RAPPORT BALISES 2003

MATZ Olivier

RIEHL Christophe

1 Cahier des charges – contraintes

Le but du système de balises de cette année est d'améliorer les balises de l'année précédente, afin de gagner en précision et en fiabilité.

a) Balises 2002

Les balises 2002 sont constituées d'émetteurs IR en tant que balises fixes émettant a tour de rôle un code modulé en 38kHz
Les récepteurs IR, au nombre de 32, et disposés en cercle permettaient de localiser la direction de chaque balise.
Les essais de triangulation ont été peu fructueux, du fait de la faible précision des mesures d'angles (pas de 360°/32)
Ce système était toutefois idéal pour localiser et aller vers un panier.
(voir rapport balises 2002 pour plus de détails)

b) Cahier des charges 2003

La principale contrainte demandée est une mesure en absolu de la position du robot (X,Y, angle), avec une précision raisonnable (exploitable plustot)
A cela s ajoutent les contraintes du règlement.

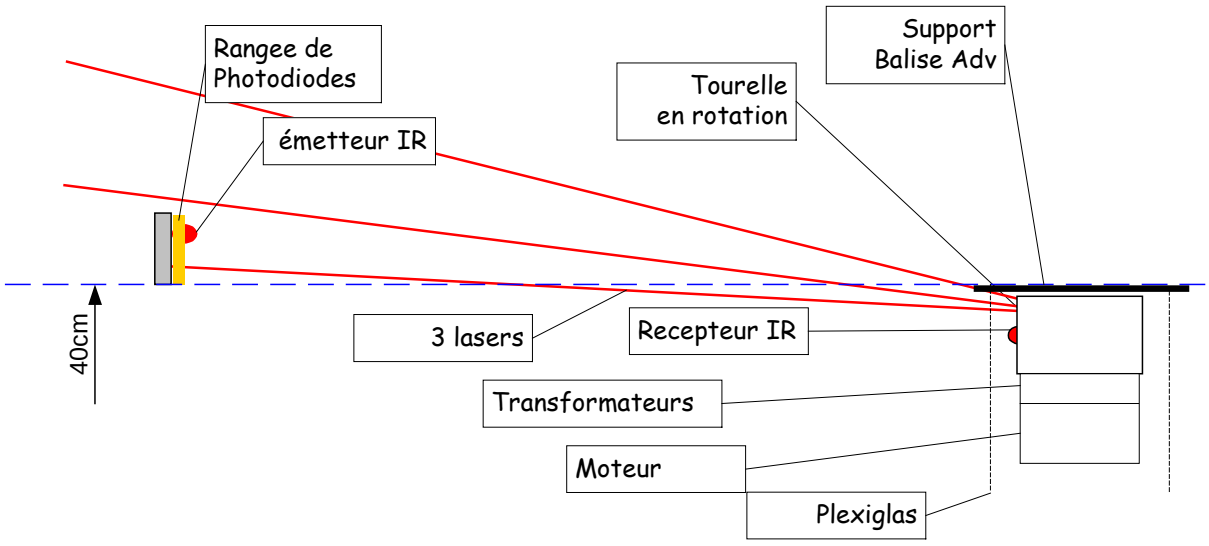
- 4 Balises compactes et autonomes (8*8*8cm) placées a 40cm de hauteur
- 1 Balise placée sur le robot adverse, compacte et autonome
- Un système (tourelle) placé sur notre robot en dessous de 40cm
- Le support de balise adverse ne doit pas gêner le repérage
- Distance mini et maximum de détection fiable : 20cm a 3m70
- Erreur de positionnement : 15cm max
- Mesure rapide (100ms max)
- Immunité aux divers parasites (électromagnetiques, IR, lumières diverses, autres lasers !!)
- Fonctionnement très fiable : jamais de position invalide en sortie
- Obiwan Kenoby

