PM ROBOTIX

Membres de l'équipe :

CHAUDELET Christophe
ELDMAIR Christina
KARMAN François
RICHARD Sandra

Equipe française

1. Présentation du robot

Le robot actuel est composé d'une **armature en bois** associé à des **tiges de métal**, ainsi que du **mécano**.
Les cartes électroniques sont situées sur le haut du robot. Au dessus se situe un carter permettant de placer le bouton d'arrêt d'urgence, ainsi que le mât support de balise.

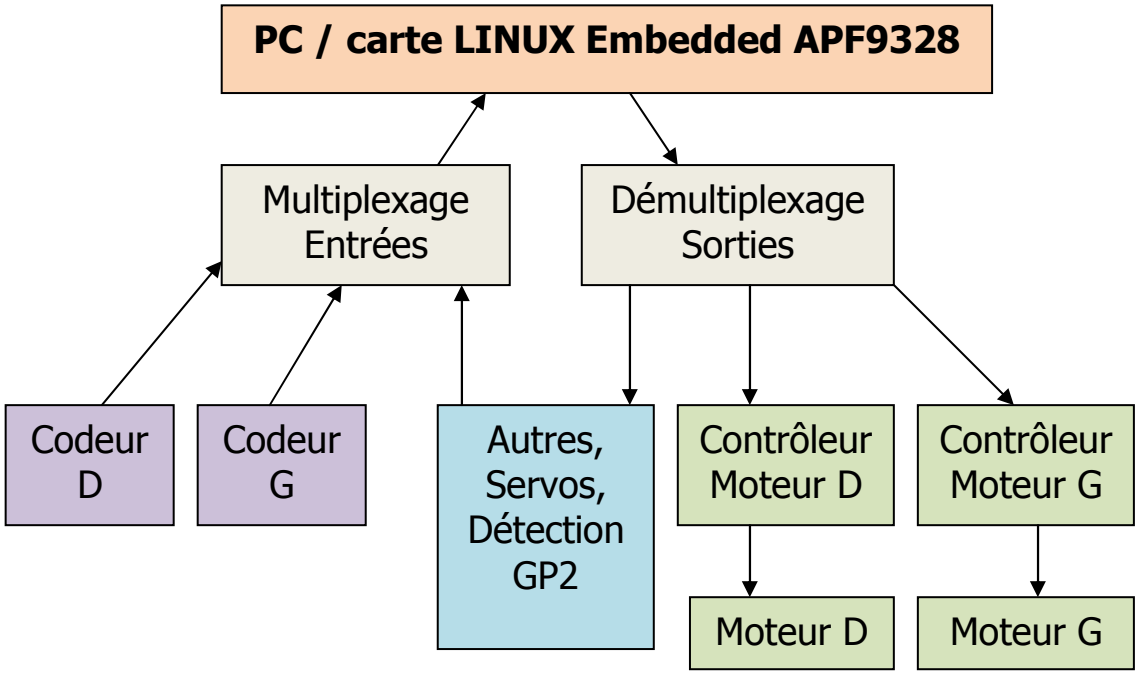
La structure porte à sa base le **bloc moteur** constitué de deux moteurs 919D541/1 liés à des roues de roller, le **mécanisme** reste très simple et permet de pousser les tomates qui se trouvent sur le chemin.

L'objectif est de pousser les tomates sur le chemin.

Photos face avant.

Le robot possède donc deux roues directrices placées sur le diamètre d'un **cercle de 38cm**. Le périmètre du robot fait donc :
P= 380*PI = 1194mm

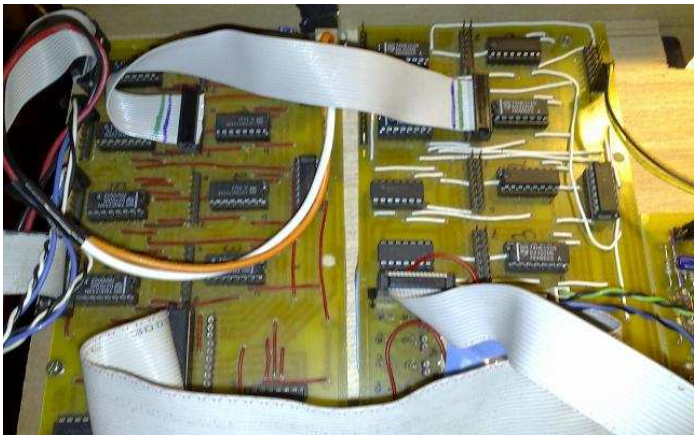
2. Fonctionnement général



Le principe de notre robot est basé sur une **architecture avec une carte LINUX embarquée ARMADEUS APF9328.**

