contient plusieurs dispositifs. Et le deuxième fil « *ccw servo angle (rad*) » spécifie l'angle en radians, pour faire tourner le servomoteur du capteur. L'angle est mesuré dans le sens inverse des aiguilles d'une montre au centre du robot lorsque vous regardez le robot d'en haut. Les valeurs positives font tourner le servomoteur vers la gauche du centre, alors que les valeurs négatives le font tourner vers la droite.

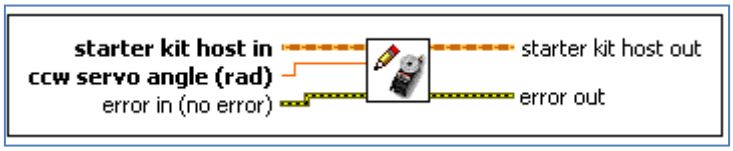


Figure IV.5: Le SubVI : *Write sensor servo angle.*

IV.3.1.4.Le SubVI ***Read PING Sensor Distance:***

C’est l’un des composant de la bibliothèque du module LabVIEW Robotics starter kit et qu’on peut retrouver dans la palette *Function>> Robotics>>Read PING Sensor Distance.*

Ce composant Lit et renvoie la distance entre le capteur de distance à ultrasons PING))) et l'obstacle le plus proche détecté par le capteur. Le capteur PING))) a une plage de détection maximale de trois mètres. Si le capteur ne détecte aucun obstacle à portée, la propriété : temps expiré est vraie.

On peut utiliser le *Sensor Servo Angle VI* pour changer l'orientation du capteur, puis utiliser *Read PING Sensor DistanceVI*pour lire la distance à tous les obstacles à cet angle du capteur. On associe l'angle de servomoteur avec le *Sensor Servo Angle VI* et la distance que ce VI lit pour identifier l'emplacement exact des obstacles par rapport au robot. Lorsqu’on connaisse l'angle et la distance à un obstacle, on peut écrire un code pour orienter le robot pour éviter les obstacles.

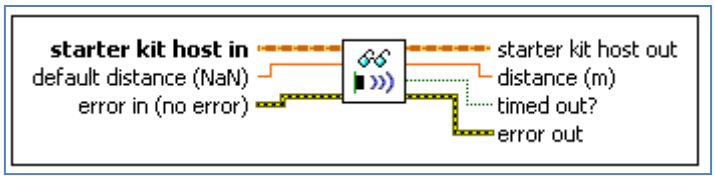


Figure IV.6: Le SubVI*: Read PING Sensor Distance.*

IV.3.1.5. Le SubVI ***Write DC Motor Velocity:***

C'est l'un des composant de la bibliothèque du module LabVIEW Robotics starter kit et qu'on peut retrouver dans la palette *Function>> Robotics>>Write DC Motor Velocity.*

Son rôle est d'appliquer les valeurs de vitesse aux drivers des moteurs sur le robot Starter Kit. Les moteurs gauche et droit sont définis par leurs positions lorsqu'on examine le robot par derrière. La vitesse maximale du moteur est de 15,7 rad / s.

Utilisez ce VI avec les VIs de direction pour convertir entre la vitesse globale du cadre de direction et les vitesses individuelles du moteur et pour mettre en œuvre d'autres comportements de direction pour un robot Starter Kit.

'*Left et Right motor ccw velocity'* spécifie la vitesse en radians par seconde (rad /s), à laquelle on souhaite que lesdeux moteurs se déplacent. Aux vitesses spécifiées, les roues tournent dans le sens inverse des aiguilles d'une montre. Les valeurs positives déplacent le robot vers l'avant et les valeurs négatives déplacent le robot vers l'arrière.

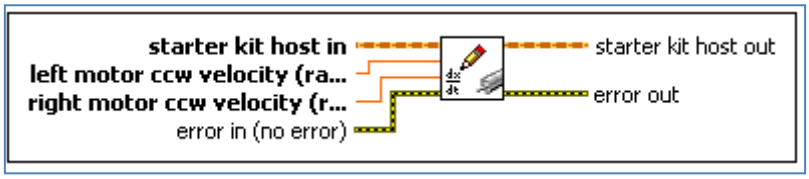


Figure IV.7: Le SubVI *Write DC motor velocity*.

IV.3.1.6. Le SubVI ***Apply Velocity to Motors:***

C'est l'un des composant de la bibliothèque du module LabVIEW Robotics et qu'on peut retrouver dans la palette *Steering VIs >>Apply Velocity to Motors.*

La vitesse centrale du robot en vitesses angulaires de la roue (Calcule la vitesse et l'angle de chaque moteur nécessaire pour satisfaire la vitesse du cadre de direction).