2 Principe

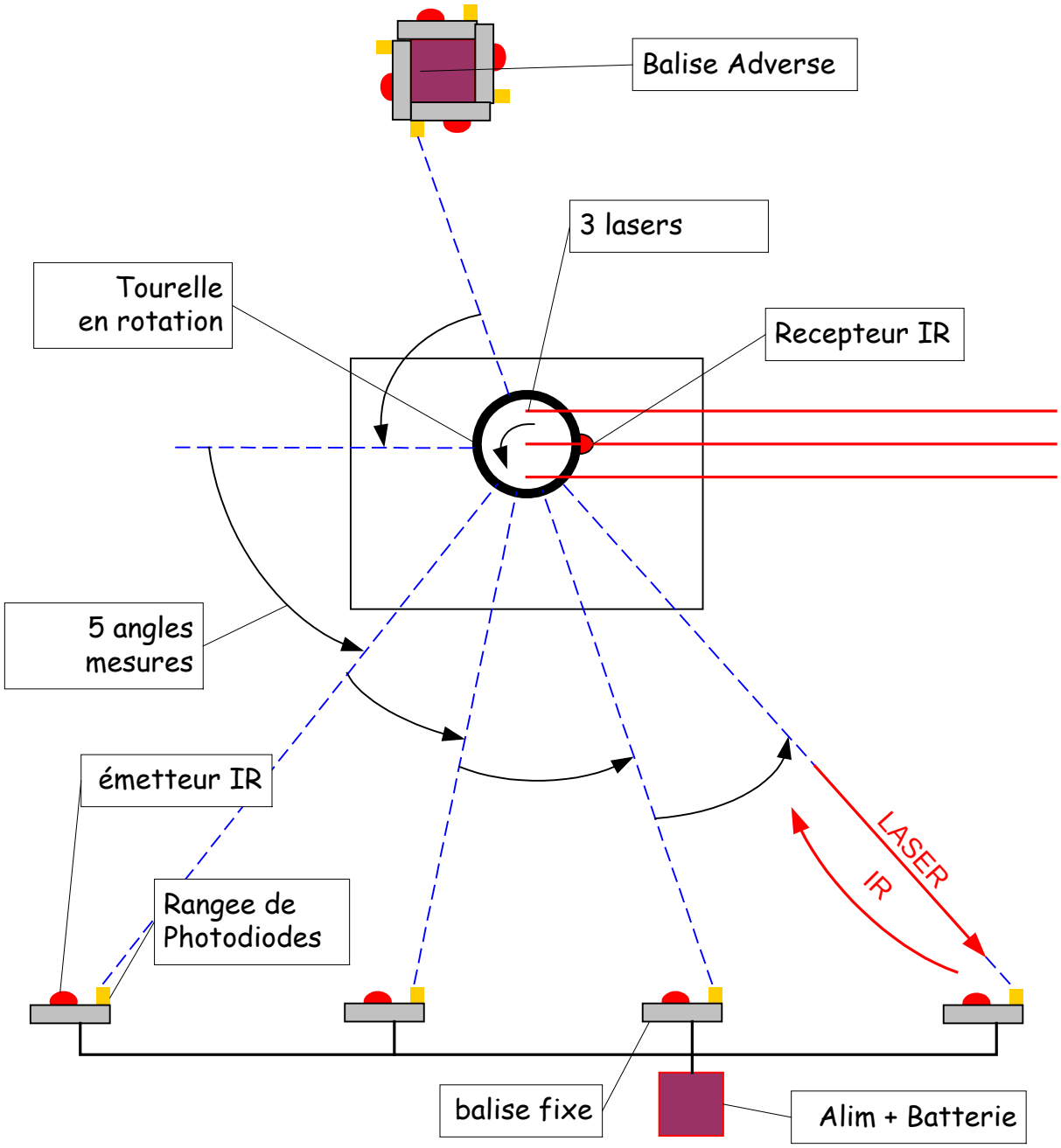
a) Généralités

Le principe adopté est un laser tournant sur le robot selon un axe vertical et émettant horizontalement.
Chaque balise captant le passage du laser peut donc déduire un angle grâce a la mesure du temps de passage.
Les balises fixes peuvent donc calculer la position du robot, et renvoyer celle ci, mais au fur et a mesure de l'avancement du projet nous avons préféré renvoyer un code pour chaque balise, et donc faire les calculs de triangulation uniquement dans le robot.
Quand une balise est touchée, elle émet une trame qui l'identifie, grâce a des diodes IR.
Le robot reçoit cette trame et en déduit l'angle ou se trouve cette balise, pour ensuite effectuer la triangulation.
Pour la balise adverse, le principe est le même a part que la fréquence est différente, et on inclut une information de distance.
Pour mesurer cette distance, on utilise 2 lasers parallèles et on mesure le temps de passage de chacun sur la rangée de photodiodes de la balise.

Vue de coté



Vue de haut


Balise Adverse
3 lasers
Tourelle
en rotation
Recepteur IR
5 angles
mesures
émetteur IR
Rangee de
Photodiodes
LASER
IR
balise fixe
Alim + Batterie

b) Tourelle

La vitesse de rotation de la tourelle est d'environ 20tr/sec (vitesse asservie).
Elle est construite sur une base de tête de lecture de magnétoscope, dont on utilise le moteur brushless et les transformateurs entre le rotor et le stator.
On y a monté 3 lasers parallèles si on regarde de dessus, et ayant des inclinaisons différentes si on regarde de coté.
On trouve également une carte réalisant l'alim de ces lasers et comportant les 2 récepteurs IR (30 et 56kHz). Cela permet d'avoir toujours les récepteurs pointés vers la balise dont on reçoit la trame.
Les liaisons entre cette carte et le robots passent a travers 3 transformateurs dont le primaire et le secondaire sont situés sur le rotor et le stator.

c) Liaison laser

Les lasers ne sont pas modulés, mais comme ils passent assez rapidement sur les photodiodes on fait un filtrage passe haut sur chaque diode.
En outre, on utilise des filtres numériques basés sur le nombre de détections simultanées : chaque balise ayant une rangée de 18 photodiodes, un laser ne peut en déclencher 3 côte à côte au maximum, on utilise donc ce phénomène pour filtrer les parasites agissant sur plusieurs photodiodes disjointes, ce qui est très éfficace.
De plus le laser n étant pas diffusé, la lumière reçue par une photodiode est assez conséquente, donc le seuil de détection peut être reglé assez haut, ce qui eliminé encore des parasites.

d) Liaison IR

La liaison de retour utilise des diodes IR (8 par balise) pour retransmettre les trames de chaque balise au robot.
Quand une trame est émise, les 4 cartes balises émettent simultanément, cela permet d'atténuer les différences de réception entre balises (le CAG des récepteurs IR faisait que l'on ne voyait pas la balise la plus éloignée)
La modulation des balises fixes est a 56kHz, la balise adverse fonctionne a 30kHz. Cela permet des trames plus courtes pour les balises fixes, dont l'enchaînement peut être assez rapide.
D'ailleurs si le laser touche la balise suivante alors que la trame n est pas finie, cette balise sera ignorée.

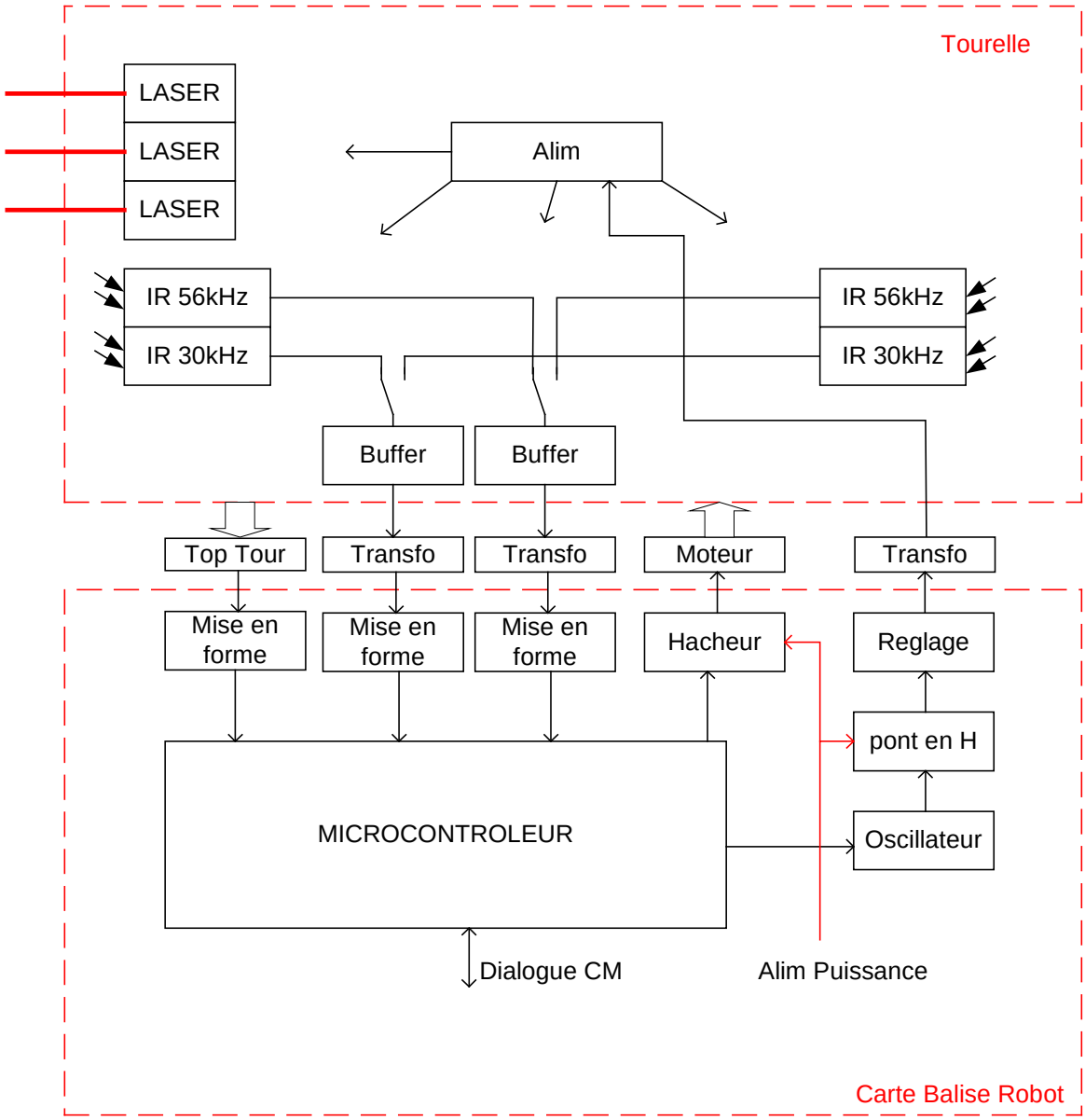
Une trame de balise fixe se compose de 4 bits : Start, MSB, LSB, stop
Le bit de Start est a 1, le stop a 0, et les 2 bits utiles codent le numéro de balise fixe. Chaque bit dure 10 périodes de modulation (179us)

