ASSEMBLAGE ET CONTRÔLE D'UN BRAS ARTICULÉ AVEC UN GANT «POWER GLOVE®»

PAUL-ÉTIENNE BELLONCIK ET JEAN-FRANÇOIS LACASSE, **COLLÈGE ANDRÉ-GRASSET**

RÉSUMÉ DU PROJET

Notre projet de recherche Dec Plus en physique-informatique représente le montage «d'un bras robotisé» contrôlé à l'aide d'un Power Glove®. Pour ce faire, il faut évidemment utiliser un Power Glove®, un ordinateur de type IBM, un ensemble de robotique Robix RCS-6 et une interface spéciale (interface Menelli) créée pour le Power Glove®. Une première étape de notre projet a été de concevoir le système de transmission et de traduction des données entre le Power Glove® et le robot. Ensuite, plusieurs heures ont été consacrées à la programmation et à la manipulation physique du bras Robix et du gant Power Glove®. Le résultat est un montage compact et fonctionnel dont nous sommes fiers. Afin d'expliquer ce système, la description de l'expérience contient quatre sections: une présentation brève du matériel, la démarche et le fonctionnement de l'appareil d'entrée, la démarche et le fonctionnement de l'appareil de sortie et finalement la fusion et l'interaction de ces deux systèmes au sein même de l'ordinateur.

CONTEXTE THÉORIQUE

La robotique partout

Que ce soit le bras mécanique qui met des disques en vinyle sur le tourne-disque d'un «juke-box», le robot dont se sert la police dans les alertes à la bombe, les machines servant à construire les microprocesseurs couche par couche ou le bras articulé qui peinture les voitures dans la chaîne de montage à l'usine, la robotique est maintenant omniprésente dans notre vie. Dans le but de remplacer des postes à fonctions répétitives et abrutissantes pour les employés, on a souvent recours à l'automatisation. L'automatisation est une technique permettant à une machine ou à un groupe de machines d'effectuer des tâches à répétition sans intervention humaine. Elle vise principalement à réduire les coûts de production et à simplifier les chaînes de montage en série.

Mais il y a des opérations qui requièrent une précision extrême au point qu'il faille reproduire exactement le bras humain, complet ou avec une simple pince, afin de mener à bien la manipulation. De tels bras robotisés sont soit programmés à l'avance avec une séquence de gestes répétitifs ou contrôlés en temps réel, comme dans notre projet.

Notre projet n'a pas besoin d'hypothèse de travail puis qu'il s'agit de la réalisation d'un système informatique. Nous nous sommes demandés s'il était possible de contrôler un bras robotisé avec notre Power Glove® et nous y avons travaillé depuis le tout début de la session. Il n'y a pas eu non plus de cueillette de résultats, car notre projet se concentre davantage sur la réalisation d'un système imaginé plutôt que de la vérification en laboratoire d'une hypothèse scientifique. Malgré ces manques au point de vue de la démarche scientifique, nous remplacerons les sections de l'analyse des données par une analyse de la mécanique des mouvements du robot.

MATÉRIEL UTILISÉ

L'ensemble Robix RCS-6

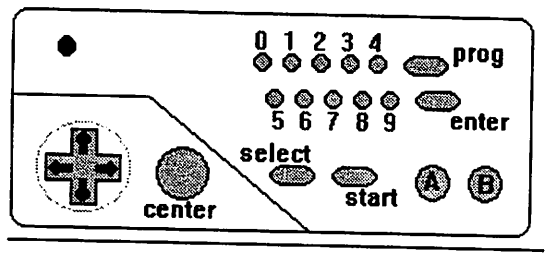
Le bras robotisé de cet ensemble contient six servomoteurs. Ces servomoteurs peuvent être assemblés en utilisant des poutres et des articulations (fournies dans l'ensemble) afin de former différentes constructions robotisées. De multiples attaches et vis assurent la fixation des différents modules. Un programme compatible aux ordinateurs de type PC est fourni sur une disquette. Ce programme, qui permet de faire bouger le robot et de lui apprendre des séquences de mouvements, fonctionne sous le système d'exploitation DOS. C'est avec une variante de ce programme que nous allons contrôler le bras.