Calcule la vitesse et l'angle de chaque moteur nécessaire pour satisfaire la vitesse de la plateforme de direction ou la vitesse de l'arc au centre d'une plateforme de direction. On doit sélectionner manuellement l'instance polymorphe à utiliser.

Sur ce SubVI on peut choisir entre deux instances polymorphes: Frame et Arc. Le Frame est la bonne instance pour DaNI 2.0. L'entrée de vitesse à trois valeurs sont :

X_dot = vitesse latérale qui est 0 pour un robot à entraînement différentiel comme DaNI 2.0

Y_dot = vitesse avant

Theta_dot = vitesse angulaire

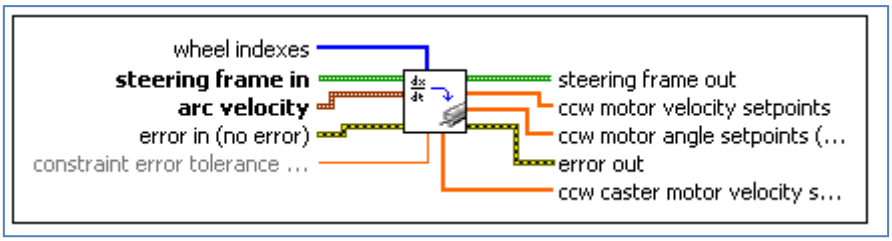


Figure IV.8: Apply velocity to motor.

IV.3.1.7.Le Sub*VI* ***Close Starter Kit :***

C'est l'un des composant de la bibliothèque du module LabVIEW Robotics starter kit et qu'on peut retrouver dans la palette *Function>> Robotics>>Close Starter Kit.*

Ce composant est utilisé pour arrêt l'exécution de programmeet fermer les références pour mettre la mémoire à la disposition d'autres programmes et pour empêcher les fuites de mémoire qui accumulent la mémoire réservée.

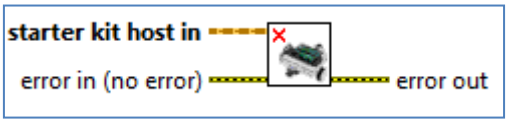


Figure IV.9: Close starter kit.

IV.3.2.1. Programme de la zone scannée :

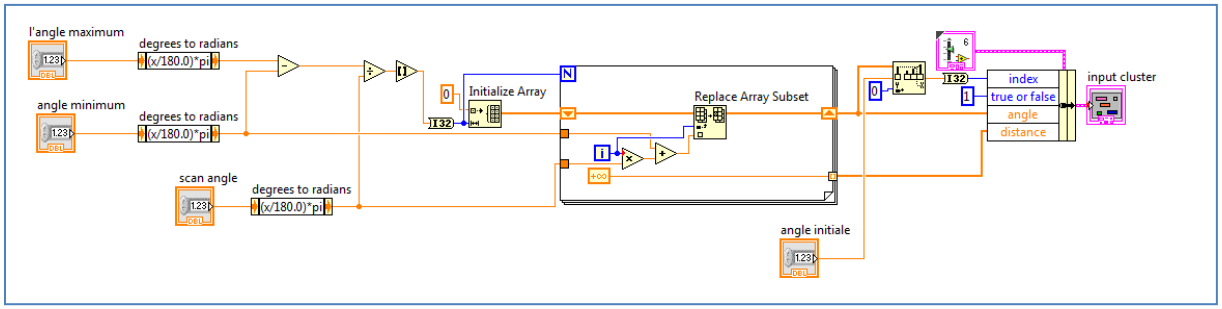


Figure IV.10 : La surface scanne.

Le programme ou le VI relatif à 'La zone scannée' illustré sur la **Figure IV.10** crée une structure de données pour l'angle et la distance sous forme de deux vecteurs de même taille définit par l'angle maximum moins l'angle minimum divisé par le « scan angle » en valeur absolue relie au nombre d'itération N de boucle For, et la fonction '*Initialize Array*' qui crée le vecteur angle remplie dans la boucle « For »en radians par la fonction *Replace Array Subset* et *shift register* (résident sur les bordures de la boucle sous forme d'un triangle. Il donne l'accès pour écrire et lire dans la mémoire. Son fonctionnement est similaire à celui d'un tunnel, il relie la droit de la boucle à sa gauche ce qui permet de récupérer les donnée et de les mémoriser à chaque exécution de la boucle. Toutes les cases du vecteur de distance sont par défaut mises à la valeur +∞.

