Projet Robot Centaure

Année 2013-2014
IMA4 S8

Étudiants :
Clément TACHÉ
Liying WANG

Enseignants :
Robert Litwak
Xiavier Redon

Table des matières :

Remerciements:

Nous tenons premièrement à remercier Monsieur Xavier Redon et Monsieur Robert Litwak pour leur aide et leurs conseils lors de la réalisation de ce projet, ainsi que l'école Polytech Lille et le département IMA pour la mise à disposition des locaux et de l'ensemble du matériel nécessaire a son bon déroulement.

Introduction:

Il est demandé en quatrième année du département IMA à Polytech Lille de réaliser un projet durant toute la durée du semestre 8. Nous avons choisi de travailler sur le Robot Centaure. Le projet Robot Centaure est un projet en cours à Polytech Lille depuis plusieurs années, lors de l'année précédente de nombreuses réparations ont été effectué sur celui-ci et il est désormais en état de fonctionner. Notre objectif était de le faire fonctionner de manière plus optimale et d'améliorer son comportement global. Pour cela l'emploi de toute les compétences enseignées lors des deux premières années passé à l'école est nécessaire et le projet requiert donc la coopération des deux filières du département IMA : Filière Système Autonome et filière Système Communiquant.

Présentation et état des lieux:

Le Robot centaure, nommé ainsi de par sa forme, est un robot de grande taille dont l'objectif est de se mouvoir au sein du hall de manière autonome et sure tout en diffusant des messages sur son écran intégré. Remis en état l'an dernier après avoir était laissé a l'abandon pendant plusieurs années, la partie matériel du robot est en bonne état et totalement fonctionnelle, la majorité des améliorations a lui apporter sont donc d'ordre logiciel.

La partie mécanique du robot est composé d'une "boite" comportant les éléments du robot, de deux roues motrices à l'arrière, d'une roue folle à l'avant et d'une structure en aluminium supportant un écran LCD.

Le robot est composé de plusieurs éléments relié électriquement entre eux: Deux batteries de 24V en série, un arduino mega, un écran, une Kinect, plusieurs capteur infrarouge, deux roues motrices indépendantes, un PC et une boite a fusible. L'ensemble des éléments sont agencés au sein du robot ce qui permet d'assurer son autonomie.

Plusieurs programmes permettent de gérer les différents éléments du robot et seront détaillés plus loin dans le rapport.

Lors de la prise en main du robot nous avons mis en évidence que les codes qui nous ont été fourni ne permettent pas au robot de fonctionner, le plus gros de ce projet a donc été de remettre en fonctionnement le robot.

Objectifs du projet:

- Permettre au robot de se déplacer selon plusieurs mode choisi par l'utilisateur: Utilisation d'une télécommande ou mode autonome.
- Résolution des problèmes de déplacement du robot lié au glissement des roues et à la présence de la roue folle.
-Documentation et réalisation d'une aide a l'utilisation et a la résolution des problèmes existant sur le robot
-Amélioration des programmes déjà existant pour "fluidifier" le fonctionnement du robot.

1. Solutions Techniques

1.1 Partie Matérielle

La partie matérielle du robot est en bon état et il n'y a que peu de modification à lui apporter, le principal problème identifié est la présence d'une roue folle à l'avant du robot qui entraîne des écarts dans la trajectoire du robot et le glissement des roues sur le sol qui ont le même effet.

1.1.1. Analyse des solutions existantes

Fonctionnement de la partie mobile du robot : L'arduino reçoit des ordres venant de l'ordinateur, ses ordres proviennent soit de la kinect, soit de la télécommande (il est aussi possible de les envoyer directement depuis le logiciel IDE). Chaque ordre est traité dans une certaine partie du programme arduino et va entrainer une modification de la valeur de la tension appliquée aux moteurs, et donc une variation de leur vitesse de rotation. Chaque roue possédant son propre moteur il est possible de diriger le robot par différentiel de vitesse entre les deux roues.

Ordinateur intégré au robot : Le robot ne comporte pas d'ordinateur intégré au début du projet, l'utilisation d'un ordinateur externe est nécessaire. En effet la carte d'alimentation 12V du PC a été grillée lors de l'année précédente. L'installation du PC embarqué a donc été prévue.