Pour la balise adv, la trame contient un seul état haut qui dure d'autant plus longtemps que la balise est proche.

e) Le traitement

Le traitement, outre les filtres, est un calcul trigonométrique effectué sur 3 angles (3 balises, la 4eme est ignorée si on en reçoit 4).
On en déduit la position X, Y, thêta du robot sur le terrain.

3 Carte balise contenue dans le rack du robot



Sur cette carte qui dessert la tourelle, on trouve un ATMEGA163, un oscillateur et un pont de puissance destinés a alimenter l'électronique de la tourelle, une alim pour le moteur brushless de tourelle, et enfin des étages de traitement des capteurs : top tour et les 2 capteurs IR 30 et 56kHz.

a) Alim tourelle

La tourelle étant alimentée au travers d'un TRANSFORMATEUR, il faut une alim alternative.
Elle est constituée d'un oscillateur (U2) dont on peut ajuster la fréquence entre 500kHz et 1MHz. Une fréquence plus basse peut brouiller les récepteurs IR ou bien faire saturer le transfo.
Cet oscillateur sort 2 signaux carrés en opposition de phase. Ces 2 sorties peuvent être forcées par un signal venant du contrôleur. On peut donc éteindre l'alim de la tourelle afin de couper le laser quand il n est pas nécessaire, ou pour des raisons de sécurité, si le uC remarque que la tourelle ne tourne pas. En mode programmation, les sorties étant en haute impédance, R15 permet d'éteindre la tourelle.
Les 2 sorties de l'oscillateur pilotent des interfaces de puissance rapides, U4 et U5 qui sont en fait des drivers de MOSFET. Ce choix n'est pas très judicieux, car ils sont un peu limites au niveau puissance et temps de commutation, mais cela à fonctionné. Par contre, contrairement au schéma, il ne faut surtout pas utiliser 2 amplis en parallèle, car, leur seuil n'étant pas exactement le même, l'un peut commuter avant l'autre, et on se retrouve en situation de court circuit! Sur la dernière carte utilisée ces pistes on donc été coupées. Il faut absolument trouver des composants de remplacement pour l'année prochaine.
En sortie, on trouve un réseau RC limitant le courant et supprimant une éventuelle composante continue.

b) Alim moteur

Le moteur étant du type triphasé synchrone, et possédant sa propre électronique de commutation, on l'alimente normalement avec une tension continue.
U12 est un régulateur de tension réglable servant à ajuster la vitesse
Ce régulateur peut être désactivé par le uC via T1 & T2 qui forcent la patte ADJ du régulateur a 0V, d'ou 1,25V en sortie, ce qui empêche le moteur de tourner. Ce montage a 2 transistors sert a ce que la tourelle s arrête en cas d'absence d'alim logique, et R16 devait bloquer le fonctionnement pendant la programmation du contrôleur, mais a été mise a VCC au lieu de la masse (ah ces sabotageurs….)
Apres quelques essais, il s est avéré que la vitesse de la tourelle pouvait fluctuer, et nous avons donc décidé de l'asservir, ce qui a été fait a l'aide du top tour, mais alors comment commander le moteur de façon analogique ? Eh ben on a appliqué une PWM (de l'ordre du kilohertz) sur la commande moteur, c est vraiment pas propre du tout comme solution, car l'électronique du moteur est alimentée par intermittence, mais ça marche !
En plus olivier a programmé un super asservissement, la vitesse est très stable.
On peut envisager l'an prochain de commander directement les 3 phases du moteur par des sinus PWM, a voir….

c) Traitement des capteurs IR

Les informations des 2 capteurs IR passant par les fameux transformateurs, arrivent sous forme de la dérivée du signal. Il est donc nécessaire de remettre en forme ce signal pour pouvoir l'exploiter correctement avec le uC. Astuce de sioux, on utilise un NE555 (un par voie, U10 et U11). Ce circuit est câblé ici en comparateur a hystérésis (1/3 et 2/3 de VCC), ou

plustot comme comparateur a fenêtre suivi d'une bascule RS, mais ça revient au même, voir DS du NE555.
Une des bornes du transfo est reliée a une référence de tension (ajustable, car il y a des dissymétries), l'autre borne du transfo est reliée a une résistance de charge (variable pour ajuster l'amplitude) et aussi a l'entrée du 555.
Quand le signal d'origine du capteur passe de 0 a 1, une impulsion positive très courte est reçue et met la sortie du 555 a 0, et elle y reste car la tension repasse a VCC/2.
Pareil pour le passage de 1 a 0
Voilà a quoi ressemble le signal sortant d'un transformateur : dessin
A améliorer ici : le circuit du transformateur oscille, et les réglages sont délicats car après le pic principal, il y en a un plus faible, négatif.

d) Capteur top tour

Le capteur de tops tours subit le même type de traitement que les capteurs IR car il donne le même type d'information.
On a juste un amplificateur en plus car le niveau est trop juste pour atteindre les 1/3 – 2/3 de VCC

e) Divers

Bon, à part le contrôleur il reste pas grand chose à dire sur la carte, le découplage est vital ici, vu que des signaux HF côtoient des infos plus faibles venant des transformateurs.