Ce VI détermine l'index du l'angle initial 0 par la fonction *Threshold*, et détermine aussi les variables booléenne *true or false* qui définit le sens de rotation de servomoteur.

IV.3.2.2.Le programme de l'Angle de rotation

Le programme ou le VI '*angle de rotation'* est illustré par la **figure IV.11.** Il s'exécute itérativement au sein d'une boucle. Il utilise la structure « *In Place Element* » pour donner au servomoteur, et à chaque exécution, l'angle de rotation, par la fonction « *Index Array* » qui fournit à sa sortie le contenu d'une case sélectionne par l'autre index.

La Structure « *Case* »change la valeur de l'index pour sélectionner l'angle suivant et change aussi la valeur de sa condition 'true or false' si la valeur de l'index est à la limité de vecteur, dans le cas où la valeur 'true or false' est « un », l'index s'incrémente et si la taille de vecteur angle, qui est déterminée par « *Arrya size* » est inférieure à l'index, la valeur 'true or false'

reste à « un », et si elle est supérieure à l'index la valeur 'true or false' devient égale à zéro. **Figure VI.12.**

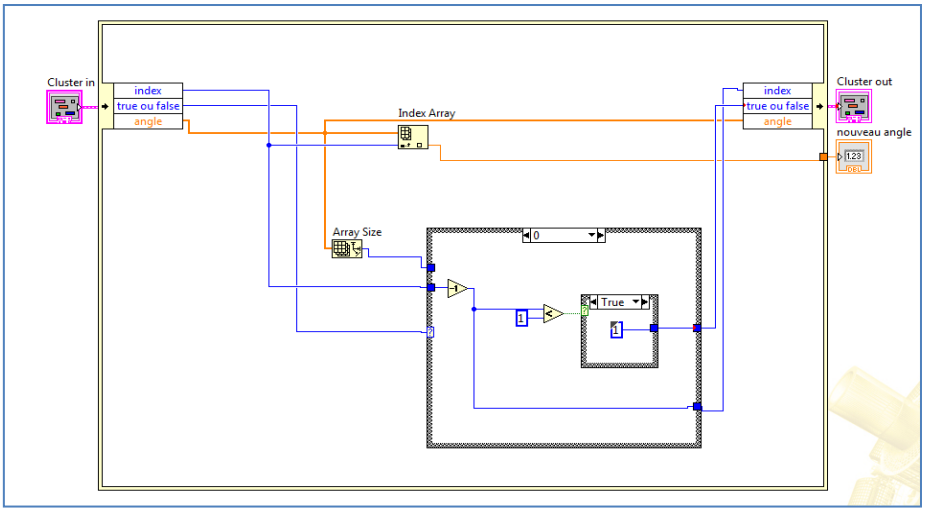
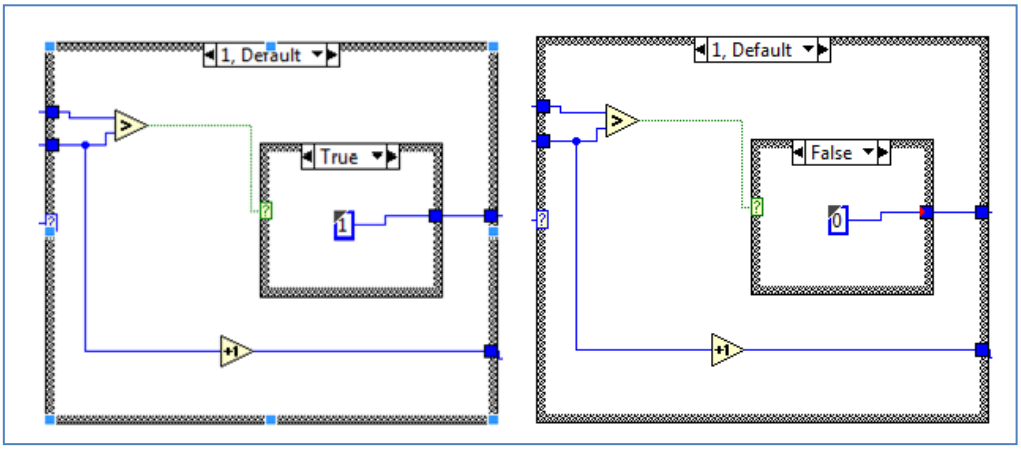


Figure IV.11 : Programme de l'angle de rotation.