Solutions pour l'arrêt d'urgence du robot : Des capteurs infrarouge sont répartis à l'avant et à l'arrière du robot afin d'éviter qu'il percute un objet ou une personne lors de ses déplacements. Ces capteurs envoient un signal à l'arduino qui va alors immobiliser le robot en arrêtant d'alimenter les moteurs. Il existe aussi un bouton d'arrêt d'urgence manuel a l'avant du robot, celui-ci permet de couper l'alimentation du robot en cas de disfonctionnement du programme. Un système équivalent est présent à l'arrière du robot.

1.1.2. Correction de trajectoire

Afin de corriger la trajectoire du robot lors de perturbations (glissement ou roue folle) il a été décidé de l'utilisation d'une boussole, cette boussole devait être compatible avec l'arduino mega utilisé, sa précision et sa fiabilité devait être élevé pour que le système de corrige pas des erreurs inexistantes. Il a été décidé d'utiliser la boussole HMC6352.

Cette boussole est prévue pour être alimentée entre 2.7 et 5,2V, sa fréquence d'acquisition est réglable entre 1 et 20Hz et sa résolution est de 0.5 degré ("répétabilité" de 1 degré). La librairie pour son utilisation est disponible en ligne et elle répond a tous les critères pour ce projet.

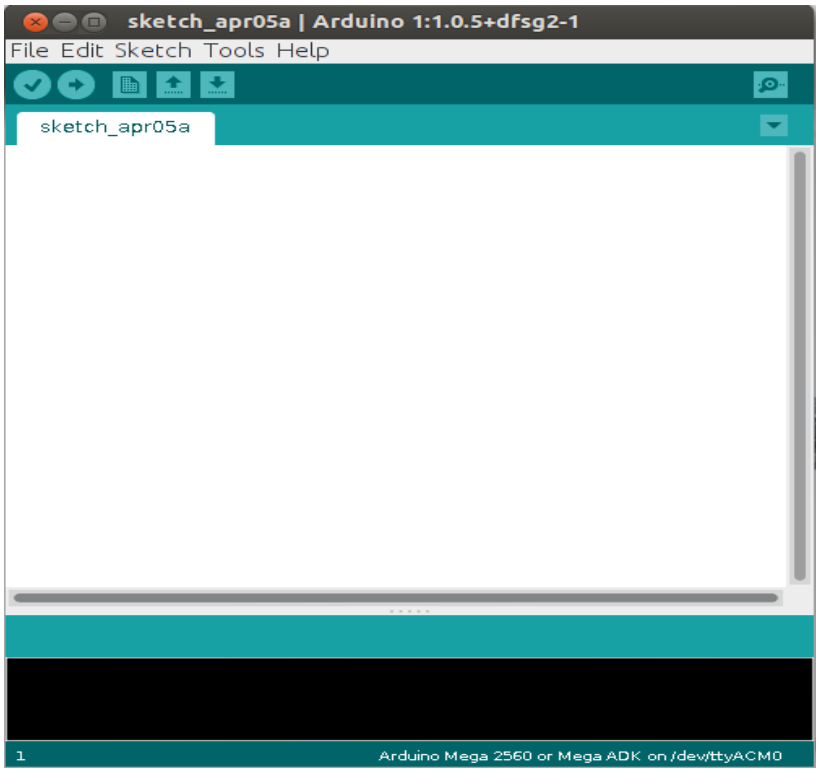
L'utilisation de cette boussole va permettre de récupérer l'orientation initiale du robot puis les orientations suivantes. Si les orientations suivantes ne correspondent pas à celle souhaité le programme arduino va permettre, en agissant sur les roues, de corriger cette orientation. Pour cela il nécessaire de mettre en place une commande par logique floue qui assurera le meilleur suivie de trajectoire possible.

Principe de la logique floue: Plus l'orientation du robot s'éloigne de celle souhaitée, et donc plus la tension en sortie de la boussole varie, plus l'arduino va modifier les valeurs de tension en entrée des moteurs de façon à augmenter la vitesse d'une des roues et donc corriger la trajectoire.

1.2 Partie Logicielle

La partie logicielle permet au Robot Centaure d'être commandé à partir d'appareils extérieurs ou de fonctionner en autonomie. Plus précisément, les appareils extérieurs sont la télécommande, la Kinect, ainsi qu'un téléphone portable utilisant une application Android. La Kinect permet au robot de fonctionner en autonomie. Les programmes se divisent en deux grandes parties: le programme Arduino, qui est écrit avec le logiciel IDE, et les programmes qui gèrent les communications entre les différentes parties du Robot.

