Projet CYCLOPE

Rapport préliminaire

Compagnie RoboVision (B)

Équipe Imagerie

LAROUCHE Eric
TREMBLAY Etienne

Équipe Interface usager – Déplacement intelligent

BERTRAND François
ST-ANDRÉ Jonathan

Équipe Communication

(Division Communication série- Communication sans fils)
BORNE Jonathan
TURCOTTE Jean-François

(Division Gestion du robot)
TREMBLAY Olivier

Équipe Électro-aimant

BOURGEOIS Alain

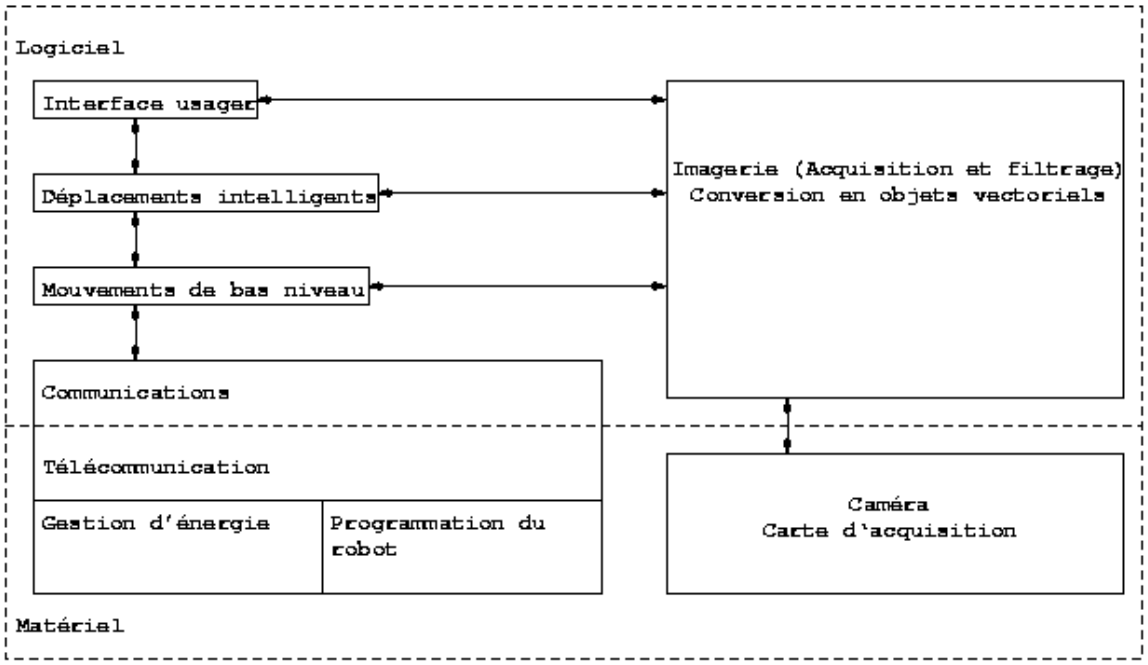
Équipe Gestion de l'énergie

BERGERON Maxime

02 novembre 1999

Description de l'aspect général du projet

Le but du projet Cyclope est de concevoir un système capable de diriger un robot (conçu par une équipe du MIT) dans un environnement où il y a des obstacles à éviter et de petits objets à saisir. Pour se diriger, le système se sert soit d'images captées à l'aide d'une caméra, soit de capteurs. On doit pouvoir commander le robot par un moyen de communication sans fil. Pour contrôler le robot, l'usager peut se connecter au système à partir de n'importe quel ordinateur sur Internet. Par un lien TCP/IP, on accède donc au serveur qui supporte l'application Java dans laquelle l'usager peut donner des commandes au robot. Le robot doit en tout temps connaître son autonomie restante (en énergie) et aller se recharger lorsque nécessaire. Il est également prévu, si le temps le permet, que le système soit en mesure d'éviter un obstacle en mouvement dans l'environnement du robot. La figure qui suit vous donne une excellente aperçue des interaction entre les différentes sections dans le système.



Dans les pages qui suivront, chacune des équipes de développement traitera des tâches impliquées par la sous-section sur laquelle elle travaille, des problèmes rencontrés lors des deux premiers mois d'activité ainsi que des tâches effectuées jusqu'à maintenant et de celles qui restent à faire.