Dans le cas où la valeur 'true or false' est à zéro l'index décrémente sa valeur et si la valeur de l'index supérieur à zéro 'true or false' prendre la valeur zéro, si la valeur de l'index égale à zéro la valeur 'true or false' devient « un » et l'opération recommencé indéfiniment. **Figure VI.12**



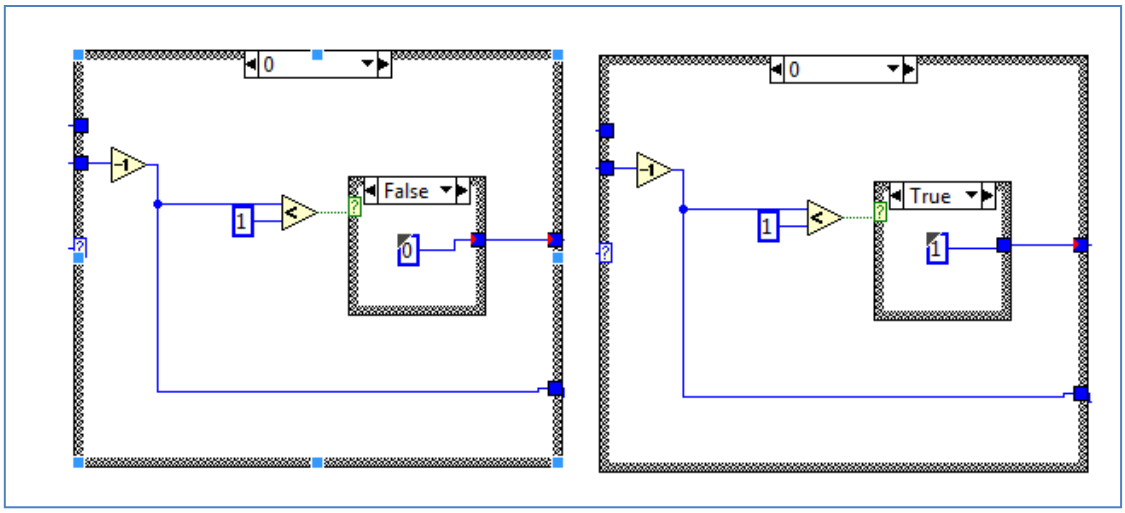


Figure IV.12 : Structure « Case » dans le programme : Angle de roation.

IV.3.2.3.Programme de remplissage du vecteur distance

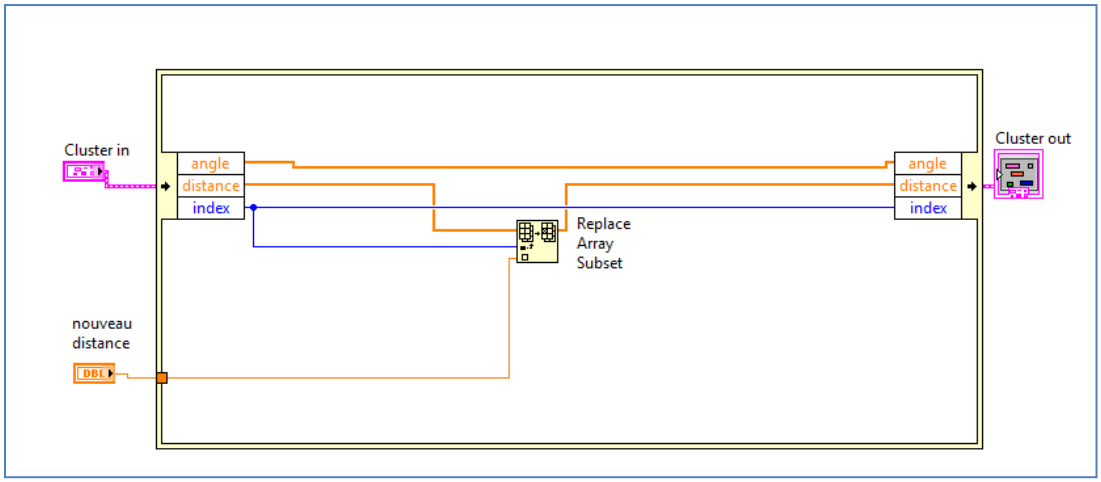


Figure IV.13 : Remplissage du vecteur dedistance.

Ce programme récupère la valeur de distance par le biais de la fonction réalisée par'*Read PING Sensor Distance*' pour chaque angle de rotation. Il écrit cette valeur dans la case du vecteur distance.Cette case a le même index avec la case d'angle dans le vecteur angle par l'utilisation de la fonction '*Replace Array Subset'* **Figure VI.13**.