Interface du Logiciel IDE qui permet de compiler et de charger directement le programme Arduino, mais aussi de communiquer avec celui-ci via la liaison série:



1.2.1 Fonctionnement par module

Cette partie du programme gère les communications entre les différents éléments du robot (Kinect, Télécommande, Arduino, Application Android) ainsi que la communication avec les utilisateurs (Images, sons). La communication entre les différentes parties du robot est réalisée en utilisant un tube, qui est un fichier, dans lequel on peut effectuer une écriture ou une lecture ce qui permet la communication entre les éléments.

a) Programme Arduino

Le programme Arduino gère le fonctionnement des moteurs des deux roues en fonction des signaux reçu en entrée de celui-ci. Pour ce faire il agit sur la tension en entrée des moteurs et donc sur leurs vitesses ce qui permet par un différentiel de vitesse de rotation des roues de diriger le Robot.

L'arduino permet aussi de renvoyer l'état des batteries ainsi que des capteurs infrarouge vers le PC. Cela permet de savoir quand la mise en charge du robot est nécessaire et quand un arrêt d'urgence c'est produit.

b) Programme Communiquant

- *Modes de communication*

  Il existe quatre types de communications dans ce système:

  La communication entre la télécommande et l'ordinateur est réalisée par le programme *gestionIrc.c*. Un capteur intégré sur le robot transforme les signaux infrarouges reçus en entiers, et les envois vers l'ordinateur via un port défini. Ensuite, pour que le liaison série puisse transmettre ces commandes vers l'arduinot, le programme *gestionIrc.c* effectue une écriture de cette commande dans le '*tubeIS*', de plus afin qu'une image puisse être chargée en réponse à cette commande, une écriture dans le '*tubeSIhm*' (Précisé dans la partie 'Interface Graphique et sonore') est effectuée sur le même principe.

La communication entre la Kinect et l'ordinateur est réalisée dans le programme *libkinect.c*, la fonction *depth_cb* calcule la route correcte puis écrit dans le *'tubeKinect'*. L'orientation de la Kinect peut aussi être réglée en appuyant sur les bouttons correspondants. (cf. Annex 4.3).

La communication entre l'application Android et l'ordinateur est réalisée dans le programme *commandes.php* et *envoyerCommande.php*.

La communication entre l'ordinateur et l'Arduino est réalisée par le programme gestionSerie ainsi que le programme de la partie Arduino. Le programme gestionSerie.c effectu une lecture dans l'un des quatre tubes, et envoie les commandes vers l'arduino. L'arduino envoie vers le PC des messages sur l'état des batteries ou des capteurs en cas d'arrêt d'urgence.