Plusieurs choix tels qu'un marcheur, un lanceur de balles de ping-pong et un serpent mécanique étaient proposés dans le manuel de la compagnie. Nous avons choisi le bras robotisé, car il nous offrait plus de possibilités et de flexibilité au point de vue de son utilisation que les autres choix. De toute façon, notre projet devait comprendre un système articulé ressemblant à un bras. Pour la phase préliminaire de notre projet, nous avons opté de suivre à la lettre les instructions de montage du bras fournies dans le manuel Robix. Cela nous a permis d'expérimenter avec le produit ainsi que le programme Robix. Décrit plus tard dans la section de l'analyse des résultats sera le nouveau modèle de bras que nous avons nous-mêmes créé vers la fin du projet.

Le gant analogique Power Glove®

Ce gant, aujourd'hui discontinué, a connu ses heures de gloire au temps de la console Nintendo 8-bit, il y a environ huit ans. Il était utilisé pour des jeux vidéo à la place de la manette de jeu. Les contrôles du gant étaient fixés manuellement ou automatiquement selon le genre de jeu. Sur le bras du gant se trouvent les mêmes boutons que sur la manette de jeu standard, ce qui permettait de revenir à n'importe quel moment à une forme de contrôle plus simple. De plus, sur cette même partie, nous retrouvons dix boutons supplémentaires dont les fonctions nous sont inconnues. Cependant, ces boutons nous serviront grandement plus tard pour notre projet.

Contrôleur sur le bras du gant



L'interface Menelli

Ron Menelli a, en 1991, réussi à réaliser une interface qui se chargeait de faire la traduction entre les données encodées venant du gant et l'ordinateur. Il existe donc ainsi un moyen de connecter un Power Glove® à un ordinateur. Cette interface est connectée au *port série* d'un ordinateur. Les plans de cette interface se re-trouvent sur le domaine public d'Internet.

DESCRIPTION DE L'EXPÉRIENCE

Périphérique d'entrée

Nous commencerons cette section en parlant du gant analogique Power Glove®, fabriqué par Nintendo®. Le fonctionnement du gant est assez simple. À chaque instant, deux émetteurs à ultrasons situés sur le dessus du gant transmettent aux trois capteurs des informations. Ces capteurs sont reliés par deux tiges de plastique et forment un «L» que l'on peut mettre autour d'un moniteur. Ces récepteurs sont responsables de localiser le gant dans les trois dimensions, un peu comme une chauve-souris le ferait pour situer une proie.

En plus des coordonnées X, Y, Z, les récepteurs peuvent détecter les mouvements de roulis en calculant le changement de position des deux sources à ultrasons. Cette différence, s'il y en a une, donnera le degré de rotation du poignet. S'il n'y a aucune différence, les capteurs concluront que le gant est droit (paume vers le bas).

Une autre caractéristique du Power Glove® mesure le degré de fermeture de quatre doigts. Au-dessus de chaque doigt, de minces bandes d'encre conduisent un courant. Plus un doigt est plié, plus la résistance de l'encre augmente et plus le courant diminue. En donnant un code différent au pouce, à l'index, au majeur et à l'annulaire, les capteurs distinguent quel(s) doigt(s) est(sont) plié(s).

Finalement, les récepteurs à ultrasons reçoivent des données sur l'état des boutons. Ces boutons ont aussi leur valeur individuelle.

Donc, le gant transmet six informations, soit: les trois coordonnées spatiales (x,y,z), la rotation du poignet, la fermeture du poing, ainsi que les boutons utilisés, et envoie cette information à la console Nintendo.

La traduction du Menelli

Comme nous ne désirons pas utiliser la console Nintendo®, mais un ordinateur, nous avons fabriqué l'interface Menelli découverte par nos recherches sur Internet.

Le signal transmis par cette interface est non seulement reçu par le PC, mais cohérent quant aux changements de position, rotation, etc., du Power Glove®.

L'interface est composée d'une puce de type MAX232, d'un *micro-contrôleur* Motorola MC68-HC811E2P, de plusieurs résistances de 10 KiloOhms, de quelques condensateurs et d'un cristal de 8 Mégahertz. Le MC68HC fonctionne grâce au programme qui a été stocké dans sa mémoire morte. Ce programme est aussi disponible sur Internet. L'interface fonctionne comme suit: le MC68HC traduit les données pour qu'elles soient compatibles avec le PC. Le MAX232 joue le rôle de convertisseur électrique. Prenant le signal de 5 volts du contrôleur Motorola, il le convertit en signal ±12 volts pour le port série de l'ordinateur.

Cependant, il ne suffit pas de brancher le Menelli en arrière de l'ordinateur pour magiquement lire le Power Glove®! Des données sont certes transmises, mais il faut les traiter. C'est ici que la programmation entre en jeu.