IV.3.2.4.Programme de détermination de la direction et de la vitesse

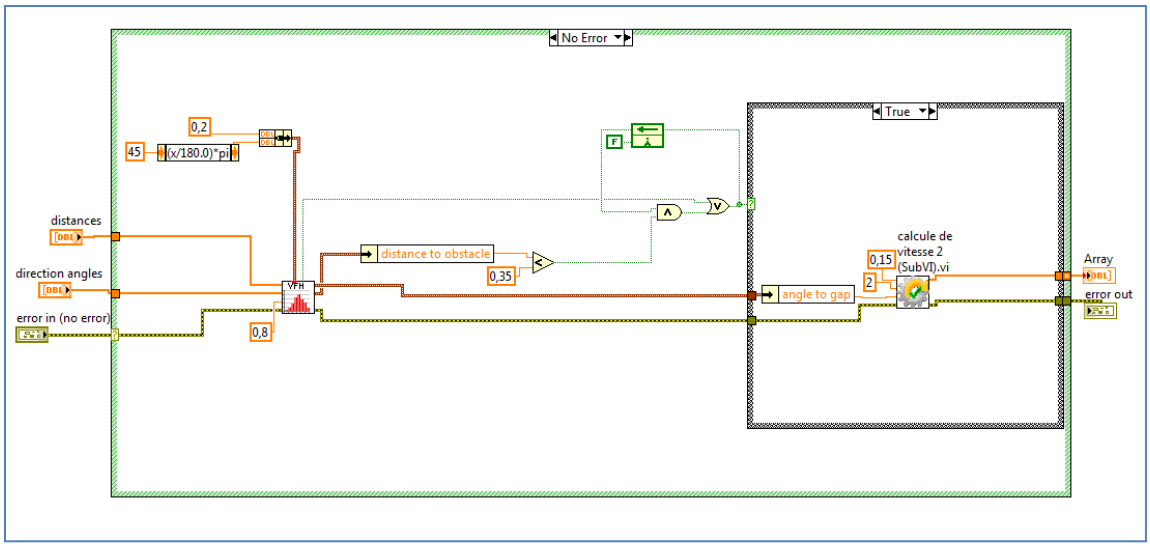


Figure IV.14 : Détermination de la direction et de la vitesse.

Comme entrées, ce programme accepte deux vecteurs : vecteur de distance et d'angle, utilisant une hiérarchie de sous VI, comme le montre la **Figure IV.14**.

Et en utilisant une hiérarchie de sous-VI, il détermine les lacunes, calcule les interstices, calcule la direction de l'écart le plus important et Vitesse du cadre de direction.

'La direction et la vitesse VI' appelle le '*Vector Field Histogram* (VFH) VI' trouvé dans (*Functions >> Robotics >> obstacle avoidance palette*) ces connexions sont présentées sur la **figure VI.15** qui identifie les obstacles et les gaps, ou les zones ouvertes, dans l'environnement du robot.

Le programme de calcul de vitesse et de direction appelle le VI d'histogramme de champ vectoriel simple (VFH) (*Functions >> Robotics >> obstacle avoidance palette*) dont les connexions sont indiquées dans la Figure IV.15 qui identifie les obstacles et les espaces ou les zones ouvertes dans l'environnement du robot. Le VFH VI produit les données d'écart les plus importantes, ce qui est un groupe de deux valeurs, l'emplacement de l'écart et la taille de l'écart. Le plus grand écart décrit la plus grande zone ouverte dans l'environnement du robot. L'angle à l'espace contient la direction de la zone ouverte par rapport au capteur du robot. La taille de l'espace contient la taille angulaire, ou le diamètre visuel, de la zone ouverte, mesurée comme un angle.

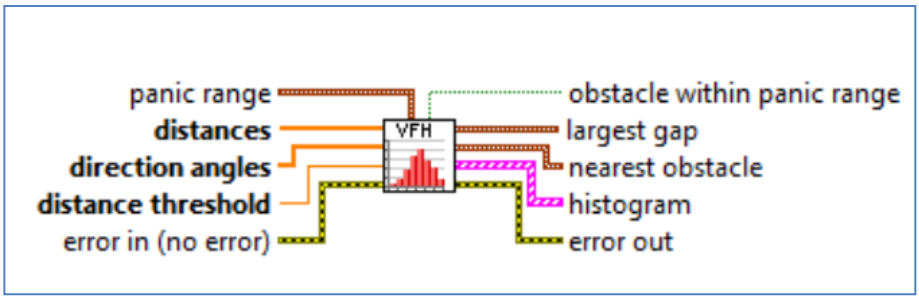


Figure IV.15 : Simple vector field histogram.

