MANUEL UTILISATION ROBOT MOBILE

Ph. GRARE

GMP - IUT TROYES

TABLE DES MATIERES

1. INTRODUCTION

Le projet tourne autour d'un jeu, fondé sur le règlement de la coupe de France de Robotique, qui confronte deux équipes de 6 à 8 étudiants GMP. L'objectif est de faire des actions, par un robot mobile, imposées par un cahier des charges fourni par plusieurs clients.

L'association Gem6 domiciliée à l'IUT GMP de TROYES permet l'achat de matériel aidant à l'amélioration du robot d'année en année. Ce projet fait appel à beaucoup de compétences GMP telles que la conception, la

fabrication, le câblage et la programmation. Le bureau de l'association doit être composé d'un Président, d'un secrétaire et d'un trésorier. Un vice-président peut être nommé par le Président.

La mécanique du robot RM est composée de deux parties, la partie avant, pour manipuler des objets et la partie arrière qui comprend le châssis avec les deux moteurs des roues et les codeurs, la partie commande qui contient l'Automate Programmable Industriel (API et les cartes de commande à relais des moteurs de la partie avant.

La programmation de l'API du robot est composée du programme principal, nommé FB16 et lancé par FC1. Ce dernier bloc lance des fonctions système, la FB110 de déplacements du robot (déplacement vers un point, pivotement d'un angle, recul, orientation vers un point ou d'un angle…) la FB114 de commande des moteurs de la PA, la FB120 de localisation du RM sur la table.

La fonction système FC100 est appelée lorsque le bouton RAZ est enfoncé, il permet l'initialisation des variables et la définition des coordonnées des points de la trajectoire.

La programmation du système de vision est faite sur une carte Arduino qui dialogue avec une caméra PIXY. Ce système peut être utilisé pour repérer la position des objets (par leur couleur ou leur taille) sur la table.

Il est possible de simuler les déplacements du RM sans l'API. Les coordonnées des déplacements s'enregistrent par la fonction système FB140. Dans une feuille Excel, il est possible de visualiser les déplacements faits par le robot virtuel ou réel. Dans cette feuille, on trouve les différentes zones de la table, le dessin des objets de jeux, le dessin du RM, le dessin du RA et du RS, la trajectoire réalisée et des boutons pour déplacer le RM, le RS et le RA. Des objets peuvent se positionner selon des bits dont leurs états sont copiés à partir de la VAT traj. Il est possible aussi d'y définir la position des points, il suffira de copier le tableau de points dans la FC100 Il est fortement conseillé de simuler le programme avant de le charger dans la mémoire de l'API avant de placer le RM sur la table.

2. LA PARTIE AVANT

La partie avant doit permettre de réaliser les fonctions pour manipuler les pièces de jeux. La partie avant comporte deux roues folles et est conçue et réalisée pour remporter le maximum de points. Les différentes fonctions sont assurées par des dispositifs manœuvrés par des petits motoréducteurs pris dans des tournevis IKEA. Le règlement précise que le robot doit posséder un évitement du robot adverse (RA), des capteurs opto-électriques sont placés à l'avant et permettent la détection du RA.

LA STRUCTURE EN PROFILÉS BOSCH

Des profilés 2017 A de marque BOSCH et de dimension 20x20 sont utilisés pour la structure. Les liaisons entre profilés sont faites par des équerres ou des cubes (plus fiables). Ces profilés permettent un montage des composants, des modifications faciles et un réglage des composants entre eux.

LES MOTEURS IKEA

Les moteurs utilisés sont en général ceux issus de tournevis électriques IKEA (achetés pour leurs batteries Li-Ion). En 2015, des cartes variateurs DRI0018 ont été achetées pour la commande des moteurs pour équiper le petit robot secondaire. Cette carte permet de fournir à deux moteurs une tension de 0 à 7V ou -7V à 0V (15A) selon une consigne de vitesse en PWM et une commande pour l'inversion. Cette solution permet de

prérégler la vitesse des actionneurs. Ces variateurs, couplés à une carte Arduino peuvent facilement piloter les moteurs IKEA