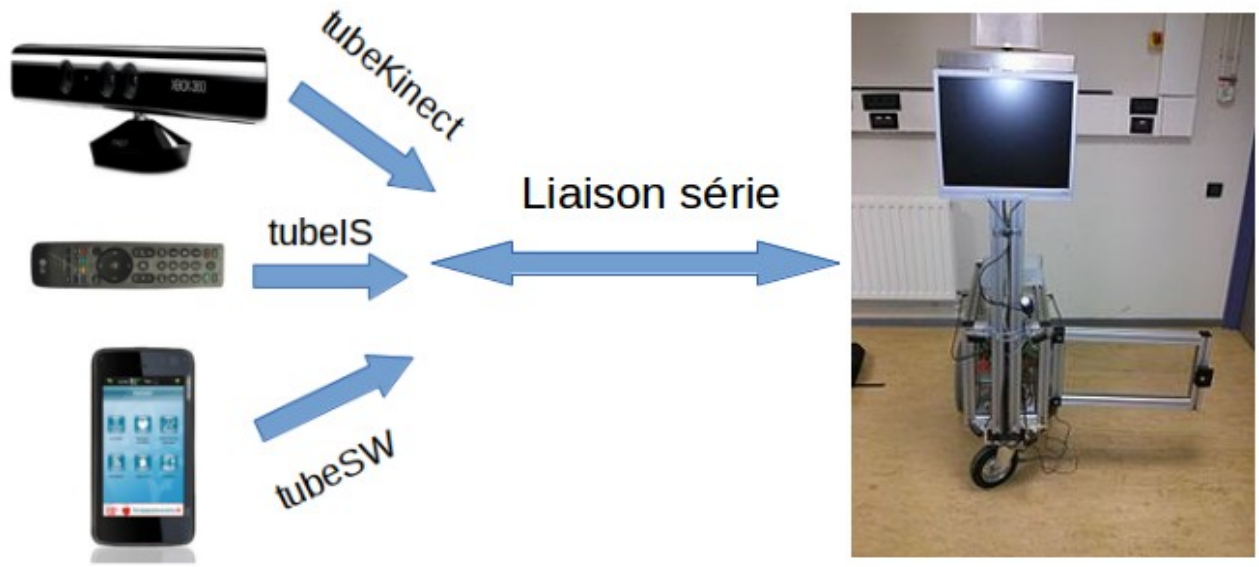
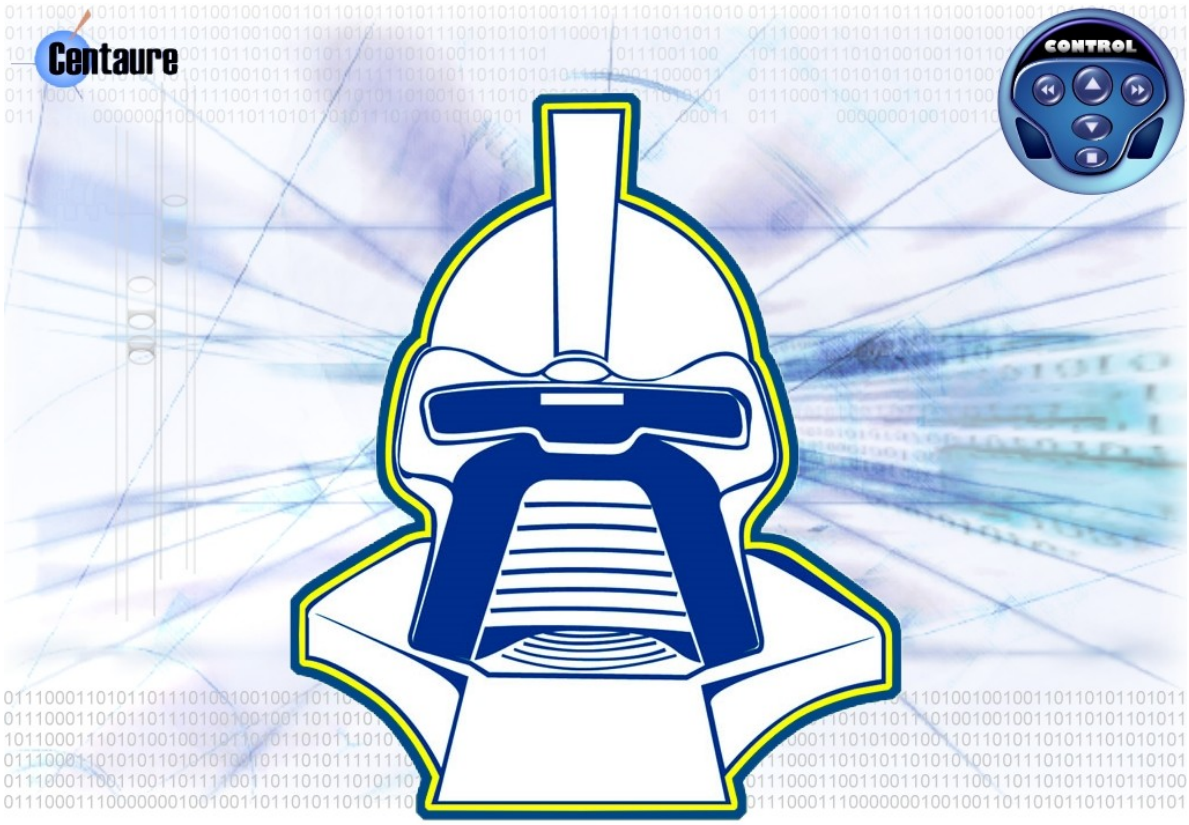


Schéma de principe

Interface Graphique et application sonore

Dans ce module, les programmes sont créés pour que le robot puisse communiquer avec son utilisateur. Il y a deux fonctionnalités principales. D'abord, l'affichage des images est réalisé par les programmes *gestionIhm.c* et *gestionEcouteIhm.c*, ces deux programmes utilisent un outil gtk2++ pour créer des interfaces graphiques. Le premier programme charge

une image d'accueil et le deuxième charge des images correspondantes quand l'utilisateur appuient sur les boutons de la télécommande ou les touches de l'ordinateur. Les appuis sur les boutons sont pris en compte par le programme *gestionIrc.c*, et les commandes sont envoyé par le *'tubeSIhm'* .



Exemple d'image affiché a l'écran

L'application sonore est réalisée par le programme *gestionSynthesVocale.c.* Ce programme est lancé quand le robot reçoit des commandes venant de la télécommande ou de l'arduino lorsqu'un problème est rencontré (ex: arrêt d'urgence).