L’équipe Imagerie

- **Importance de la section imagerie**
- **Le lien avec les autres sections**
- **Principale tâches à exécuter en imagerie**
- **Les sous-sections**
  - **Acquisition**
  - **Filtrage et seuillage**
  - **Recherche des objets**
  - **Table de correspondance (image & plancher)**
- **Ce qui a été fait**
- **Ce qui reste à faire**

Importance de la section imagerie

La section Imagerie du projet RobotVision est cruciale pour le succès de notre produit parce que c’est à partir des informations fournies par celle-ci que tout le reste du système fonctionnera. Nous en sommes conscients et nous avons mis les efforts pour répondre aux attentes et même les dépasser.

Ainsi, il importe que cette partie soit la plus précise possible afin de bien guider notre robot dans son environnement. De plus, particulièrement en raison du fait que la caractéristique spéciale de notre système réside en la possibilité d’éviter un obstacle en mouvement, l’aspect rapidité de traitement de l’image revêt une importance capitale. En effet, afin d’être en mesure de bien détecter l’obstacle en mouvement, on devra supporter un taux élevé de rafraîchissement des informations provenant de l’imagerie. Nous avons donc dû développer les algorithmes utilisés dans cette section en se souciant au plus haut point de la vitesse des opérations.

Le lien avec les autres sections

La section imagerie interfacera avec les sections interface usager, mouvement intelligent et mouvement de bas niveau. Ces couches pourront utiliser une fonction de l’imagerie pour obtenir les coordonnées vectorielles des objets sur le plancher.

Principales tâches à exécuter en imagerie

- Acquérir une image à l’aide de la librairie MIL Lite 5.1 de Matrox.
- Filtrer cette image pour en extraire les côtés des objets (obstacles et objets à ramasser).
- Transformer cette image filtrée en image de deux couleurs (noir & blanc).
- Rechercher les objets dans la zone utile de l’image
- Trouver les coordonnées images des sommets des objets.

- Placer celles-ci dans une structure de données.
- Construire une table de correspondance entre certains points du plancher et de l'image.
- Convertir les points images de la structure de données en coordonnée plancher à l'aide de la table de correspondance.
- Fournir une liste des points décrivant chaque objet aux autres sections.

Les sous-sections

Acquisition

L'acquisition de l'image est réalisée à l'aide de la librairie Mil Lite 5.1 de Matrox. On doit utiliser des fonctions d'allocation pour allouer de l'espace pour un système, une application et un « buffer » pour emmagasiner l'image. On doit également acquérir l'image avec la fonction « Mgrab » de la librairie pour capturer l'image. On doit ensuite désallouer la mémoire.

En soit, cette partie du travail n'est pas très difficile. En effet, la documentation disponible en format PDF était assez claire et les fonctions peu difficiles à utiliser. Par contre, nous avons passé plus de temps que prévu sur cette partie en raison des problèmes rencontrés avec le compilateur. Le compilateur utilisé est Microsoft Visual C++. Nous avons eu des problèmes lors du « linkage » parce que nous n'avions pas fixés les bons paramètres internes (.lib et .dll). De plus, les fonctions d'affichage de la librairie Matrox ne sont pas utilisables en mode débug. Ceci nous a causé certains problèmes au début du projet.

Filtrage et seuillage

Afin de filtrer l'image pour faire ressortir les arêtes des objets, nous avons envisagé principalement deux alternatives. Tout d'abord, nous avons parcouru plusieurs sites sur Internet concernant le traitement d'images. Nous avions comme objectif de trouver du code source qui effectuait le traitement recherché. Nous avons trouvé certaines librairies de traitement d'images en langage C++. Nous les avons brièvement analysées, mais elles n'étaient pas directement utilisables dans notre projet.

En parallèle, nous avons consulté certains livres traitant du traitement d'images. Nous avons constaté que développer nos propres fonctions de traitement d'images n'était pas vraiment difficile. Nous avons donc développé une fonction qui effectuait la convolution d'une matrice filtre 3x3 avec une matrice image (640x480). Par la suite, nous avons dû trouver quel filtre nous allions utiliser pour faire ressortir les arêtes des objets dans l'image. Notre choix s'est arrêté sur le filtre de Sobel. Ce type de filtre permet faire ressortir les arêtes dans l'image tout en éliminant la plupart du bruit présent dans l'image. Il combine le résultat de deux autres filtres qui servent à calculer le gradient. Cette opération est suivi d'un seuillage pour transformer l'image contenant jusque là 256 tons de gris en une image noir et blanc.