2.1. Multiplexage / Démultiplexage

Le robot communique via le port parallèle aux 2 cartes de **multiplexage/démultiplexage**. Ces 2 dernières multiplient les entrées/sorties afin de permettre au PC de dialoguer avec les différents composants du robot. Nous obtenons ainsi **64 sorties et 48 entrées** avec seulement 10 entrées/sorties.



MULTIPLEXAGE I/O Carte blanche pour les entrées, carte rouge pour les sorties.

2.2. La carte APF 9328

L'APF9328 est une mini carte électronique composé d'un processeur optimisé et bénéficiant d'un fort taux prix/performance.
Elle est équipée avec :

- **Processeur ARM9 200MHz**
- **SDRAM/FLASH**
- **10/100Mbits Ethernet port**
- **a (200K gate) Spartan 3 FPGA**
- **RS232/USB**
- **97 I/O pins**
- **DAC / ADC**

http://www.armadeus.com

2.3. Outils de management de projet

Cette architecture nous permet de nous baser sur des principes simples et efficaces. Etant donné que les membres de l'équipe ne peuvent se rencontrer très souvent, nous gérons le projet via des **outils de management de projet (SVN, Trac)**. Nous pouvons ainsi nous partager les tâches de mécanique, électronique et informatique, mais aussi tous intervenir sur la programmation.

Pour le moment, tout le robot est surtout basé sur des techniques de **programmation en C++**. Notre objectif est de démontrer que nos choix techniques sont cohérents, et de les faire évoluer dans le cas contraire.

http://www.assembla.com

A propos des batteries, nous avons choisi d'utiliser **la technologie au plomb, merci à YUASA**, fiable et peu couteuse. Nous aurions aimé utiliser des **LiPo** (**Lithium-Polymère**) mais celles-ci demandent un budget beaucoup plus important, et une électronique plus précise pour mettre en place des protections suffisantes de sécurité.

L'asservissement des moteurs est effectué de manière logicielle (La correction PID est effectuée par le programme C++). Nous avons une **solution maison**, étant donné que les moteurs ne sont pas de type professionnel, et ne nous donne pas une symétrie parfaite.

3. Détection de l'adversaire

La détection de l'adversaire s'effectue grâce à 2 types de capteurs infrarouge (**GP2D02 et GP2D12**) permettant de détecter un obstacle jusqu'à 60cm. Les GP2 utilisés pour ce système, seront situé à une hauteur supérieure à la bordure du terrain et sur un axe horizontal.

De plus, le robot connaitra sa position sur le terrain par son **positionnement** (effectué par 2 roues codeuses droite et gauche liées aux roues) et calculée au fur et à mesure des déplacements.

Ainsi avec ses 2 premières informations, nous plaçons cette tâche de détection en priorité haute dans le programme afin de stopper le robot, et de le faire reculer puis tourner pour prendre une autre trajectoire.

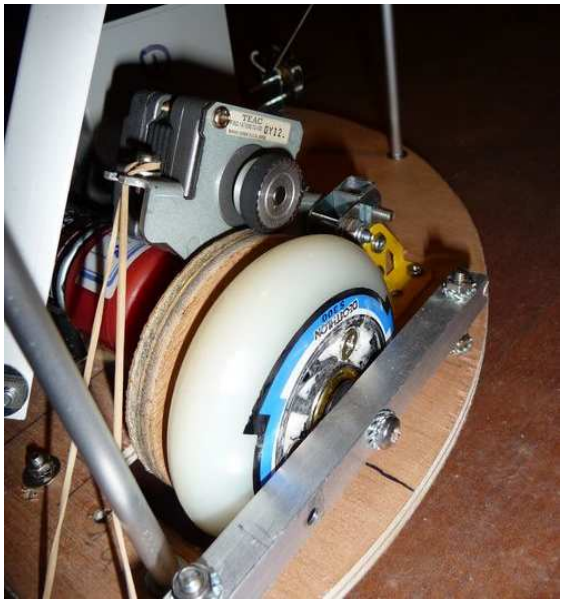
4. Informatique embarquée

Le programme est entièrement en **C++** afin de développer entièrement en objet.