- *Réalisation du multitâches*

Il y a plusieurs modes de communication entre les différentes tâches dans ce système. Afin de gérer ces différentes tâches, le programme *gestionThread* est réalisé. L'un des intérêts de l'utilisation de multi-thread est qu'il permet de diminuer le temps de changements de

contextes entre différentes tâches et d'augmenter l'efficacité d'exécution du programme. Tous les threads créés sont configurés 'detached' pour qu'ils ne s'attendent pas entre eux.

1.2.2 Compilation et exécution du programme

a) Partie Arduino

Le logiciel IDE doit être installé pour compiler et exécuter le programme d'Arduino.

Le type de la carte (*cf. Partie matérielle*) doit aussi être configuré dans IDE :

Tools/Board

Le port série doit être configuré:

b) Partie Communication

- Installation des bibliothèques

Les bibliothèques suivantes sont à installer pour compiler le programme si vous travaillez sur un autre ordinateur que celui installé dans le Robot:

libfreenect-dev

freeglut3-dev

libusb-1.0

libirc

libirman

(Il y aura potentiellement autres bibliothèques à installer selon les ordinateurs)

- Configuration des ports

  Avant de compiler les programmes, il faut configurer les deux ports:

  - Le port pour la télécommande infrarouge dans le fichier ./Demon/gestionIrc.c :

  #define PORTCOMM1 "/dev/ttyUSB0"

  - Le port pour la communication avec l'Arduino dans le fichier./Demon/gestionSerie.c:

  #define PORTCOMM2 "/dev/ttyACM0"

  (Attention! Le port série dans le logiciel IDE doit être configuré de la même façon que celui-ci, sinon le robot ne peut pas être commandé.)

- Compilation

  Pour tester le programme:

  Un Makefile global doit être exécuté avant de compiler les sous programmes.

  Pour débuger le programme:

  'make debug' permet d'afficher les information de debug.

  Pour débuger la partie Ihm :

  'make debugihm' permet de débuger les programmes concernant ihm.

  Pour lancer la fenêtre de Kinect:

  'make -video' .

  Pour ajouter ou supprimer des programmes:

  Un makefile dans le répertoire Demon permet d'ajouter ou supprimer des objets.

- Exécution du programme

  Le programme principal centaure.c se situe dans le répertoire. /Demon.

  Pour exécuter le programme, il faut se positionner dans le répertoire global et effectuer la commande:

  (cd. /Demon ; . /centaure).

2.1 Amélioration du programme

2.1.1. Partie Arduino

Le code de la partie Arduino a dû être retravaillée afin d'assurer la cohérence entre les différentes partie du programme :

- Suppression de toutes les fonctions liées à l'utilisation de la boussole, leur présence empêchait le déplacement du robot puisque celle-ci n'était plus présente.
- Modification des ordres correspondant à chaque déplacement et simplification des fonctions de déplacement afin de faire disparaitre un grand nombre de bug lié à leur utilisation (blocage d'une roue, roues non synchronisées, etc…).
- Ces modifications ont permis de contrôler le robot directement via le port série et l'utilisation du logiciel IDE en attendant la correction du code partie communiquante.

2.1.2. Partie Communiquante

- gestionSerie

Sachant qu'au début du projet, la télécommande ne commandait pas le robot, nous avons étudié le programme gestionSérie, après avoir testé les programmes, nous avons trouvé les problèmes suivants:

D'abord, dans le programme gesitonIrc.c, les commandes envoyés par la télécommande sont récupérées par le programme, mais au lieu de les envoyé vers la liaison série, elles sont envoyées vers la Kinect.

Ensuite, dans le programme gestionSerie.c, la fonction qui récupère les commandes venants des quatre tubes n'est pas bien configurée. Plus précisément, la fonction select( ) suspends l'exécution du programme courant. Celui-ci reprend si au moins une des conditions suivantes est vérifiée :

- Réception sur un flux
- Fin d'émission sur un flux
- Réception d'un message d'erreur sur un flux

- Échéance de temps.