Dans cette section, nous avons eu certains problèmes avec l'allocation de tableaux 2D de grandes dimensions. En effet, lorsque l'on déclare plusieurs tableaux de 640x480, le compilateur indique

qu'il y a un « stack overflow ». La première solution fut d'augmenter la taille de la pile. Toutefois, en plus d'être très peu élégante cette alternative causait une exception de type « access violation ». Nous avons donc envisagé d'allouer préalablement la mémoire à l'aide de l'opérateur « malloc » du langage C ou de l'opérateur « new » du langage C++. La solution retenue a été l'opérateur « new » en raison de sa simplicité d'utilisation pour la déclaration de tableaux 2D. Nous avons aussi eu quelques problèmes avec le passage de tableaux 2D en paramètre à des fonctions.

Il est important de mentionner que la plupart de ces problèmes découlent du fait que notre compagnie s'est donné comme objectif d'utiliser, si possible, uniquement le langage C pour faciliter l'intégration de tous les modules ensemble. Habitués d'utiliser des classes implantant des structures de données toutes faites, nous avons donc dû approfondir nos connaissances dans ce domaine afin d'éviter d'utiliser des objets de classes extérieures. Il est à noter que même si l'opérateur « new » est un élément du langage C++, il a été déterminé que l'intégration aux autres modules n'en serait pas affectée.

Recherche des objets

A partir de l'image noir et blanc obtenue de l'étape de filtrage et de seuillage, nous avons à détecter les objets dans la zone utile de l'image. A l'heure actuelle, les algorithmes pour effectuer cette tâche ne sont pas encore tout à fait fixés. Ils le seront dans les jours qui suivent car il est prévu que ce module sera implanté au début de la semaine du 8 novembre.

Table de correspondance (image & plancher)

Pour effectuer la correspondance entre notre image et le plan plancher, nous avons envisagé principalement trois alternatives. Tout d'abord, nous avons évalué le cas limite qui consiste à effectuer la correspondance entre tous les points de l'image et ceux du plancher. On s'est vite rendu compte que cette avenue était irréalisable. La seconde option fut celle de calculer la correspondance entre le plancher et l'image à partir de l'angle de la caméra et des distances entre le plan caméra et le plan plancher. Nous avons écarté cette avenue parce que nos connaissances en vision numérique sur la transformation de coordonnées n'était pas assez grandes. De plus, cette option demandait de mesurer avec précision l'angle de la caméra et les distances. L'option que nous avons adoptée est en quelque sorte une modification de la première méthode. Nous avons décidé d'effectuer la correspondance de seulement quelques points (environ 25). Lorsqu'un point cherché ne sera pas sur l'un des points connus de la table de correspondance, nous utiliserons une fonction d'interpolation pour calculer la correspondance. Le principal risque avec cette méthode est du côté de la précision dans nos mesures. Ainsi, nous devrons être très minutieux lorsque nous exécuterons celles-ci.

Ce qui est fait

L’acquisition de l’image a été effectuée. Le filtrage et le seuillage ont été réalisés. L’implantation de la fonction d’interpolation a été faite. Les algorithmes pour la détection sont présentement analysés et une décision sera prise sous peu.

Ce qui reste à faire

Effectuer la correspondance physique entre le plan image et le plan plancher en mesurant certains points dans la salle où le robot naviguera. Effectuer le choix des algorithmes pour la détection des objets et les implanter. Vérifier que le tout fonctionne adéquatement sur les deux postes NT du laboratoire avec des objets de différentes tailles. Organiser les fonctions de sorte qu’elles retournent une liste des points décrivant chaque objet. Finalement, effectuer l’intégration avec les autre départements de la compagnie.