Des données incompréhensibles

Quand les informations du gant arrivent au PC, elles ont la forme d'une suite infinie de caractères du genre: «Ìh*_ f*_Ëj*_Yð Üð_». Cette série de données est évidemment incompréhensible. Il faut donc, tout d'abord, savoir ce que ces caractères représentent. À ce stade-ci, il devient important de bien comprendre le fonctionnement

du Menelli. L'interface, selon la programmation chargée en mémoire, peut opérer en deux modes. Elle peut soit envoyer continuellement les informations du gant (mode continu), ou soit ne rien envoyer jusqu'à ce qu'on lui pose une question (mode requête). Ces deux modes fonctionnent très différemment.

Si nous demandons au microcontrôleur des informations en mode continu, il nous donnera une séquence répétée indéfiniment de six caractères séparée d'une espace. Or, cela correspond exactement aux renseignements envoyés par le gant (x, y, z, poignet, doigts, et boutons)! Ces «paquets» de valeurs, qui défilent continuellement, doivent être séparés par des espaces afin d'en accommoder la lecture. Heureusement c'est ce que fait le Menelli.

En mode requête, la puce nous retourne six valeurs seulement quand nous lui envoyons un signal (le caractère «?»). Autrement, elle reste silencieuse.

Ayant décodé le modèle de communication utilisé, nous pouvons traiter ces informations maintenant devenues intelligibles.

Nous pouvons donc faire fonctionner le matériel, mais il nous reste encore à concevoir un logiciel qui exploitera ces informations! À ce stade-ci, nous avons réalisé une application nous permettant de bien voir ce que nous recevions. Nous avons donc créé une «fenêtre sur l'interface Menelli». Ce programme nous permet de voir exactement quelles valeurs numériques, car c'est le seul langage que le gant puisse nous communiquer, sont envoyées selon la position du gant dans l'espace. Mais avant d'aller plus loin, il fallait bien commencer à faire un bout de programme.

De la programmation pour le Menelli: choisir le bon langage

Avant de commencer à programmer, il faut évidemment choisir le langage idéal correspondant à nos besoins: lire les informations du port série, où est connecté le Menelli. Un langage simple devait suffire. Le Quick-Basic (ou QBasic), encore plus simple que Turbo Pascal, semblait un bon choix mais nous étions à la recherche d'un langage qui nous permettrait facilement de programmer dans Windows (donc avec un support graphique avancé, souris, etc.). Le meilleur choix s'avéra être Visual Basic 4.0, autant pour sa ressemblance à Qbasic que pour son interface graphique sous Windows 95 avec *programmation par objet.*

Une autre traduction s'imposait. Certes, il y avait un lien entre des caractères comme «{\l.;» et la vraie position numérique du Power Glove®, mais il fallait trouver lequel!

En recherchant dans les fichiers textes sur le gant que nous avions trouvés sur Internet, nous nous sommes rendus compte que bien des programmeurs se référaient au langage hexadécimal afin de décoder le gant. Cette solution nous parut tout à coup évidente car ce langage était utilisé par les ordinateurs pour identifier des adresses de mémoire, des codes de couleur, bref, de tout.

Nous avions une piste pour mieux visualiser les données de l'interface. Nous avons donc entrepris de faire un programme en Visual Basic (VB) fonctionnant sous Windows 95 et traduisant chaque élément de la chaîne de caractères en valeur hexadécimale. Après plusieurs tentatives, nos efforts n'ont pas été vains: nous avions alors une application lisant les fameuses données d'une façon intelligible. Il ne restait qu'à prendre note des différentes positions du gant.

Voici donc un résumé de notre montage discuté jusqu'à présent:

1 Power Glove® et émetteurs: envoient des données *encryptées* concernant les coordonnées du gant.
2 Interface Menelli: permet la conversion de ces signaux en format RS-232 compatible aux PC.
3 Application en Visual Basic: permet de traduire des valeurs incompréhensibles en informations significatives.
4 L'interface Menelli: Comme nous voulions contrôler le bras en temps réel, il fallait éliminer toute source d'erreur dans la synchronisation de la puce du Menelli avec la cadence de notre port série. Nous avons donc opté de mettre l'interface en mode requête pour que ce soit notre programme qui contrôle les demandes d'envoi de données. De cette façon, le logiciel contrôle le matériel et non le contraire.

La première partie de notre montage expliquée, passons à la suivante.