Le VFH VI détermine également s'il existe un obstacle dans la zone de panique. La plage de panique est une zone de l'environnement à éviter, c'est-à-dire que l'obstacle est trop proche pour continuer à avancer et éviter. Le code VI itinérant commande à DaNI de se retirer s'il existe des obstacles dans la plage de panique. La structure de données de la zone de panique est un cluster de deux éléments. La distance de seuil de panique spécifie la distance maximale à laquelle l'obstacle dans la plage de panique est TRUE. L'angle de seuil de panique spécifie l'angle maximal auquel l'obstacle dans la plage de panique est TRUE. L'obstacle dans la plage de panique est TRUE si un objet dans l'angle de seuil de panique est plus proche du capteur que la distance de panique.

Les entrées des angles de distance et de direction sont obtenues à partir de la structure de données des obstacles scannés. Les distances sont un éventail de données de portée en obstacles. Les angles de direction sont un ensemble d'angles par rapport au centre du capteur correspondant à chacune des distances. Les valeurs positives des angles représentent des emplacements à droite du centre du capteur, et les valeurs négatives représentent des positions à gauche du centre du capteur.

L'entrée de seuil de distance définit des obstacles. Ce VI ignore tout objet à des distances supérieures au seuil de distance.

La sortie d'obstacle la plus proche est une structure de données composée d'un groupe de deux éléments. L'angle d'obstacle contient la direction de l'obstacle le plus proche par rapport au capteur. La distance à l'obstacle contient la distance entre le capteur et l'obstacle le plus proche.

Panic range distance et *angle* est 0.2 m et 45° et la valeur de*distance threshold* est 0,8 m.

La sortie de la VFH est envoyée au ‘Calcule de vitesse’ comme le montre la **Figure IV.14**. si l’obstacle à l’interieur de la zone *threshold* ou *panic* le‘Calcule de vitesse’ execute le code qui

est illustré à la **Figure VI.16**. qui donnée a sa sortie un vecteur des trois valeurs commande la vitesse et rotation de robot :

$$X_{dot} = 0,$$

$$Y\_dot = -la\ vitesse\ maximum\ d'avancement.$$

$$theta\_dot = \frac{2*angle\ to\ gap*la\ vitesse\ maximum\ de\ rotation}{\pi}.$$

La vitesse maximum d'avancement est 0.15 m/s.

La vitesse maximum de rotation est 2 rad/s.

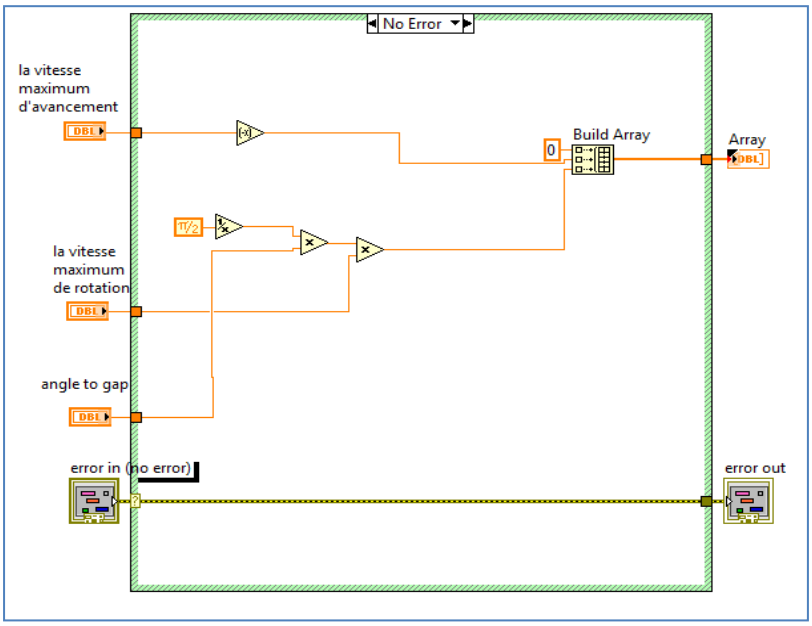


Figure IV.15 : Programme de calcul de vitesse 2.