L’équipe Communication (Gestion du robot)

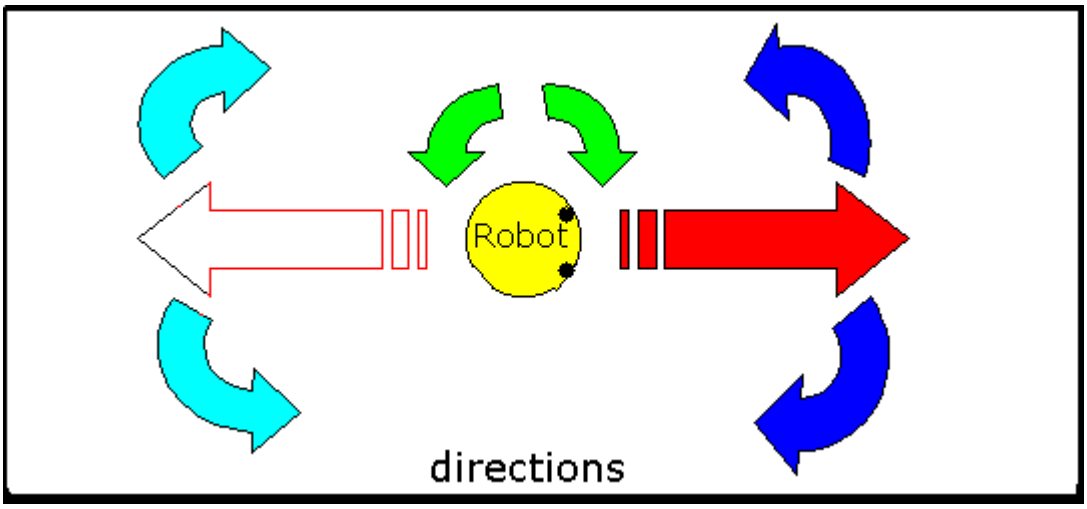
Objectif de cette section

Le robot doit répondre à des instructions obtenues à partir du PC. Il doit donc posséder un jeu d'instructions qui permettra au PC de le contrôler le plus simplement possible facilitant ainsi le travail des mouvements intelligents et de la communication sans fil en ayant le moins de paramètres possibles à passer. Le robot doit aussi être capable d'assister l'imagerie dans la détection des obstacles, et dans la fluidité et la linéarité de ses déplacements. (Voir les figures ci-jointes)

Les fonctions suivantes existent pour contrôler les mouvements du robot à partir du PC (voir la figure direction):

Fonctions de direction

-robot_avance(vitesse);
-robot_recule(vitesse);
-robot_gauche(angle);
-robot_droite(angle)
-robot_arrete();



Fonctions du bras magnétique

-robot_echappe();
-robot_ramasse();

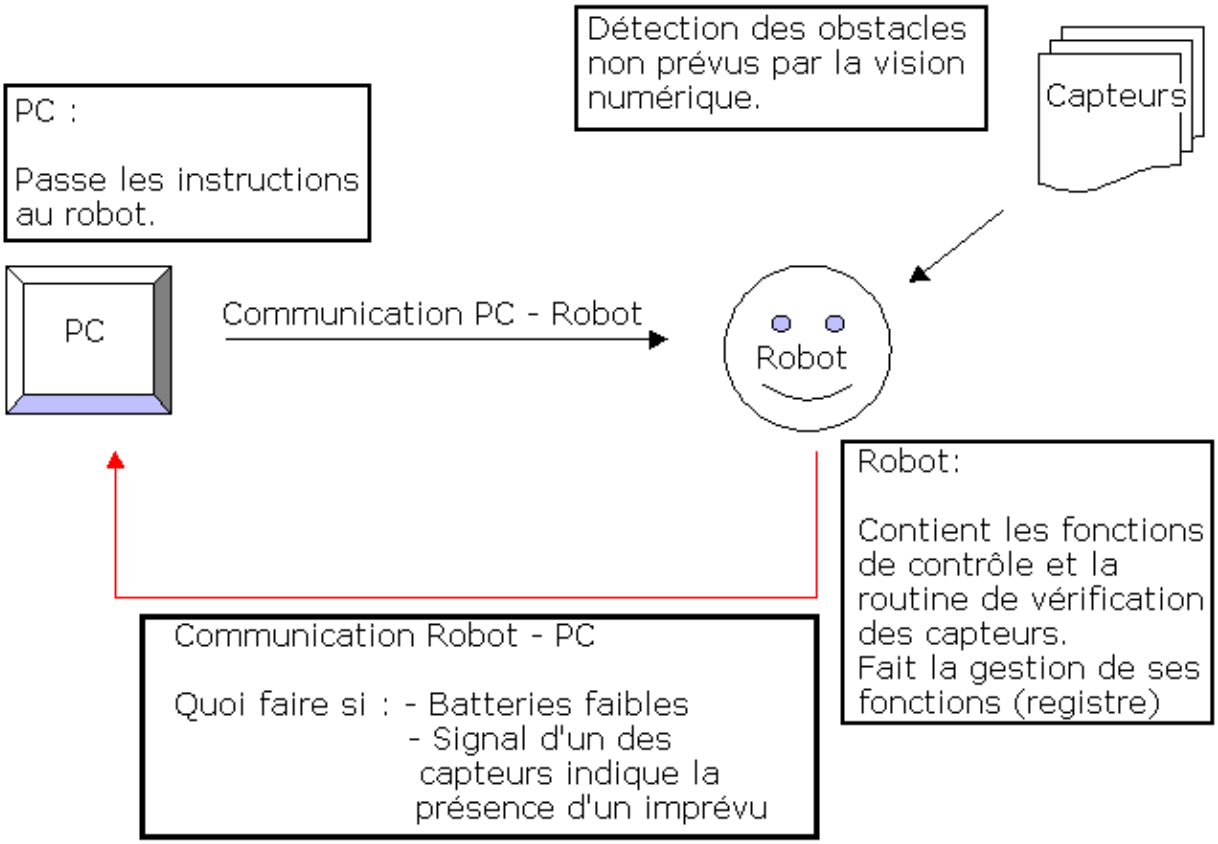
D'autres fonctions sont utilisées à l'interne pour le contrôle des capteurs. Dans le cas des