PÉRIPHÉRIQUE DE SORTIE

Cet ensemble de robotique comprend deux pièces d'équipement qui nous sont essentielles pour assurer le bon fonctionnement du robot: l'interface ROBIX RCS-6 et les six (6) servomoteurs. Nous vous expliquerons au meilleur de nos connaissances ces composantes.

Les servomoteurs

Un servo est un moteur composé d'un potentiomètre qui est le dispositif capteur de position servant à alimenter la boucle de rétroaction. La commande de position est réalisée en alimentant le moteur par une série d'impulsions. Le contrôleur ajuste la hauteur, la largeur et la vitesse de répétition des impulsions. Généralement, et dans notre cas, cet appareil est l'interface ROBIX RCS-6 que nous

étudierons plus loin. Nos servos sont de marque HiTech®. Cela importe peu, sachant que les servos ont tous une taille basée sur un standard. De plus, ils utilisent tous une méthode de contrôle similaire. Contrairement aux moteurs conventionnels, les servos sont limités à une rotation maximum de 180 degrés.

Nos servos sont de forme rectangulaire et sont munis d'un arbre de rotation dentelé à une des extrémités. Trois fils de couleur différente sortent du côté opposé à l'arbre de rotation et vont se brancher à l'interface. Ces trois fils sont ceux du voltage, du contrôle et de la mise à la terre. Les servos fonctionnent normalement sous une tension de 4,8 volts mais ce voltage peut varier entre quatre et six. Le fil de contrôle sert à positionner le servo et la mise à la terre renvoie le courant. Ces trois fils sont connectés à l'interface ROBIX RCS-6 qui lui contient un processeur avec lequel nous devons communiquer par la programmation. À l'arbre de rotation peut être ajouté un système d'attache afin de pouvoir faire exécuter un travail au servo. Dans le cas de notre robot, l'arbre de rotation est directement lié à l'articulation.

Les servos se contrôlent en leur envoyant des impulsions de différentes largeurs. Ces impulsions ont une largeur minimum, maximum et de repos ainsi qu'une vitesse de répétition. Ces valeurs ne sont malheureusement pas standards et diffèrent d'un modèle de servo à l'autre. Par contre, une convention a été établie et semble être bien suivie. La convention instaure la règle qu'une impulsion de 1500uS correspond à la position de repos (neutre). La position de repos signifie que le servo a le même potentiel de rotation en sens horaire qu'anti-horaire.

Les servos sont des moteurs actifs, ce qui signifie que s'ils reçoivent la commande de tourner jusqu'à une certaine position, ils garderont activement cette même position. Si une force externe tente de les déloger, ils forcent en sens opposé à cette force de manière à lui résister. Bien sûr, la force que peut tolérer un servo n'est pas très grande. Le moment de force de retenue du moteur équivaut à placer 2,25 kg à la distance de cinq (5) centimètres de l'arbre de rotation.

Les servos ne peuvent pas garder leur position indéfiniment. L'impulsion de position doit donc être renvoyée pour ordonner au servo de ne pas bouger. La vitesse de répétition ne doit pas être plus longue que 25ms. Si après ce laps de temps aucune impulsion n'est envoyée en direction du servo, celui-ci se désactive et ne garde pas sa position comme désirée.

Dans notre projet, nous n'aurons pas de contrôle direct sur les largeurs d'impulsions et la vitesse de répétitions. Les positions du maximum, du minimum et du repos correspondent respectivement à 2000uS, 1000uS et 1500uS. Une autre particularité de nos moteurs est la rapidité de l'accélération, c'est-à-dire le temps que prend le moteur pour bouger de la position maximum à la position minimum. Nos servos HiTech® prennent moins de deux secondes pour effectuer ce parcours ce qui est très bien.

L'Interface ROBIX RCS-6

L'interface Robix possède un type de sortie et deux types d'entrées. La sortie contrôle les servos en envoyant les impulsions décrites en détail plus loin. Cette pièce se connecte via le port d'imprimante de l'ordinateur, appelé aussi «LPT1». Nous n'avons utilisé que les six sorties servomoteurs, et la connexion imprimante. Nous avons décidé d'omettre les autres qui ne nous étaient pas utiles pour le projet. Nous ne maîtrisons pas totalement le fonctionnement de ce qui se trouve sur cette interface, mais nous comprenons son rôle.