4 Tourelle

a) Mécanique

La mécanique de la tourelle est un scanner de magnétoscope VHS 4 têtes (tête de scope + moteur + transfos – têtes de lecture + lasers + carte alimentée + lasers = tourelle de balise laser , c'est l'équation magique)
On a enlevé les têtes du tambour, puis M. DELAUNAY y a fraisé 3 rainures pour le logement des lasers avec des vis de maintien et un capot par dessus.
Un contrepoids de plomb, coulé dans le tambour assure l'équilibrage de tout le système.
Les lasers ont des inclinaisons (ajustées a coups de lime sur le corps des diodes laser) différentes pour qu'il y ait au moins un laser qui atteigne la cible quelle que soit la distance (20cm-4m)
Le tambour est en fait monté a l'en vers pour laisser de la place a la carte électronique, qui est tournée a son emplacement pour être équilibrée, et ne pas avoir de saillie sur le système (25 tr/sec c'est pas rien).
Le moteur est laissé tel quel, il a juste fallu identifier les connections.
La tourelle entière est placée sur un support ajustable fixe au rack elec.
Autour de la tourelle, un morceau de bouteille plastique sert à supporter le support de balise.

C'est une très bonne mécanique, idéale pour cette application, et M. DELAUNAY a fait du très bon travail, comme d'habitude.

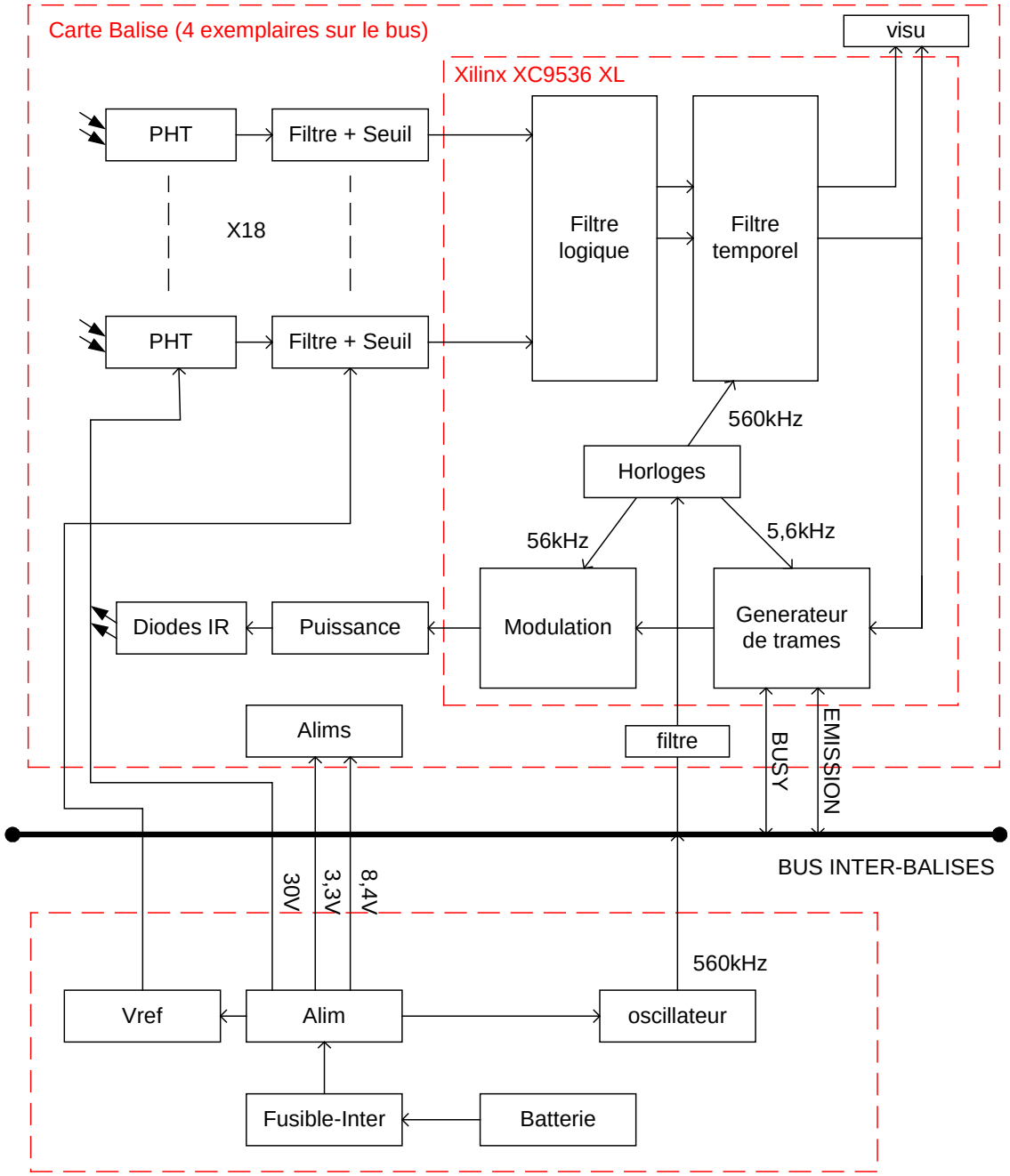
b) Électronique

On y trouve d'abord l'alim : un pont redresseur de 4 diodes rapides avec un filtrage. Les lasers sont juste alimentés a travers une résistance de limitation chacun (méthode douteuse, ils se dégradent avec le temps…) et une résistance pour ajuster l'intensité globale. Un régulateur fournit l'alim pour les récepteurs IR.

4 récepteurs IR étaient prévus, un couple 30 et 56 kHz dirigé dans le même sens que les lasers, et un vers l'arrière, avec un interrupteur pouvant sélectionner entre les fréquences, mais finalement on a utilise que les capteurs avant.
Les capteurs commandent des buffers qui envoient les impulsions sur les 2 transfos, et sur des leds de visu (c est joli la nuit…)

Cette carte est assez touffue, l'espace est limité, et surtout elle doit être a peu près équilibrée mécaniquement!!
(le 1er qui me trouve un autorouteur qui fait ça gagne un AVR)

5 Balises



Les 2 systèmes de balises (fixes et adverse) sont composées d'un pack d'accus de petites dimensions (7* AAA), d'une carte d'alim, et de 4 cartes balises dialoguant sur un bus et comportant chacune émetteurs, récepteurs, et traitement.