S'il n'ya aucun obstacle a éviter par le robot 'calcule de vitesse' exécute le code qui est illustré à la Figure VI.16. qui donne à sa sortie un vecteur de trois valeurs commande lvitesse et rotation de robot :

$$X\_dot = 0.$$

$$Y\_dot = \frac{(2*|angle\ to\ gap|+1)*la\ vitesse\ maximum\ d'avencement}{\pi}.$$

$$theta\_dot = \frac{(2*angle\ to\ gap)*la\ vitesse\ maximum\ de\ rotation}{\pi}.$$

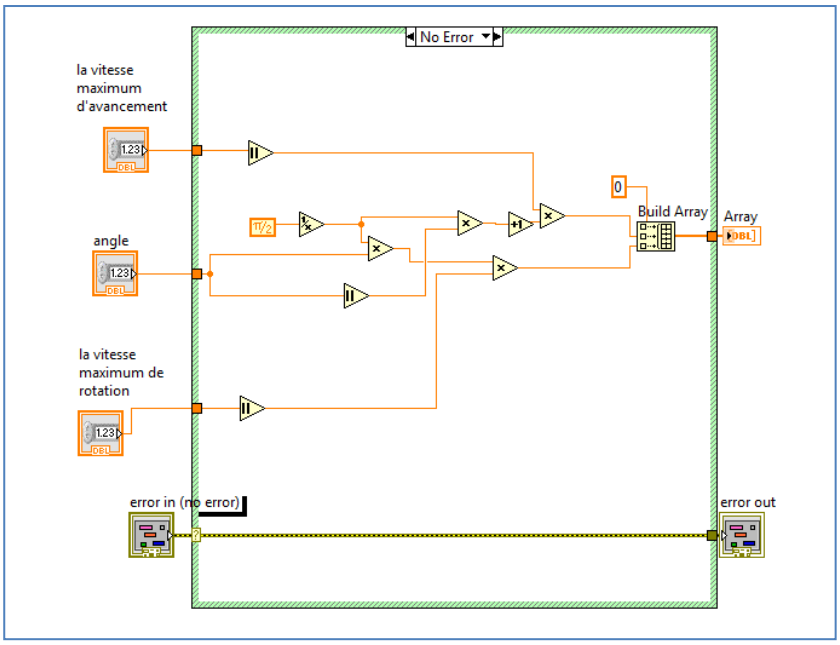


Figure IV.16 : Programme de calcul de vitesse 1.

IV.3.3.1.Tracée du graphe d'obstacle

Sur le « *Front Panel* » du programme principal, un graphe de type XY affiche les obstacles récupère par le capteur ultrasonore comme le montre la **Figure IV.17**.

Pour atteindre ce résultat, on a convertie les deux vecteurs distance et angle de la forme polaire à la forme rectangulaire par la fonction « *Polar To Re/Im* » et concaténé les résulte à l'aide de la fonction « *Build Arry* ».

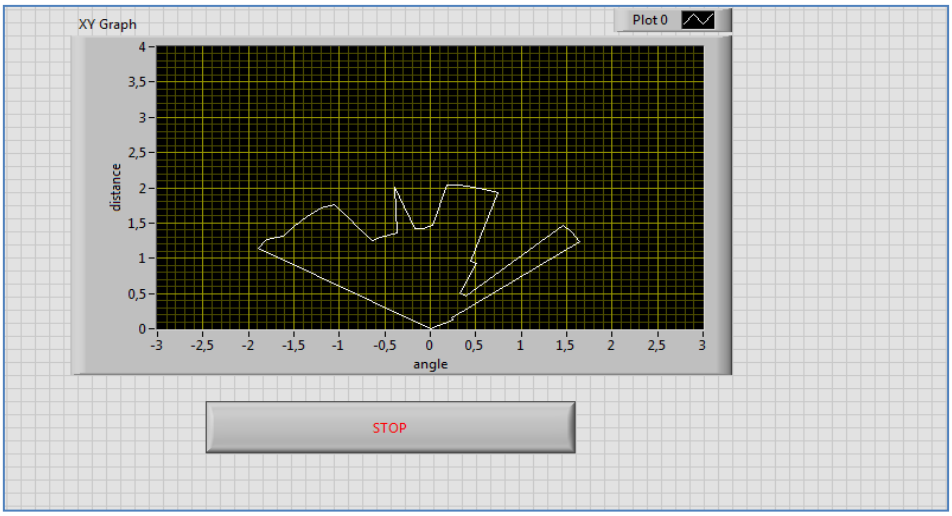


Figure IV.17 : Graphe d'obstacle sur le « Front Pannel ».

IV.4.Conclusion

Lors de ce chapitre, on a essayé d'expliquer le fonctionnement programme qui guide le robot DaNi 2.0 pour éviter les obstacles présents dans son domaine de navigation. Ceci a été fait en expliquant le rôle et la portée de chaque SubVI à part et puis à travers l'interaction de ces SubVIs pour aboutir à la fonction globale recherchée.