Afin de mieux comprendre le signal électrique que l'interface Robix envoie au robot, nous avons mesuré le signal transmis avec un oscilloscope. Les diagrammes obtenus montrent une oscillation ainsi qu'une ligne se répétant à intervalles réguliers. Cette ligne est transmise par le fil jaune (ou le signal), et correspond à une base de temps sur laquelle se fiera le robot afin de se déplacer en synchronisation au message envoyé. L'oscillation possède une dépression peu accentuée ainsi qu'un creux plus prononcé ultérieurement. Ce dernier signal est donné par le fil rouge et représente la quantité d'énergie envoyée aux moteurs. Si les moteurs sont soumis à une tension, les creux de l'oscilloscope s'élargissent et s'approfondissent, le voltage étant plus grand pour maintenir le moteur à sa place. Si le moteur bouge (volontairement), les creux diminueront, la quantité d'énergie étant plus faible pour bouger un servomoteur que pour le laisser immobile.

Le programme Robix

Avec l'ensemble Robix est fournie une disquette contenant le programme permettant de communiquer avec notre robot et de lui apprendre des manœuvres. Bien que nous n'ayons pas utilisé ce programme, nous allons l'expliquer afin que le lecteur comprenne comment le contrôle du bras robotisé se fait par le logiciel.

Le logiciel a été écrit en QuickBasic mais les *librairies* ont été faites en langage C (un langage plus compliqué). Le programme, pour éviter de la programmation à une personne néophyte qui voudrait seulement manipuler le robot, emploie une syntaxe traduite du langage originel de programmation. Ce langage, appelé langage script, simplifie énormément la commande du bras articulé. À titre d'exemple, si nous voulions bouger le moteur #3 de 40 unités, il suffisait d'écrire dans l'éditeur du programme

Robix «move 3 by 40». Le contrôle peut soit être relatif (à une position définie à l'avance), soit absolu. Dans ce dernier cas, le robot bougera par rapport à son point d'origine, qui est habituellement le point milieu entre les deux extrêmes physiques de son «arc de mouvement».

En plus d'un simple écran éditeur, ce même programme contient un mode d'enseignement («teach mode»), permettant l'élaboration de séquences de mouvements et de routines pour le bras. Ce mode est utilisé aussi pour contrôler chaque moteur individuellement en temps réel.

Mode enseignement Robix

Mais il n'y a pas que ce programme sur la disquette. En cherchant plus loin, on trouve un fichier nommé «Qbdemo0.bas». Ce fichier contient un programme écrit en QuickBasic permettant de faire des applications utilisant le bras. Nous nous sommes basés sur le modèle de ce programme afin de faire notre propre application pour contrôler le robot.

La programmation pour le contrôle du robot

Jusqu'à présent, nous avons vu les deux programmes principaux sur la disquette fournie avec l'ensemble de robotique. Mais aucun d'entre eux ne faisait ce que nous voulions, soit être capable de faire bouger les six servomoteurs du bras en même temps à partir d'une source autre qu'un fichier déjà écrit en langage script. Nous voulions guider notre robot à l'aide du Power Glove®, et donc nous devions faire nous-mêmes ce programme, en nous inspirant, à titre de référence, sur l'application et les fichiers de code déjà présents.

Évidemment, voyant que tous les programmes pour le robot Robix étaient écrits en QuickBasic, nous avons opté pour ce langage. En nous basant sur les procédures du fichier de code Quickbasic vues, nous avons d'abord tenté de contrôler chaque moteur du robot à l'aide d'une touche sur le clavier. Il nous a fallu plusieurs semaines seulement pour faire un programme nous permettant simplement de tourner le moteur d'un sens ou l'autre. Tout cela devait s'appliquer aux six moteurs. Mais cela ne suffisait pas: nous ne savions pas si ce programme pouvait contrôler six variables simultanément (soit les six moteurs), le clavier ne pouvant accepter qu'une donnée à la fois.

C'est alors que nous avons pensé au langage script. Ce langage, moins complexe que notre propre code, pouvait, avec une unique phrase, bouger plus qu'un servomoteur à la fois. En alliant ce dernier langage à notre petit programme, il était maintenant possible de tenir plus d'une variable en ligne de compte, et donc de guider plus d'une articulation à la fois. Il ne restait donc, en principe, qu'à ajouter le gant.

Voici le résumé de cette dernière section:

1 Le programme Robix (ou notre programme hybride) envoie des commandes via le port imprimante.
2 L'interface Robix RCS-6 reçoit ces ordres, les traduit en signaux électriques oscillant dans le temps, et les envoie via les sorties vers les servomoteurs.
3 Au niveau du bras, chaque moteur reçoit son impulsion et bouge du nombre d'unités commandées par l'application sur l'ordinateur.