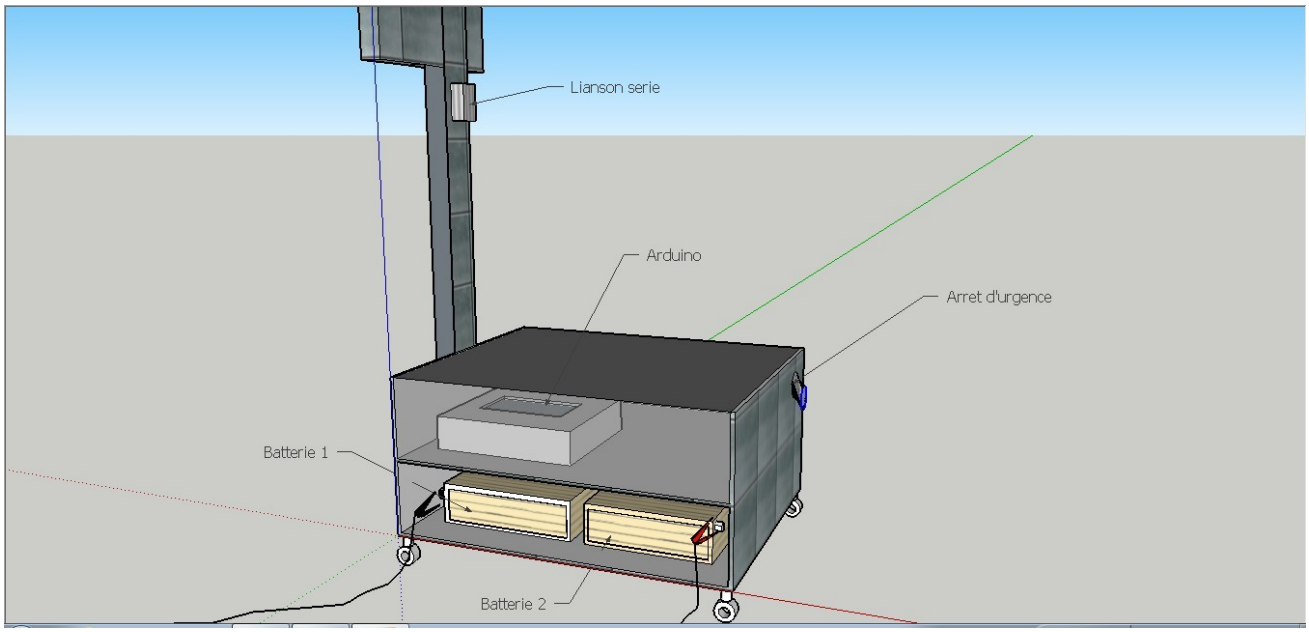
L'analyse de la valeur retournée par select( ) permet d'identifier l'évènement. (-1 = erreur, 0 = échéance, autre = évènement flux).

Le premier paramètre est le plus grand numéro de flux surveillé augmenté de 1, mais l'ancien programme ne définit pas la taille maximale parmi les quatre tubes.

Après avoir modifié les deux programmes, le robot peut être commandé par le télécommande.

- gestionIhm

Selon le programme principal, le thread Ihm et EcouteIhm sont correctement lancé, mais pendant l'exécution du programme, aucune image n'est chargée. Nous avons trouvé que la condition pour charger une image n'est pas correcte. Une image peut être chargée si elle est vérifiée par la fonction testImage().Nous avons aussi ajouté dans le programme une fonction qui affiche une image de dépannage quand l'utilisateur appuie sur la touche 'D' du clavier.



- Rangement du code et ‘Warning’

Afin de rendre le programme plus lisible, nous avons supprimé les codes qui ne sont pas utiles. Le programme gestionCaptureImage.c n’est plus utile car le capteur d’image est déjà remplacé par la Kinect.

Nous avons trouvé que la plupart des ‘Warnings’ sont générés par des programmes qui ne sont pas utilisés. Nous avons modifié le makefile pour que ces programmes ne soit pas compilés.

3. Conclusion

3.1 Limitation

- Partie Autonome

Le problème lié aux erreurs de trajectoire lors d'un déplacement rectiligne n'est pas totalement résolu puisque la boussole HMC6352 n'a pas pu être achetée et donc que la commande floue n'a pas pu être implantée. Il s'agit du dernier problème majeur sur le robot puisque l'ensemble de la partie matérielle est fonctionnelle et que le robot peut désormais fonctionner en autonomie totale.

- Partie Communiquante

Kinect:

L'exécution du programme avec le lancement de la Kinect peut conduire le programme à s'arrêter brutalement.

Interface Graphique :

Le chargement des images avec l'outil GTK2+ n'est pas assez bien géré. Les tailles des images doivent être ajustées pour que le chargement de plusieurs images soit possible.

Interface Son :

Le programme qui lance le son ne peut pas être testé sur l'ordinateur embarqué car celui-ci ne comporte pas de carte son, cependant ce programme marche en utilisant un ordinateur portable relié au robot.