fonctions pour les capteurs, celles-ci passent le message au PC si elles détectent une perturbation (si les tests se voient positifs).

Fonctions des capteurs

-robot_bumper();
-robot_ir();

Tous les types de fonctions contiennent des «Flags» qui indiquent l'action des moteurs ou de n'importe quel autre élément ou fonction du robot. La fonction principale du robot est celle qui demande les tests à l'interne du robot pour vérifier l'état des capteurs. On peut voir dans la figure ci-dessous un schéma global du système et des interactions entre chacune des parties.



Gestion et contrôle du robot

Ce qui est fait

La plupart des fonctions énumérées ci-haut sont codées, mais non testées sur le robot. Celles qui l'ont été fonctionnent, mais devront être modifiées pour permettre l'asservissement grâce aux « shaft encoders ».

Ce qui reste à faire

- Implanter les algorithmes papier dans le robot.
- Faire les tests de ces fonctions sur le robot.
- Faire le lien avec la fonction qui vérifiera l'état des piles lus.
- Faire le lien avec la routine d'interruption de la communication sans fil.

Principal problème rencontré

Le principal problème que nous avons rencontré est, curieusement, le robot en tant que tel. A notre avis, il n'est pas normal que cela prennent deux heures pour être capable d'importer le code dans le robot. En effet, le logiciel IC ne reconnaît pas le « board » pendant tout ce temps. Ce qui entraîne comme conséquence directe, un retard dans l'échéancier de départ. Nous ne sommes donc pas aussi avancé que prévu dans la vérification et le « débuggage » des algorithmes en raison de cet inconvénient.

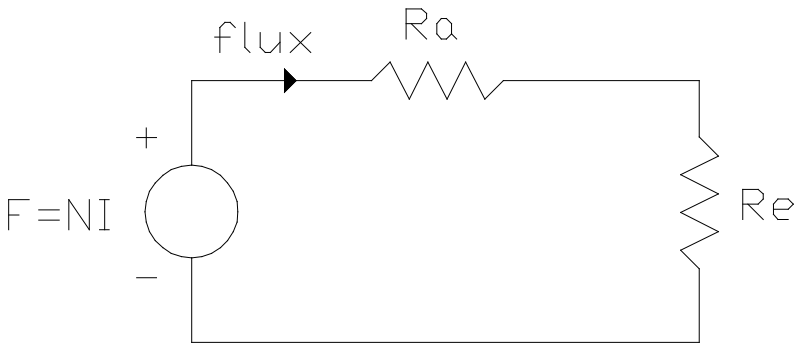
L’équipe Électro-aimant (Système de capture)

Système de capture

- Magnétisation DC
- Magnétisation AC
- Bras mécanique

Magnétisation DC

Circuit magnétique équivalent :