Enchaînons maintenant avec la dernière section de cet article.

LA FUSION DES DEUX APPAREILS

Nous recevons les informations du Power Glove®. Nous pouvons, en théorie, contrôler le bras en temps réel. Que reste-t-il à faire? Allier ces deux programmes! En prenant les six variables importantes du programme de Visual Basic et en les transmettant via un fichier au programme hybride de Quickbasic/script, nous pouvons terminer le projet.

Nous avons d'abord pensé que ces deux langages (que sont Visual Basic et QuickBasic) étaient compatibles, ce qui signifie que nous pouvions simplement fusionner leur code. Bien que ces deux langages proviennent du même auteur (Microsoft) et portent nom semblable, il n'en est pas ainsi. Cela posa un sérieux problème. Mais nous ne pouvions abandonner si près du but.

Nous tenions mordicus à avoir une interface graphique dans Windows 95, donc pas question de traduire le programme de VB en QuickBasic. Nous ne savions pas non plus comment recréer notre programme hybride pour le robot en Visual Basic, car la programmation par objet est très complexe. De toute façon, il fallait trouver une solution rapidement, car nous étions déjà à la mi-session.

Une autre solution imaginée était de demander à la compagnie Robix si elle disposait de *fichiers DLL* afin que nous puissions intégrer le contrôle même du Robot dans Windows 95. Mais la compagnie nous affirma qu'elle n'avait malheureusement pas créé de tels fichiers.

La clé du problème (ou une des clés) se trouvait dans Windows 95. Ce système d'exploitation, fonctionnant en mode 32-bit, peut faire rouler deux applications simultanément. En ouvrant le programme de VB et le programme hybride en même temps, nous avions réglé une partie du problème. Mais que faire du transfert de données critiques qui doit avoir lieu? Sans cette étape primordiale, nous ne pouvions guider le bras.

Il existait pourtant un moyen: un fichier texte. En écrivant sur le *disque dur* de l'ordinateur les informations que le

gant transmet au programme de Visual Basic, ces mêmes données peuvent êtres soumises au programme de Quick-Basic. Cette dernière application n'aura qu'à ouvrir ce fichier, à prendre les variables importantes, et à les utiliser comme bon lui semble. Ainsi, le transfert de données se fait complètement. À chaque fois que l'application de Visual Basic reçoit des informations, elle les écrit dans un fichier texte qui est aussitôt lu par le programme hybride commandant le bras.

Maintenant, une grande partie du projet était réalisée. Mais il restait à contrôler efficacement le robot et à calibrer ses mouvements. Nous discuterons de tout ceci dans la section de l'analyse des données.

ANALYSE DES DONNÉES

Ayant enfin réussi à contrôler le bras en temps réel, nous étudierons à présent le contrôle et la calibration du bras.

Les variables hexadécimales lues du fichier texte sont en format de chaînes de caractères (ex.: «A0») et doivent être converties en nombres entiers (ex.: 160) afin de pouvoir être additionnées et multipliées à d'autres valeurs. La première étape de notre programme hybride est donc de convertir les informations provenant du fichier texte. Ensuite, un procédé individuel à chaque type d'information est utilisé.

Les coordonnées en X et en Y suivent essentiellement la même transformation: elles sont comparées à la dernière valeur reçue, et la différence est multipliée par un facteur afin de donner la nouvelle position du bras. Un mécanisme plus simple et plus rapide est utilisé pour la pince: au-delà d'une certaine valeur, elle se referme complètement. Le poignet passe par une prise de décision en relation à chacun de ses 12 degrés possibles. Cette méthode est plus lente mais permet une manipulation plus précise des positions limitées du poignet. Les boutons seront expliqués plus loin dans la section.

Analyse des articulations et des mouvements individuels du robot.

Nous avons dû changer certaines articulations du robot afin de rendre ce dernier plus compatible aux diverses fonctions du gant. La première articulation est apparentée à un bras et est montée sur un axe vertical. Elle tourne de 180 degrés sur un axe horizontal. La deuxième articulation est principalement une réplique de la première mais est utilisée pour l'effet «moulinet» (expliqué plus tard). Les troisième et quatrième servomoteurs sont fixés d'un côté du bras afin de bouger dans un rayon de 180 degrés sur un plan vertical. Le cinquième moteur tourne le reste du bras (soit la pince et son articulation) comme un poignet. Enfin, le dernier moteur active une pince, tournant dans un sens pour la fermer et de l'autre pour l'ouvrir.