3.2 Pour aller plus loin

Afin de rendre le robot plus interactif pour le grand public la réalisation d'un programme communiquant avec une base de donnée d'image liée à l'école et défilant à l'écran est envisageable, cela permettrait de faire circuler le robot dans l'école lors de certain événement (en fonction du nombre de personne dans celui-ci).

Certain bug du programme sont encore à régler et la boussole doit être implantée sur le robot.

4. Annexe

4.1 Pannes possibles et les solutions

La télécommande ne contrôle pas le robot:

Solution:

Vérifiez dans le programme ./Demon/gestionIrc.c si le port PORTCOMM1est correctement défini. Pour regarder les informations sur les ports USB:

- ls dev/tty

Vérifiez également dans le programme ./Demons/gestionSerie.c si le port PORTCOMM2 est défini de la même facon que celui de l'Arduino. Pour regarder le port utilisé par Arduino, utilisez le logiciel IDE, l'information est dans Tools/Serial port.
Vérifier si tous les devices sont connectés. Pour démarrer le robot, il faut brancher la liaison série avec l'Arduino, le récepteur infrarouge, ainsi que la kinect.

L'écran clignote

Solution:

Vérifiez le niveau de la batterie ainsi que le fil de télé entre l'écran et l'ordinateur.

La batterie est pleine mais n'alimente pas le robot:

Solution:

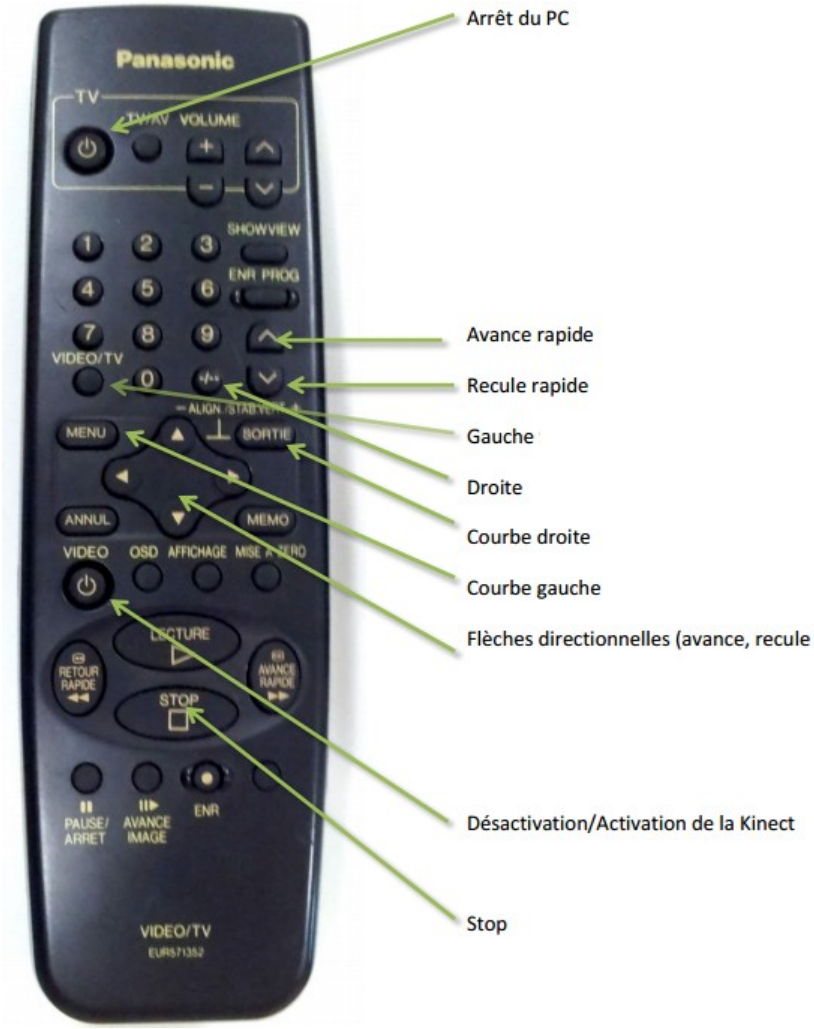
Vérifiez l'état des fusibles. Un faux contact au démarrage peut entraîner leur rupture.

Le robot ne réponds à la télécommande en créant du bruit mais ne bouge pas :

Solution :

Vérifiez si le bouton d'urgence est relâché.
Vérifiez aussi l'état du fil d'intérrupteur à côté des batteries.