Conclusion

Générale

CONCLUSION GENERALE

A travers le travail décrit dans ce mémoire, nous avons pu contrôler et téléguider le robot mobile « DaNI » 2.0 de National Instruments à évites les obstacles dans un environnement inconnu en utilisant le langage de programmation graphique LabVIEW. Pour arriver à cette fin, nous avons utilisé les signaux de mesure issus des capteurs optiques de vitesse/position montés sur les axes de rotation des moteur du robot ainsi que le signal délivré par le détecteur d'obstacle PING))). Ces capteurs sont reliés à la carte embarquée montée sur le robot et qui assure le transfert de données depuis et vers le robot. A l'issue de ce travail, nous avons obtenus de bons résultats, et les tests de laboratoires sont prometteurs et montrent que le robot est téléguidé avec succès et arrive à éviter presque 80% des obstacles mis exprès dans son environnement. Nous signalons ici, le problème de détection des obstacles fins tels que les pieds des chaises et des tables que le détecteur ultrason a parfois du mal à identifier. Comme remède à cette tare, et en perspective d'amélioration de l'efficacité de détection d'obstacles on préconise l'utilisation des capteurs infrarouge ou caméras montés en stéréo-vision. Lors de la phase des tests, on a également remarqué qu'après quelques mètres de distance entre l'ordinateur du bureau et le robot DaNI 2.0 on perd complètement la communication entre les deux entités. Chose qui est essentiellement due à l'utilisation d'une connexion wifi Ethernet dont la portée n'est pas aussi importante que ça.

Au cours des phases de réalisation de ce travail, on s'est rendu compte qu'un aboutissement sure de cette tâche exige la maîtrise de plusieurs notions techniques aussi bien dans le domaine de l'électronique que celui de l'informatique, et c'est ce que nous avons eu l'occasion à découvrir et d'apprendre.

Aussi, ayant travaillé dans un laboratoire de recherche, nous avons également pu acquérir un esprit de travail d'équipe. Chose qui est très utile pour la compréhension et l'analyse des phénomènes physique et la résolution des problèmes techniques qui n'en finit pas de surgir tout au long des phases du travail à réaliser.

Bibliographie

BIBLIOGRAPHE

[1] BERNARD BAYLE « robot mobile » université de Strasbourg.

[2] BOUFERA FATMA «contribution des outils de l'intelligence artificielle dans la robotique mobile » université d'Oran, novembre 2014.

[3] C, Cappelle, El Badaoui El Najjar, M., Pomorski, D., Charpillet, F., "Détection, suivi Et géolocalisation d'obstacles à l'aide d'un modèle 3D géo-référencé, une caméra et Un GPS : validation avec un lidar", Conférence Internationale Francophone d'Automatique, CIFA'08, Bucarest, Roumanie, 3-5 Septembre, 2008.

[4] Davide FILLAT. « Robot Mobile » Ecole nationale supérieure de technique avancées Parise Tech, octobre 2016.

[5] F. LARGE. Navigation autonome d'un robot mobile en environnement dynamique et incertain. Thèse de doctorat, Institut National de recherche en informatique et en automatique, Université de Savoie, 2003.

[6] Lewin A.R.W. Edwards, *"Open-Source Robotics and Process Control Cookbook: Designing and Building Robust, Dependable Real-Time Systems",* 2005, Elsevier Inc, ISBN: 0-7506-7778-3.

[7] MAHER KAFEL, « Localisation et commande d'un robot mobile autonome », mémoire université Québec, novembre 1991.

[8] R. SIEGWART, R. NOURBAKHSH. "Introduction to autonomous mobile robots". The MIT Press, 2004. 155.

[9] RICHARD CLOUTER R « Conception électronique et informatique d'un robot mobile pour usage dans un environnement domiciliaire », Université de sherbrooke canada Faculté de génie électrique et génie informatique, 2007.

[10] ROBERT H.BISHOP «Labview 2009» National Instruments Université de texas, December 28th 2009, 752 p, ISBN 9780132141291.

[11] Robin R. Murphy, « *Introduction to AI Robotics* », (2000) The Massachusetts Institute of Technology Press, ISBN 0-262-13383-0.

[12] S. LENS. Locomotion d'un robot mobile. Thèse de doctorat, Faculté des Sciences Appliquées, Université de Liege, 2008.

[13] STEPHANE LENS « locomotion d'un robot mobiles» université de liège, mai 2008

[14] Thomas Bräunl, "*EMBEDDED ROBOTICS: Mobile Robot Design and Applications with Embedded Systems"*, Second Edition, Springer 1998.

[15] Kansas State University «Robotics Programming Study Guide» Disponible sur http://faculty.salina.k-state.edu/tim/robotics_sg/.