Calculs et mesures

N = 90 tours I = 2.1 A e (entrefer) = 0.02 m l = 0.08 m
A = 2.1^{E} –5 m² u_r = 450 (noyau en acier doux)
R = l / uA (réluctance)

Ra = 0.08 / (450u_0 x 2.1^{E} –5) = 6.74^{E} 6 At/Wb (réluctance de l’acier)
Re = 0.02 / (u_0 x 2.1^{E} –5) = 7.58^{E} 8 At/Wb (réluctance de l’entrefer)

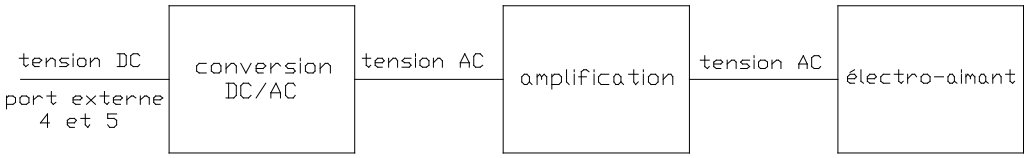
$$\phi = \frac{NI}{Ra + Re} \qquad \phi = \frac{90 \times 2.1}{Ra + Re} = 3.70^{E}\text{-}14 \text{ Wb}$$

$$B = \frac{\phi}{A} = 1.76^{E} \text{–}9 \text{ Tesla}$$

Il faut un courant beaucoup trop élevé pour exciter la bobine lorsque la magnétisation DC est utilisée. Les piles du robot ne nous donnent pas un tel courant.

Magnétisation AC

Schéma de principe :



Les calculs pour les dimensions et les matériaux de l'électro-aimant sont terminés. Il est principalement constitué d'un noyau en acier doux et bobiné avec un fil de cuivre verni. Il reste à concevoir le circuit permettant de convertir la tension DC en tension AC et d'obtenir un courant d'excitation élevé.

Bras mécanique

Le bras mécanique servant à supporter l'électro-aimant sera fait d'aluminium pour rendre le bras le plus léger possible. Un contrepoids sera ajouté à l'arrière du robot pour compenser la masse de l'électro-aimant. Les dimensions et les diverses mesures ont été effectuées. L'installation sera faite sous peu.

L’équipe Gestion de l’énergie

La gestion d’énergie nous permet:

- de connaître la consommation d’énergie.
- d'estimer l’autonomie du robot.
- d'informer le robot lorsque la tension des batteries est basse

La consommation d’énergie nous permet de connaître la puissance nécessaire au robot pour exécuter ses différentes fonctions. Comme par exemple, lorsque le robot avance a 100% de sa vitesse avec une charge.

La fonction ci-dessus est surtout pour l’évaluation de l'autonomie du robot. En observant la puissance a chaque instant, il est possible de calculer la puissance moyenne sur une certaine période. A partir de cette puissance moyenne et de la puissance disponible par la batterie au début de l'utilisation, nous pouvons en déduire l’énergie qui reste dans ses batteries. Le but de tous ces calculs est de prévoir le moment ou la recharge sera nécessaire.

Un autre système sera présent sur le robot, c’est un indicateur de la tension des batteries qui indiquera de manière précise lorsque qu'une charge sera nécessaire. Un signal est envoyé a une des entrées du robot, lorsqu'il y a une baisse de tension sous un seuil critique, prédéterminé a l'avance. Le seuil est calculé pour permettre au robot une certaine autonomie pour se rendre jusqu’à la zone de charge sans tomber en panne!

Ce qui est fait

Le design du détecteur de tension ainsi que le calcul de ses composants pour effectuer un montage dans les prochains jours et exécuter les tests nécessaires.

L’évaluation de différente manière de connaître la consommation d’énergie du robot de manière précise ou par une estimation.

Ce qui reste à faire

La fabrication du système pour connaître et évaluer la consommation d’énergie.

Communication RF

Description du type de communication

Pour la communication entre le robot et l'ordinateur, nous avons choisi la communication AM puisque suffisante pour nos besoins et beaucoup plus simple que la communication dans les GHz. Notre débit de transmission est de 4800 bauds à 433 MHz.

Modules émetteur/récepteur

Pour notre modulation AM, nous avons opté pour des modules fabriqués par la compagnie ontarienne ABACOM Technologie - www.abacom-tech.com - Ce sont des modules émetteur/récepteur AM fonctionnant à une fréquence de 433.92 MHz à un débit de 4800 bauds half-duplex. Ces module supportent les données RS-232 et le circuit de fonctionnement est très simple à implanter.