4.2 Utilisation de la télécommande.



4.3 Utilisation de la Kinect.

Pour que la fenêtre puisse être lancée à l'exécution, pour la compilation:

- make video

Clavier qui gère les fonctionnements du kinect (Définis dans le programme *libkinect.c* la fonction *KeyPressed*):

'w': Ajuster l'angle du kinect à 30 degree vers le haut.

'x' : Ajuster l'angle du kinect à 30 degree vers le bas.

'y' : Récupérer des zones ombrées.

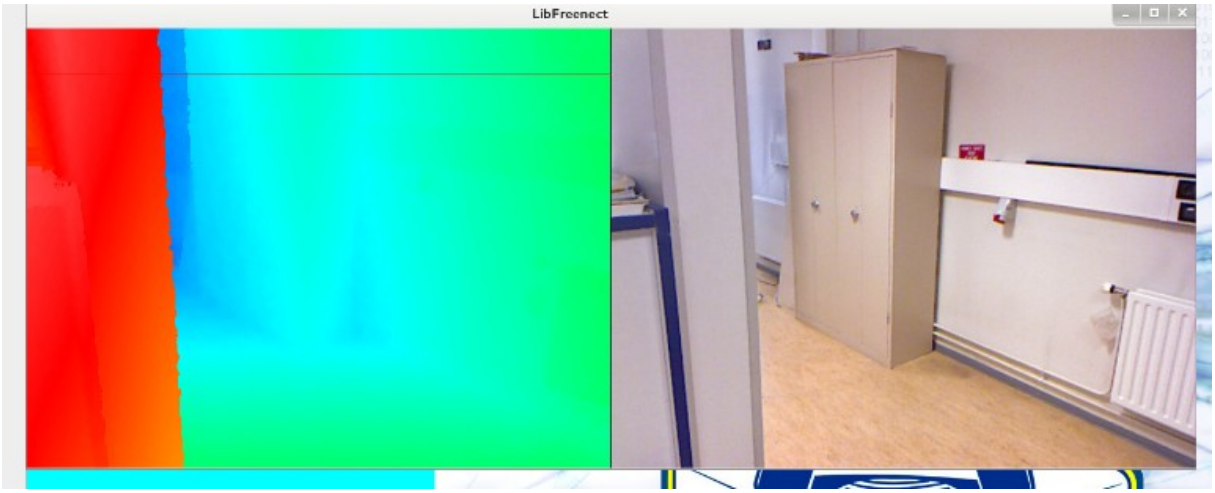
'1' : Led du Kinect vert

'2' : Led du Kinect rouge

'3' : Led du Kinect jaune

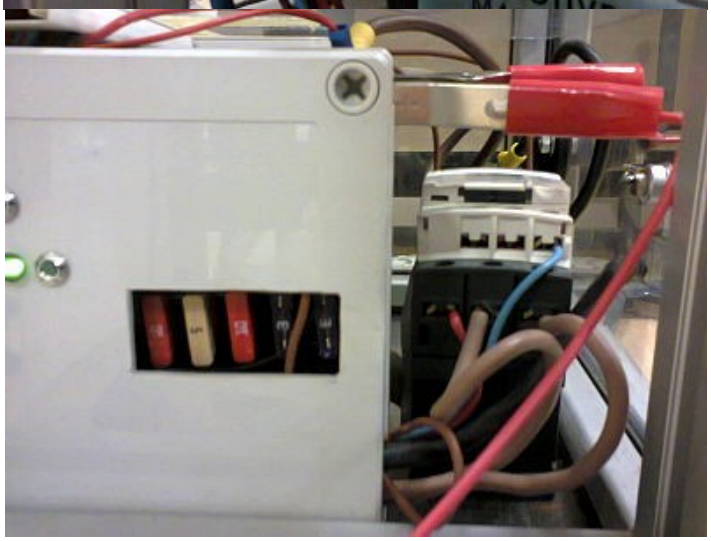
'4' : Led du Kinect vert clignote

'6' : Led du Kinect éteints.



4.4 Batteries.

La pince rouge du chargeur se branche sur la batterie de droite, la pince noire du chargeur sur la batterie de gauche.



4.5 Capteurs du Robot.

Les capteurs du robot sont opérationnels et répartis de la manière suivante :
Ils permettent au robot d'éviter de percuter un obstacle lors d'un disfonctionnement du programme kinect (ou d'une mauvais manipulation de l'utilisateur).