Le robot fonctionne sous **Linux compilé Armadeus**. Pour programmer le robot et transférer le programme, nous utilisons une **connexion SSH avec un câble RJ45**. L'environnement de programmation est sous **Ubuntu avec Netbeans 6** (version C++) qui intègre très facilement la gestion de projet **SVN**.

5. CODEUR = MOTEUR PAS-A-PAS

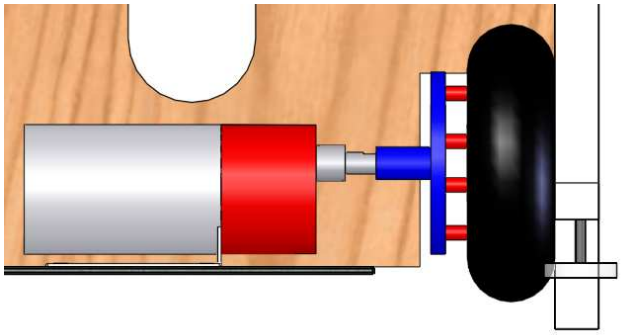
N'ayant pas le budget pour des codeurs professionnels, nous avons étudié puis testé l'utilisation de **moteurs pas-à-pas en codeur**. Grâce à une petite électronique détectant les faibles impulsions des bobines lorsque le moteur pas-à-pas tourne, un signal carré est envoyé au robot. Les moteurs pas-à-pas utilisés ont été récupérés sur des **vieux lecteurs de disquette 5'1/4**. Nous avons même réussi à **obtenir une quadrature de phase** sur chacun d'eux, afin de connaître le sens de rotation.

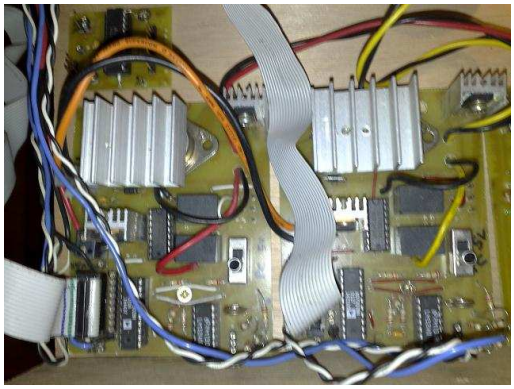


Moteurs pas-à-pas en codeur sur roue en bois faite maison

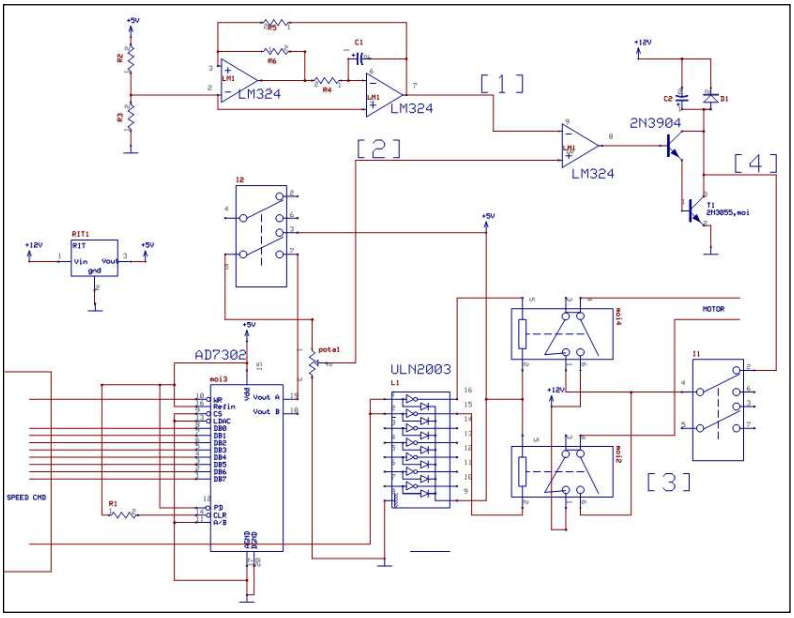
6. Motorisation « bon marché »

La **roue de roller** est montée libre en rotation sur une vis fixée dans le carter. Deux roulements assurent une liaison pivot sans frottements. Un système d'accouplement en bois, composé d'entretoises de métal, permet de lier la roue de roller à l'axe moteur.