Protocole de communication

Notre protocole de communication est programmé en C++. Pour pouvoir accéder au port série de Windows NT, nous utilisons la classe CSerial.

L'ordinateur transmet alors quatre caractères. Le premier représente le numéro de la commande désiré, le second est le premier paramètre de la fonction à exécuter, le troisième est le second paramètre de la fonction à exécuter et finalement, le dernier est un caractère de contrôle. Il est à noter que nos modules de communication sont half-duplex ce qui veut dire que la communication ne se fait que dans un seul sens et, par conséquent, le robot ne transmet des information que sur demande de l'ordinateur.

Le robot pour ça part transmet quelques informations tels l'état des batteries, l'états des senseurs infrarouges ainsi que celle des interrupteurs de contact. Il transmet aussi un caractère de contrôle. La routine de réception du robot ce fait en sous-routine du programme principal afin de ne pas avoir à tout arrêter pour établir la communication qui devra être très fréquente étant donné les états des senseurs qui doivent être vérifié sur d'assez courts délais.

Ce qui est fait

Au niveau du protocole de communication, nous avons établit la communication. Nous somme capable de transmettre et de recevoir des donnés tant du côté du robot que du côté de l'ordinateur. Le protocole lui-même est en voie d'être terminé et le système de contrôle de la communication reste à programmer.

Pour ce qui est des modules de communication RF, nous les avons en main depuis une semaine environ. Nous n'avons donc pas encore effectué de test dessus cependant, le circuit est très simple à implanter et ne devrait poser aucun problème.

Ce qui reste à faire

Parmi les principales tâches restantes, il y a bien sûr l'intégration de la partie communication avec les autres parties. Mais pour la partie communication elle-même, il y a quelques petites choses à faire encore. L'intégration du protocole avec les modules RF, la mise au point du contrôle de la communication ainsi que finaliser le protocole de communication.

Principal problème rencontré

Le principal problème rencontré jusqu'à présent et de nature software. En effet, le mélange Windows NT et Visual Devloper Studio a rendu la programmation beaucoup plus difficile qu'elle ne l'aurait dû l'être. De plus, les sécurités de Windows NT nous compliqué l'accès au port série de l'ordinateur.

Interface usager

Objectif

L'interface permettra à un usager de contrôle robot en cliquant avec la souris à divers endroit sur le graphique. L'espace de travail est représenté sur l'environnement graphique ie tous les objets sont représentés. L'interface usager est programmé en JAVA puisque ce langage est bien adapté pour les applications qui demandent du graphisme.

Problèmes

La sécurité dans le langage Java empêche un applet Java qui a été téléchargé à partir d'un serveur web de communiquer avec un ordinateur différent que ce serveur web. Dans ce cas-ci, notre applet Java pourrait être téléchargé à partir du serveur web du département mais ne pourrais pas, ensuite, communiquer avec le poste NT qui contrôle le robot.

Solution

La meilleure solution est d'implémenter notre propre petit serveur web qui sera fait pour ne servir que quelques fichiers. Ce petit serveur web sera un programme séparé de notre programme pour le robot et ne sera nécessaire que dans les cas ou le poste qui contrôle le robot ne possédera pas de serveur web par lui-même.

Ce qui est fait

Le programme du petit serveur web est pratiquement terminé. Il est fonctionnel sur les plates-formes UNIX et LINUX. Il est fonctionnel à 90% sur Windows NT.

Ce qui reste à faire

L'interface usager qui sera programmé en JAVA.

L’équipe du mouvement intelligent

Fonctionnement

À l'aide de points fournis par l'imagerie qui représentent le contour des objets, le programme pourra calculer le chemin que devra emprunter le robot pour se rendre du point A au point B sans heurter d'objets lors de son déplacement.

Problèmes

L'algorithme implanté pour trouver son chemin fonctionne très bien dans le cas où il y a seulement un objet. Par contre, lorsqu'il y a plusieurs objets dans l'espace de travail, l'algorithme réussit à trouver un chemin, mais le robot passe par New York pour faire Québec Montréal.

Solutions

Optimiser l'algorithme de calcul pour déterminer s'il n'y pas de chemin plus court que celui trouver en premier lieu.