Voici donc un résumé des fonctions motrices du bras:

moteur #1	- mouvement X
moteur #2	- mouvement X (moulinet)
moteur #3	- mouvement Y
moteur #4	- mouvement Y (moulinet)
moteur #5	- poignet
moteur #6	- pince

L'effet «moulinet»

Afin de reproduire un effet de profondeur dans le contrôle du bras robotisé, nous avons employé deux servomoteurs à l'application de cette caractéristique peu commune, un dans le sens horizontal et l'autre dans le sens vertical. Les articulations fonctionnant en moulinet tournent du même nombre d'unités que le moteur principal, mais en sens opposé, ce qui produit une rétraction du bras dans le sens X ou Y. Ces moteurs ne sont actionnés que si une différence notable dans l'axe des Z est détectée, autrement ils restent immobiles et le bras ne tourne qu'avec ses articulations principales, soit les moteurs 1 et 3. À titre d'exemple, si l'utilisateur du gant déplace son gant de 500 unités vers la droite en l'éloignant des capteurs (donc un mouvement diagonal), le moteur #1 bougera de 500 unités et le moteur #2 tournera de -500 unités, préservant la position vers l'écran de la pince tout en la rétractant.

INTERPRÉTATION ET UTILISATION DES RÉSULTATS

Nous avons réussi notre système de transmission de données. Les informations du gant se transmettent sans erreur au robot.

Un tel montage a été fait de manière amateur. Ce même système pourrait être monté en usine. Un technicien équipé d'un gant analogique de haute précision pourrait coordonner les mouvements de plusieurs bras robotisés semblables reliés en chaînes afin d'augmenter la vitesse et l'efficacité de la production. Les robots pourraient faire le même geste simultanément (comme peinturer plusieurs pièces d'automobiles en même temps).

Autres objectifs atteints

Ne pouvant faire taire notre imagination, nous avons cru bon d'améliorer notre projet.

La première idée qui nous est passée par la tête est celle de pouvoir contrôler notre robot sans contact direct avec celui-ci. Nous avons donc installé une caméra vidéo et avons affiché l'image sur le moniteur. De cette manière, il nous est possible de contrôler le robot par «télévision».

La capacité pour l'utilisateur de pouvoir sélectionner deux modes de précision différents nous permet de hausser la fiabilité du bras robotisé. En mode haute précision, le robot bouge de quelques degrés à la fois, ce qui nous permet d'effectuer des tâches où un simple faux mouvement pourrait engendrer une catastrophe mondiale! En mode basse précision, il se meut beaucoup plus rapidement, ce qui nous permet d'effectuer des mouvements de manière à accélérer, par exemple, le transport de boîtes d'une mine à une usine.

Un autre ajout très pratique consiste à créer une routine nous permettant de remettre le robot en position de repos. Dans la même optique, nous avons programmé des boutons afin de pouvoir contrôler individuellement chaque servomoteur. Cela devient extrêmement précis si utilisé conjointement avec le mode haute précision!

Améliorations prévues

Certaines améliorations ont été prévues. Le problème lors de la création d'un projet est le temps qui est trop rapide ou trop court. Nous avons donc pensé à quelques améliorations qui pourraient être faites d'ici notre participation à l'A.R.C.

Maintenant que nous contrôlons tous les moteurs du robot avec le Power Glove®, notre objectif premier est atteint. Même si la date limite du projet est maintenant passée, nous avons greffé d'autres objectifs.

En enregistrant les données de façon permanente dans un fichier texte différent, il serait possible de créer des routines ou des séquences de mouvements. D'un bouton sur le gant ou d'une touche du clavier, le robot pourrait aller prendre un outil particulier placé à ses côtés et ensuite permettre à l'usager de reprendre le contrôle.

Présentement la pince ne ferme que totalement. Il serait possible d'actionner différents niveaux de fermeture du poing avec un des boutons sur le contrôleur du gant.

Suggestion de recherches futures

Notre système de transmission de données est complexe, certes, mais la quantité d'informations trans-mises est minime: seulement six valeurs. Si on ne tient pas compte des langages utilisés, ce montage fonctionnerait très bien sur un XT (la première génération des PC en 1978). Le flot de données étant très rapide, on pourrait facilement contrôler ce robot par communication téléphonique ou par Internet. Il suffirait dans ce dernier cas d'envoyer les informations critiques par modem au lieu de les inscrire dans un fichier texte.