a) Carte d'alimentation – horloge

Le pack batteries étant de 8,4V étant de y trouve un régulateur a découpage sortant la tension pour la logique (3,3V), mais aussi, grâce a un 2ème enroulement sur la self, la tension de polarisation des photodiodes (~30V, max 20mA). Ici cette tension dérivée est obtenue par un fonctionnement de type boost, afin de maintenir la tension constante, il vaudrait mieux inverser le 2ème enroulement et lui bobiner plus de spires (10*44, dur dur….). Attention à ne pas dépasser les 40V (tension max photodiodes)
On trouve aussi sur cette carte un oscillateur (U2) servant a clocker les xilinx des cartes balises. La fréquence retenue est de 560kHz fréquence qui permet, par division d'obtenir les 56kHz d'émission IR, mais cette horloge sert aussi a échantillonner les signaux d'entrée, et comme le temps de passage du laser peut être très court, il faut s'assurer qu il sera toujours assez supérieur a cet échantillonnage, d'ou cette valeur élevée.
On trouve encore sur cette carte inter marche arrêt, fusible, 2 résistances de tirage pour le bus inter cartes (R4-R5) et enfin un potentiomètre ajustant la référence de tension de comparaison.

b) Carte balise

On a tout d'abord 18 récepteurs complets, comprenant une photodiode, suivie d'un filtre passe haut, puis d'un comparateur.
La photodiode est polarisée par une résistance a une tension élevée (Vpol = 30V) afin de ne pas saturer pour des lumières continues tout en gardant une sensibilité assez élevée (déterminée par R1)
Le filtre passe haut C1-R19 est dimensionné afin de laisser passer les impulsions laser, que la distance soit faible ou élevée, avec une vitesse de rotation mini de 10tr/sec.
Le 50/100Hz ambiant est bien filtré(voir chronogrammes relevés).
Le comparateur indique quand un pic dépasse Vref.

Les 18 récepteurs constituent donc une barrette de détection qui signale quand le laser passe sur la balise et à quel niveau.
Les photodiodes on été implantées légèrement en biais pour éviter les zones mortes entre elles.

Passons à l'autre schéma :
Derièrre chaque comparateur on trouve une résistance de pull up.
Les comparateurs a collecteurs ouverts ont été choisis pour plusieurs raisons : les xilinx sont alimentes en 3v3, et les comparateurs en 8,4V, et en plus ça évite un parasitage trop important de l'alim des AOP pendant la commutation.
D'ailleurs toutes les tensions arrivant a la partie analogique sont filtrées sinon on aurait du bruit venant des couplages (4m de câbles, commutations IR, logique, alim a découpage…)
Vref est aussi divisée pour éviter de transporter une tension trop faible sur plusieurs mètres.

Le xilinx XC9536 XL (3v3) a été choisi car c'est le moins cher ! (~4E) On y entre les 18 entrées, les 2 lignes de bus inter cartes (+ une autre pour l'horloge commune).
On avait prévu un oscillateur a quartz, mais il ne fonctionne pas directement sur le xil, on s'est en fait servi de son implantation pour mettre un filtre qui épure l'horloge reçue (100p/10K ?), car après qqes mètres d'une liaison non adaptée, l'horloge est assez crade (adapter la liaison ??)

Le xilinx est également relié au connecteur de programmation, prolongé par un connecteur de test sortant 4 signaux pour le debug (très pratique !!).
On trouve aussi 3 leds de signalisation et la commande de puissance des leds IR qui se fait en 2 puissances par des transistors mos.
Ce schéma est a améliorer : pdt la programmation les mos flottent => res de tirage obligatoire, et il serait bon de limiter le courant constant pouvant accidentellement circuler dans les leds (condensateur ??) Le réglage du courant était utile, pas les 2 niveaux…

Par contre le xilinx était un peu étroit pour toutes les fonctions nécessaires, un 9572 aurait été confortable, mais on ne le trouve pas en 44pins.

c) Software VHDL

Il contient pleins de modules :
-horloges : diviseurs générant une horloge d'échantillonnage(560kHz), une horloge modulation(56) et une horloge bit de trame (5,6kHz)
-capteurs : équations logiques qui permettent d'éliminer les cas interdits (2 récepteurs distants activés par ex.)
-timers : compteurs qui permettent la validation d'une détection. Par ex si une détection invalide suit une valide (ou inversement), l'ensemble est invalide. Sans ce filtre, le précédant ne sert a rien, car lors d'une erreur les 2 récepteurs ne s'activent pas forcément en même temps !
Pour pouvoir dévalider par après une détection valide, on retarde en fait la sortie et on annule si entre temps il y a une erreur.
-gener_trame : module générant la trame (personne ne s'en doutait…) en fonction du numéro de balise, et des qu'il y a détection valide.
-carte : module avec les déclarations générales, et qqes fonctions qui devraient être un peu mieux rangées
-testbench : banc de test de simulation, qui produit quelques signaux de test

La liaison avec les autres cartes est faite sur 2 lignes en collecteurs ouverts(à VCC) occupé est mis a un par la balise qui émet sa trame pour empêcher les autres de commencer aussi une trame, et emission_ext fait émettre les autres balises en même temps, pour les problèmes de CAG récepteurs IR. Ces lignes étant des collecteurs ouverts sont donc bidirectionnelles, entrées en temps normal et sorties quand on génere une trame.

Tout ce code est vraiment très optimisé (crade quoi), car le xilinx est vraiment juste. Ce n'est pas écrit de façon synchrone, on a parfois des complications a cause de ça, surtout si l'alim n'est pas propre (pics de commutation des IR par ex…)

Pour le module balises adv, quelques changements sont a noter : à la première détection, un compteur démarre a partir de 0, à la 2 eme il change de sens et son horloge ralentit, et on émet des IR tant qu il n est pas a 0, donc on émet une durée proportionnelle a la durée entre les 2 lasers (mesure de distance approx)

6 Software

a) Non ce n'est pas le bordel

Ce code est fait très proprement. A quelques détails près. Ca a été fait un peu dans l'urgence, comme chaque année, et comme toutes les équipes.

Voici la liste des modules importants :

- main : Ben les inits de départ et la boucle principale.
- tourelle : gestion de la rotation de la tourelle et de l'allumage des lasers.
- traite : décodage des signaux IR, calcul de la position
- rtc, SPIpfd*, ap_uart, printf_P, utils, attente : les grands classiques des programmes du robot. On retrouve ces modules partout, reportez-vous au rapport sur le soft général (comment ça y'en a pas ?). Bon j'en parlerai vaguement plus loin.

Il n'y a pas tant de choses que ça, on va essayer de boucler ça vite fait bien fait.

b) L'algorithme

Cahier des charges

Tof aime bien mettre des cahier des charges alors je lui en met un ça lui fera surement vachement plaisir.

- Décoder les signaux

Ceci peut être fait soit en mémorisant les signaux et en faisant un décodage après en une seconde passe, soit en décodant à la volée les signaux reçus.