Il y a une carte électronique par moteur. Nous avons créé ces 2 cartes avec des **composants bon marché** et un **système simple**. Chaque carte permet de transcrire une commande numérique en analogique sur le principe d'un **PWM** (Pulse Width Modulation), ou modulation de largeur d'impulsion.



Système de motorisation TRES simple

6.1. Principe :

A partir d'un mot binaire de 8 bits, le **convertisseur Numérique-Analogique** (CAN ou DAC) transforme la valeur du mot binaire en tension. Cette tension est ensuite injectée dans le système d'amplification opérationnel.

Un comparateur à Hystérésis (signal carré) est intégré avec un autre AOP pour obtenir un signal triangulaire constant. Ce dernier est ensuite comparé avec la tension variable de CNA pour obtenir un carré où la largeur d'impulsion varie. Ce signal est envoyé comme commande à un étage de puissance, constitué de transistors sur le principe de montage Darlington. Le signal de puissance est transmis au moteur à courant continu.

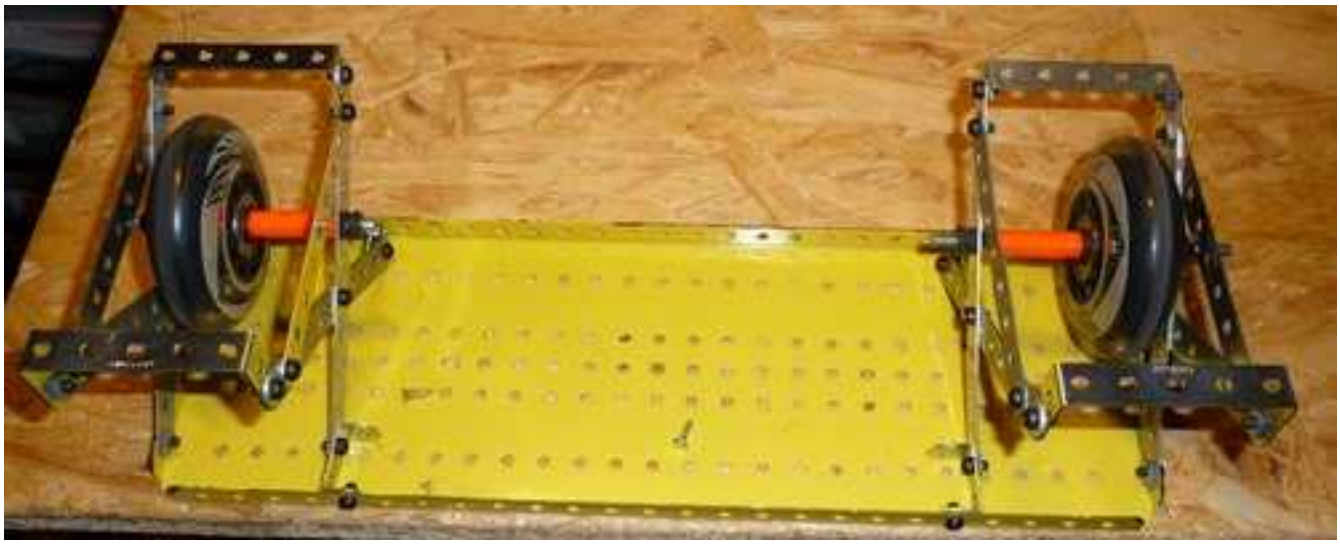
La puissance du moteur varie donc par rapport au mot binaire appliqué.

Composants utilisés :

Composant	Description
LM324N	Quad low power operational amplifiers
AD7302BN	Parallel Input Dual Voltage Output 8-bits DAC
ULN2003	High current Darlington transistor array
2N3904	NPN switching transistor
2N3055E	Complementary silicon Power Transitors
REL FRS1B	Relais 5V 1NO 1NC

6.2. Simulateur de surface

Ce système nous permet de simuler les déplacements du robot sans utiliser un espace de la taille d'un supermarché ;o) simplement en le posant sur un bureau.



7. Photos

photos

photos