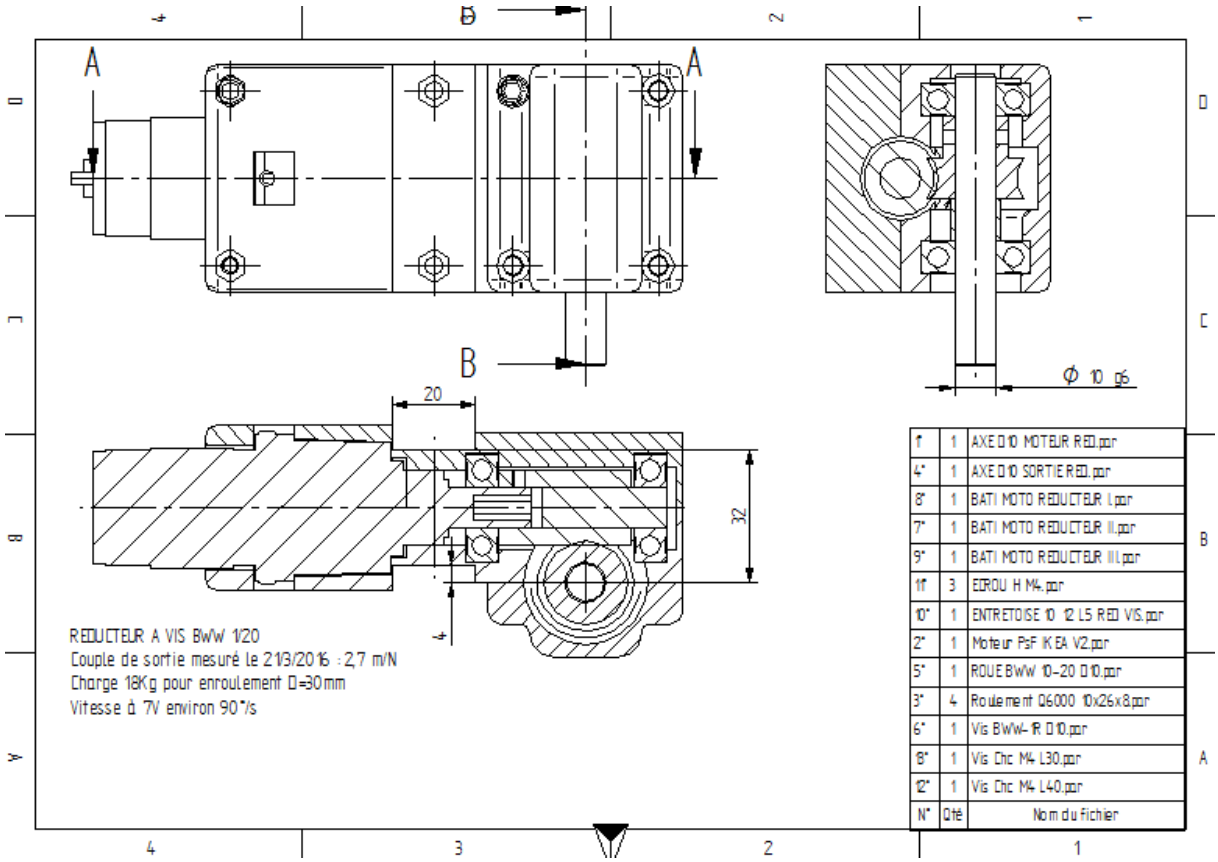
MOTOREDUCTEURS

Il a été ajouté des étages épicycloïdaux et une bague allonge aux motoréducteur IKEA. Ceci permet de diminuer la vitesse et d'augmenter le couple. A l'origine le motoréducteur IKEA possède 3 étages, et tourne à 3 tr.s-1. Une réduction de 5 est obtenue par étage supplémentaire. Il est possible de fabriquer des motoréducteurs à 2, à 4, à 5 ou à 6 étages.

Ces motos-réducteurs ont des bagues allonges en plastique et présentent une faiblesse. Ils ont néanmoins été utilisés en les renforçant par du ruban adhésif toilé. Une nouvelle version non encore fabriquée devrait offrir résistante plus grande.

MOTORÉDUCTEUR A VIS 1/20

En fin 2016 un réducteur à roue et vis sans fin (RÉDUCTEUR A VIS 1-20) de rapport de réduction 1/20 a été fabriqué. Les pièces du bâti du motoréducteur sont faites en impression 3D. Ainsi il est facile de maintenir les roulements et le moteur. Ce réducteur peut comporter deux axes de sortie et peut convenir pour concevoir une pince.



LES CAPTEURS DE FIN DE COURSE (SWITCHES)

Bien qu'il soit possible d'utiliser des temporisations pour donner l'information de fin d'un mouvement, il est préférable de tester si le mouvement a bien eu lieu par un capteur de fin de course. On distingue deux sortes

de capteurs de fin de course, ceux avec un déplacement d'une tige déplacée elle-même par un organe mobile, ceux à établissement de masse et ceux sans contact type inductif par exemple. Chaque moteur IKEA est branché avec une DB9 dans laquelle les fils des capteurs (les entrées API) passent avec l'alimentation 24V et les deux fils pour le moteur.