- Avoir une mesure de l'angle la plus précise possible pour chaque balise

La mesure d'angle va se rapporter à une mesure de temps, nous l'avons vu, car la tourelle retourne un TOP à chaque tour. Ce point implique deux choses : Le décodage doit être rapide (peu d'instructions) afin de ne pas ralentir la mesure du temps sur les timers, ou tout du moins ce temps de traitement doit être connu et constant. En second lieu, la vitesse de la tourelle doit être constante et connue (asservie en fait). Bon là je fais comme si c'était évident qu'il fallait l'asservir mais en pratique ça a mis quelques semaines avant qu'on ne se décide.

- Retourner une position juste sinon ne rien retourner

C'est pas trivial. Il s'agit de recenser un maximum de cas que l'on considérera comme des cas d'erreurs. On essayera de discuter de ça un peu plus bas.

Une première idée

On a vu (tof l'a expliqué non ?) que les sorties des capteurs IR sont reliés aux broches d'interruptions externes INT0 et INT1 du microcontrôleur. Il est possible de configurer ce dernier pour qu'une routine d'interruption s'exécute sur n'importe quel front (je rappelle pour ceux qui ne suivent pas qu'il existe deux fronts, le montant et le descendant... je ne vous ferais pas l'affront de croire que vous ne le saviez pas). L'idée géniale est d'avoir un timer qui tourne en permanence et à chaque changement d'état sur une broche d'interruption (INT0 = balises fixes et INT1 = balise adverse) la valeur de ce timer est sauvegardée dans un tableau. Lorsque le tableau est plein (la taille doit être adaptée), on le parcourt et on décode les trammes.

Lorsque l'on a récupéré les angles de chaque balise, on a facilement la position du robot. En fait, trois balises suffisent. Nous verrons dans au paragraphe suivant l'équation qui permet d'obtenir la position à partir des angles des balises relatifs au robot.

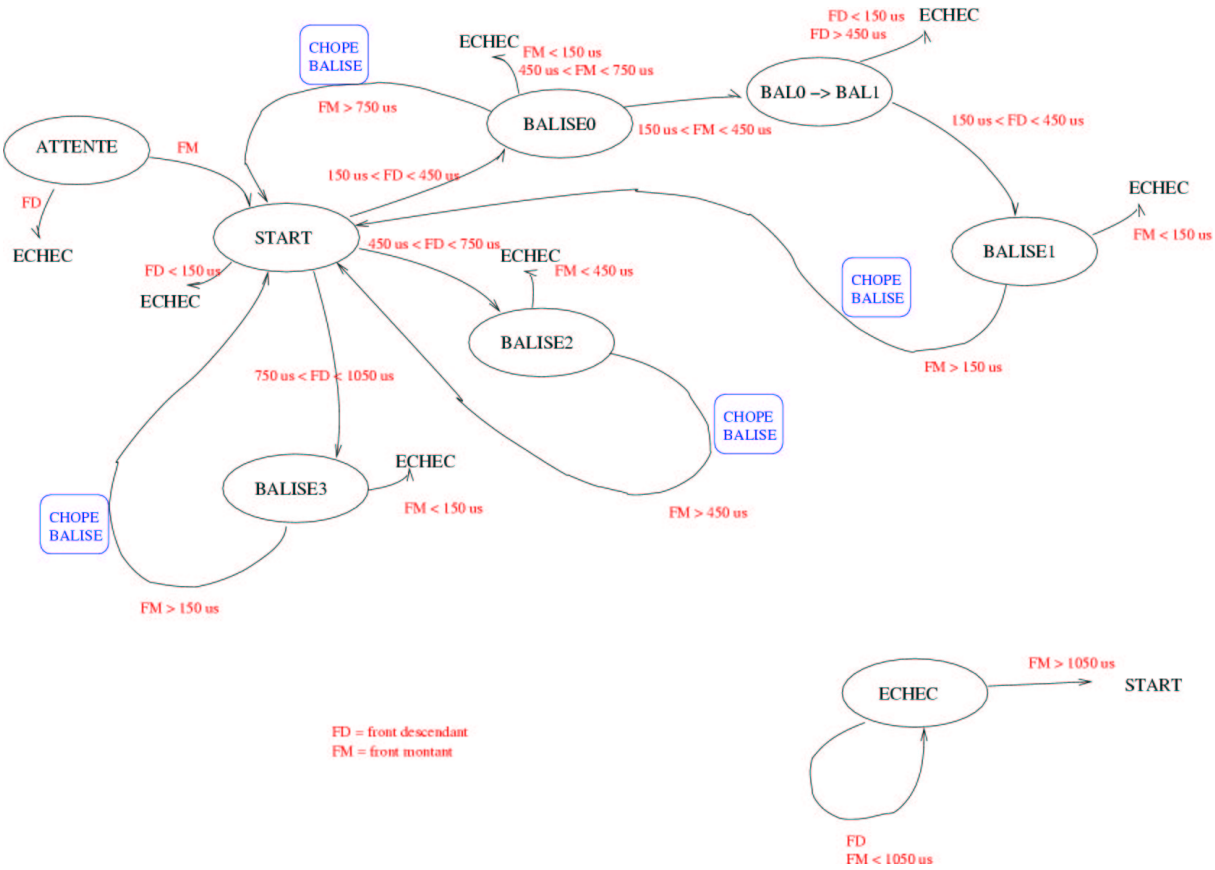
Revenons à notre idée. Pourquoi il y a mieux ? Déjà car c'est un peu lent, on est obligé d'attendre que le tableau soit plein (en plus il faut qu'il soit suffisamment grand pour contenir a coup sûr les trammes des 4 balises). Et une fois qu'il est plein, il faut décoder, puis faire le calcul. C'est un fonctionnement un peu trop séquentiel à mon goût. Y'a aussi une autre raison : j'ai commencé à réflechir au code, et je me suis rendu compte que j'étais mal parti. Bref, je suis reviendu au point de départ.

Une deuxième idée

Cette deuxième idée met en place un décodage des trammes à la volée dans la fonction d'interruption. Pour chaque broche d'interruption, on a un séquenceur : il est d'abord en mode attente, puis en fonction du front qu'il reçoit et du moment auquel il arrive, le séquenceur détermine si une balise a été captée, si on doit attendre la suite du signal ou si le signal n'est pas conforme (dans ce cas il y a une erreur et on attend un "trou" pour se resynchroniser).

Voici un petit schéma qui représente bien le fonctionnement de l'automate. A chaque front, l'automate décide dans quel état passer. Les temps qui figurent sur le schéma ressemblent aux temps utilisés dans la version finale (300 us par bit), mais ils ne sont là qu'à titre indicatif. Il est intéressant de comparer l'automate avec les trammes séries que l'on reçoit : "1000" pour la balise 0, "1010" pour la balise 1, "1100" pour la balise 2 et "1110" pour la balise 3.