Conclusion et remerciements

C'est donc par un effort de créativité que nous avons pu mener à terme notre projet. De plus, ce projet nous a apporté de nouvelles notions de physique (fonctionnement des servomoteurs) et nous a appris de nouvelles commandes en programmation. Mais surtout, notre recherche nous a appris que tout est possible en ce qui concerne l'informatique et que, avec un peu d'imagination et de sens logique, nous pouvons arriver à faire n'importe quoi, même si nous ne sommes pas encore des programmeurs aguerris.

Remerciements spéciaux

Nous voudrions remercier les personnes suivantes:

- René Cossette, professeur de physique, pour son support;
- Ghislain Parent, technicien en électronique à Nortel pour son aide précieuse au point de vue du matériel ainsi que de la compréhension théorique de notre système.
- Louis Robichaud, technicien au laboratoire de physique, pour ses connaissances en électronique;
- Ginette Trudeau, coordinatrice du programme Dec Plus, pour nous avoir donné la chance de mener un projet d'une telle envergure et pour nous avoir encouragés de manière soutenue tout au long de la session;
- Nous terminerons cette section par un grand remerciement à tous ceux qui nous ont encouragés, soit nos parents et amis.

BIBLIOGRAPHIE

Notre projet, curieux à dire, n'a pas requis beaucoup de documentation écrite. Nous possédions la plupart des principes et connaissances requises pour l'élaboration d'un tel système de transmission de données, étant déjà des passionnés de l'informatique. Tout de même, il nous a fallu l'expertise de certains spécialistes (voir les remerciements) afin de mener à terme notre projet dans le temps requis. Voici quand même les quelques sources consultées.

Références informatiques:

- Fichier d'aide (vb.hlp) du langage Microsoft Visual Basic 4.0 version 32 bit pour Windows95.
- Fichier d'aide (Qb45advr.hlp) du langage Microsoft QuickBasic 4.5 pour DOS.

Adresses Internet utilisées:

- Renseignements utiles sur le gant et l'interface Menelli. ftp://ftp.psych.utoronto.ca/pub/vr-386/
- Questions fréquemment posées sur le Power Glove®. http://www.spies.com/jet/vr/faq-0.3_toc.html#SEC1

Schémas et détails sur le Menelli

- http://www.cms.dmu.ac.uk/~cph/Public/Menelli/

GLOSSAIRE

Voici une explication de tous les mots apparaissant en italiques qui pourraient causer problème aux non-initiés!

- Code: C'est l'ensemble des routines, des lignes d'instructions, et des paramètres dictés dans un programme. C'est essentiellement ce qui fait fonctionner une application.
- Disque dur: Disque magnétique à l'intérieur du PC où tous les programmes sont stockés.
- Encryptées: Données enregistrées selon un algorithme, ne pouvant être lues que par la personne possédant la clef de cette formule.
- Fichiers DLL: Fichiers contenant des libraires de routines et de procédures servant dans l'interface graphique de Windows.
- Librairies: un ensemble de fonctions couramment utilisées par un programme. Au lieu de les intégrer dans le fichier (ce qui prendrait plus de mémoire), on les store séparément dans un fichier distinct afin qu'elles puissent être partagées par d'autres applications.
- Microcontrôleur: Une puce possédant non seulement un microprocesseur, mais ses propres et sorties, sa propre mémoire vive et son propre espace de stockage, ou mémoire morte. Un programme doit être chargé dans la mémoire de cette puce pour que l'on puisse s'en servir.
- Port série: prise située en arrière d'un PC servant à connecter un périphérique (équipement) externe, comme une souris ou un modem. Un ordinateur en a au moins deux, codés COM1 et COM2 en langage technique.
- Programmation par objet: Contrairement à la programmation en Basic ou en Turbo Pascal, qui ne requiert que de simples lignes de code (instructions), la P.P.O. est un nouveau concept en informatique, adapté pour des systèmes d'exploitation graphiques comme Windows 95. Chaque section du code est dorénavant associée à un objet (bouton, glisseur, boîte de texte, icône, liste...) sur l'écran. Certaines sous-procédures sont maintenant des procédures par événement, donc engagées seulement quand l'utilisateur fait telle ou telle action à l'écran. Ceci complexifie assez le programme mais compense en apparence professionnelle: n'importe quelle application, simple ou complexe, faite en Visual Basic, incorpore toute les fonctions de base de Windows (système de fenêtre, support de souris automatique, de CD-ROM, de multitâche, etc.), en plus de ressembler graphiquement à n'importe quel autre logiciel commercial fait pour ce système d'exploitation.