Pour la balise adverse, c'est pareil mais en plus simple.



Il reste encore a mémoriser le temps à chaque start pour connaître le moment précis du début de l'émission, ce qui n'est pas trop difficile. Une fois qu'on a les temps pour chaque balise, ils sont convertis en angles relatifs au robot, on filtre un peu (du genre "tiens la balise 2 était à 94 degrés il y a 100 ms et maintenant elle est à 200 degrés" => invalidons la mesure). Puis on calcule la position (x,y,a), on refiltre un peu (histoire que la position du robot ne bouge pas de 25 mètres entre deux mesures, d'ailleurs il n'est plus sur le terrain), et le tour est joué.

Des détails dans la partie Implémentation.

Je vous remet encore une petite idée ?

Allez c'est la fête c'est ma tournée d'idées. Il y a un défaut majeur dans le code des balises, mis à part le fait que c'est pas super propre.

Le problème est que l'horloge de l'automate est synchrone aux transitions. Ce n'est pas très grâve à première vue mais c'est vite gênant : Pour détecter un "trou" (absence de signal IR) il faut attendre le front montant suivant. Pas cool.

D'ailleurs cet automate pourrait être bien mieux défini. Déjà je pense que l'automate doit être dans une boucle a part (dans RTC par exemple). Et il faut pouvoir ajouter en plus des deux tokens "front descendant" et "front montant" (Yves ?) un token "aucun front". On sait alors qu'aucun front pendant plus de 1ms signifie un "trou". Bon ça a pas l'air révolutionnaire comme ça, mais croyez-moi, ça évitera plein de saloperies dans le code.

c) Le code

J'ai préférer appeler cette section "le code" plutôt que "l'implémentation", car au moins on sait qu'un code c'est censé être difficile à déchiffrer.

Deux petites généralités avant de commencer :

- Il y a beaucoup de variables globales, pour la simple et bonne raison que des variables locales ça va dans la pile ou dans les registres, et sont créées à chaque appel de la fonction. C'est donc plus lent car le compilo doit sauvegarder le contexte, et on veut que l'interruption soit traitée le plus vite possible : rappelez-vous, la mesure du temps doit être précise. Enfin, je dis ça, c'est quand même un peu con d'avoir un code qui fait la mesure du temps à la microseconde près alors que le capteur infrarouge a une incertitude bien plus grande.

- le mot clé "volatile" signifie pour une variable que sa valeur est toujours à jour dans son emplacement mémoire. En effet, par exemple, il arrive que lors d'un calcul utilisant et modifiant une variable globale normale, le compilo souhaite optimiser et stocke les résultats intermédiaires de cette variable dans un registre. Parfois, dans une boucle, certaines variables ne sont utilisées qu'en tant que registre. Il y a de fortes chances que le programme suivant ne fonctionne pas :

```
int a;

SIGNAL(SIG_INTERRUPT0) {
  lcd_printf("%d\n",a);
}

int main() {
  while(1)
    debug_printf("%d\n",a);
}
```

La variable a doit être déclarée par :

volatile int a;

Les modules standards

- utils : c'est un module qui permet un accès plus évolué à la mémoire EEPROM. On y place aussi les macros ou fonctions inline de bas niveau. Ne sert à rien sur cette carte (mais que fait-il là alors ?).

- ap_uart : transmission et réception de caractères sur la liaison série. La liaison série pouvait être aiguillée vers le PC grâce au Xilinx, pour débugger.

- rtc : une interruption qui a lieu toutes les 256 us. Tiens encore un module qui n'a servi a rien.

- SPIpfdS : SPI, protocole full-duplex, coté esclave. Des fonctions de bas niveau pour l'émission et la réception de données vers la carte mère (le maaaaîître).

- SPIpfd : Les différentes commandes et statuts qui constituent les dialogues avec la carte mère.

- attente : Ben des fonctions de tempo, tout ce qu'il y a de plus bête. Ce sont des boucles dont la longueur est calculée pour mesurer un temps précis. Attention tout de même le processeur est monopolisé pendant tout ce temps.

- printf_P : Implémentation du printf. Il y a uart_printf qui envoie les données vers l'uart (fonction bloquante, lente, mais simple). Et il y a aussi debug_printf qui n'est pas bloquant (car tout est écrit dans un buffer et émis sous interruptions). Si debug_printf est appelé alors que le buffer précédent n'est pas vide, il est ignoré.

- debug : contient des fonctions appelées par le debug_printf. Il y a 3 fonctions, une pour emettre un caractère (il est en fait copié dans un buffer), une pour emettre le buffer (appelée à la fin du debug_printf) et une pour savoir si un buffer est en cours d'émission ou non.

Le module tourelle

Ce module sert à asservir la vitesse de la tourelle. L'asservissement est plutôt mou et lent, mais il est assez stable. On n'a de toute façon pas besoin d'un temps de monté rapide. Lorsque l'on allume le robot, même avec la commande à fond, le moteur met un certain temps à atteindre sa vitesse de croisière.

Les lignes suivantes servent à générer un PWM lent sur le timer 2. En modifiant la valeur du registre OCR2 (Output Compare Register), on change la valeur du PWM.

```
INTERRUPT(SIG_OUTPUT_COMPARE2)
{
  MOTEUROFF();
}

INTERRUPT(SIG_OVERFLOW2)
{
  MOTEURON();
}
```

On se sert du timer 1 pour :
- mesurer la vitesse de la tourelle
- declencher l'asservissement de cette dernière toutes les 65 ms
- mesurer les temps des détection des balises (on va le voir plus bas)

Ce timer 16 bits tourne en boucle à 1Mhz (incrémentation toutes les microsecondes).

Voici comment fonctionne la mesure de vitesse de la tourelle :
Le Input Capture Port est relié au top tour de la tourelle. A chaque top tour, la fonction SIGNAL(SIG_INPUT_CAPTURE1) est appelée. Elle sauvegarde la valeur du timer 1 à ce

moment ansi que la valeur du timer 1 lors du précédent top tour. Il suffit de faire la différence de ces deux valeurs et on a le temps que met la tourelle pour faire un tour (il est de l'ordre de 25 ms à fond). Enfin, il faut s'assurer que la tourelle ne tourne pas moins vite que 65ms, sinon notre différence de valeur de timer sur 16 bits ne veut plus rien dire. C'est le rôle de la variable temps65536us qui est incrémentée à chaque top tour et remise à zero à chaque overflow. Si lors de l'appel de l'asservissement cette variable vaut 0, alors la tourelle tourne trop lentement, donc on met la gomme.

Si le moteur est trop lent pendant plusieurs appels de l'asservissement, on en profite pour éteindre le laser pour des raisons de sécurité.

Sinon l'asservissement est une sorte de proportionnel-intégral un peu batard. Je vais pas détailler trop ça, il faut juste savoir que ça marche (promis l'année prochaine je le fais mieux).

Le module traite

De quoi traite ce module ? hum... Il fait le gros du boulot, décodage et calcul.

En fait rien de dur. C'est long mais pas difficle. Il s'agit de la "traduction" de l'automate vu dans la partie précédente. Sauf qu'il positionne les variables utlisées dans le main. C'est là que c'est pas propre (oui le reste passait encore). Entre le tempsMachinActuel ou tempsTrucRec2PrecedentStocke, j'avoue qu'on s'y perd un peu... euh... ok beaucoup.

Déjà le principe est compris je pense, il s'agit de retourner le temps qui correspond à la balise détectée. Mais comme on l'a vu dans le partie Algorithme, il n'est pas possible de détecter un "trou" tant que le front montant suivant n'est pas survenu. En clair, impossible de valider la détection de la balise n tant qu'on a pas chopé le front montant du bit de start de la balise n+1.

On se retrouve à fournir la valeur de certaines balises en retard, et il est alors impossible de déduire l'angle de la balise car on ne sait pas par rapport à quelles valeurs du topTour elle est définie. Bref, on doit fournir des infos supplémentaires au main (la valeur du topTour qui correspond à la dernière balise détectée, ...et d'autre trucs dont plusieurs ne doivent plus servir à rien). Résultat : cette partie est imbittable, démerdez vous. ;)

Ce qui aurait pu être bien, c'est la réinitialisation du timer à chaque top tour (on executerait l'asservissement de la tourelle par la rtc du timer 0), de façon à ce que la référence ne bouge pas à chaque tour. Le problème c'est que la mesure des temps entre deux fronts ne pourrait plus se faire par une simple soustraction de la valeur actuelle du timer par la valeur précédente (en modulo 16 bits, comme le timer).

Dans ce module il y a également une fonction qui calcule la position du robot en fonction de l'angle des balises. Ça marche très bien. C'est une équation générée par crazyfred et lebomb avec Maple. Elle a été optimisée à fond et j'ai été épaté par le peu de temps que mettait le uC pour faire ce calcul (je sais plus exactement ais j'aurais dit 1 ou 2 ms).

Le module main

Voici son fonctionnement global :

initialisations
faire à l'infini
 | si une balise a été récupérée
 | | calculer son angle
 | si on a 3 mesures récentes, fiables, et tout et tout
 | | lancer le calcul de la position

C'est tout con, mais il n'empêche que c'est pas évident à trouver quand on a que le code sous les yeux. Tout comme le module traite, c'est vrai que c'est pas du grand art en propreté de code. Heureusement j'ai toujours l'excuse "on était pressé", et d'ailleurs c'est loin d'être faux.

Conclusion

Donc, à améliorer, la détection des "trous" en plaçant le séquenceur sous rtc aussi. Eventuellement trouver un uC qui a un timer 16 bits en plus ou utiliser un 8 bits + un compteur. Proprifier le tout, Ajouter un peu de sel et c'est parfait.

Le principe est là, il faut maintenant peaufiner l'implémentation
(c'est marrant je crois que j'ai dû conclure sur un truc un peu pareil pour la carte son).

7 Conclusion

a) Qu il est bien ce système ! (louanges …)

Ce système a donne satisfaction par sa fiabilité (même si on l'a pas utilisé en match). Ses points forts sont :

- Précision suffisante
- Fiabilité très bonne (on a pas vérifié lors du match de 16eme,mais je pense qu elles fonctionnaient)
- Facilite d'utilisation, pas de réglages compliqués, a part à la fabrication (lasers)
- Position presque instantanée, a 25 Hz.

En gros c'est un très bon système, fiable même si il est complexe.
Je pense qu'en 2003, aucune autre équipe n'avait un repérage absolu aussi efficace !
L'an prochain il va sûrement être très utile car il y aura des objectifs fixes a atteindre avec assez de précision (apparemment le règlement choisi est un rugby).

b) Qu'il est pas bien ce système ! (Améliorations – Défauts)

On peut encore améliorer ce système l'an prochain, car il y a encore quelques défauts

- Précision de 10cm pas encore satisfaisante, probablement la faute à la prise en compte des angles dans les calculs
- La transmission IR est lente, on pourrait la remplacer par de la HF. Avec de la HF, la tourelle est simplifiée (récepteurs fixes), et on peut transmettre beaucoup plus d'infos. Le traitement peut ensuite être fait dans les balises, en ajoutant un uC (un seul pour 4 balises), qui s'occuperait du calcul, codage et transmission.Ca permet de décharger les xilinx et aussi d'avoir la position pour debug depuis une partie fixe (bal fixes). On peut aussi estimer les distances et revérifier la cohérence générale.
- les lasers devraient être asservis en luminosité, et pas seulement ajustés en courant
- le moteur peut être commandé directement en triphasé, certaines têtes ont peut être aussi un moteur CC
- le filtrage de la mise en forme des transfos est à améliorer
- le pont en H d'alim tourelle chauffe trop
- Les filtres temporels ne sont pas encore prévus pour un autre système de laser balayant (adverse) : Le principe est d'ignorer tous les signaux qui ne sont pas cycliques exactement à la fréquence de rotation (s'arranger pour avoir 2 freqs commutables)
- La liaison IR peut servir à faire une balise du même type que l'an dernier en cas d'urgence (panne par ex, ou manque de temps), mais ça prend aussi du temps
- Pb de liaison IR : la balise 3 est assez mal reçue, probablement car son impulsion est longue (récepteur pas prévu pour ça !)
- Bon on a des balises qui fonctionnent, il faudrait penser à les utiliser !!

c) Remerciements

Merci à l'ENSEIRB qui nous donne les moyens de réaliser nos projets.

Un grand merci au personnel de l'ENSEIRB qui nous soutient :

M. DELAUNAY pour son travail toujours de très bonne qualité, M. LEGALL, notre bien aimé magasinier, M. MICOULEAU, M. LALANDE, et nos profs de parfois tolérer nos petites absences.

Nos sponsors, qui sont également des partenaires très precieux, sont : GERTRUDE, ALTARES, l'UNIVERSITE DE BORDEAUX I, MICRO SYSTEMS, et le CREDIT MUTUEL

Merci à M. LAUNAY, et M. HOCHENOFFER, de l'ENSAM, pour leur contribution à la mécanique.

Les membres du club awant travaillé sur les balises sont :

Olivier MATZ pour la partie code
Olivier CAMBON & Christophe RIEHL pour l'électronique
Olivier CAMBON et Frédéric LOCHON pour la partie calculs

8 Annexes

Les schémas sont dans des fichers séparés, ps ou pdf