LES CAPTEURS OPTOÉLECTRONIQUES DE COULEUR

Les capteurs de couleur, plus simples d'utilisation que les caméras, fournissent un signal si la couleur de l'objet vu par le capteur est celle qui a été apprise au préalable. Un interrupteur permet de passer en mode *Teach*, le capteur apprend la couleur que l'on lui fait voir, lorsqu'il est à en mode *Run* la sortie du capteur passe à 1 lorsque la couleur de l'objet vu est de même signature. Attention, la distance d'analyse n'est pas très grande, entre 5 et 10mm.

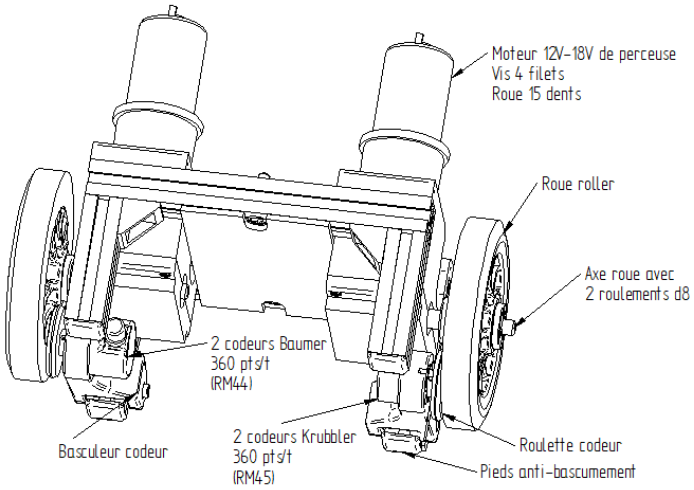
3. LA PARTIE ARRIERE

LE CHÂSSIS

Le châssis regroupe les roues motrices avec ses deux moteurs de perceuse, les deux codeurs de roues qui roulent sur la table.

Le châssis grâce à deux silentblocs, oscille pour que les deux roues motrices soient toujours en contact avec le sol.

Deux connecteurs DB9 assure la liaison électrique entre les moteurs, les capteurs TOR et la partie commande. Deux connecteurs M8 assure la liaison entre les deux codeurs et les entrées rapides (de I0.0 à I0.7) sur la CPU M13.



LES ROUES MOTRICES

Les roues du robot sont des roues de roller dont la bande de roulement a été chariotée pour obtenir une largeur plus importante. Le diamètre des roues est de 110 mm, leur fixation est faite par deux petites jantes dont l'une est goupillée sur l'axe.

LES MOTEURS 18V DES ROUES

Les deux moteurs des roues sont issus de perceuse sans fil ne comportant qu'une seule vitesse.

RÉDUCTEUR À ROUE ET VIS SANS FIN

On souhaite une vitesse du robot de 1m/s au maximum. Les moteurs 18V tournent quatre fois trop vite, l'utilisation d'un réducteur est nécessaire. Les moteurs sont placés verticalement pour un gain de place appréciable. Une vis de 4 filets et une roue de 15 dents permettent d'assurer la réduction (de 4/15).

LES CODEURS INCRÉMENTAUX

Ils offrent un signal carré sur deux voies A et B vers des entrées rapides sur la CPU. Un tour correspond à 360 points. Deux types de codeurs, de marque KRUBBLER et BAUMER sont montés sur leur basculeur codeur ; un ressort doit maintenir le contact au sol. Le déplacement de chaque roue est mesuré par deux roulettes qui roulent sans glisser sur le sol. Beaucoup de robot utilisent ce principe, en plaçant l'axe d'un codeur incrémental directement sur la roue. (Ce principe est utilisé pour le petit robot utilisé pour le TPA).

L'axe de la roulette doit être situé dans le plan vertical passant par les axes des roues. Les roulettes doivent avoir un point de contact le plus faible possible et doivent être espacés d'une distance invariante et doit être mesurée précisément est reportée dans le programme RM

FONCTION DE TRANSFERT - MESURE DU DÉPLACEMENT D'UNE ROUE

L'axe du codeur tourne sur la roulette codeur. Le codeur possède 360 pts par tour. Un tour de l'axe du codeur correspond à un déplacement de la roue de 3,14159* Dcodeur

Par exemple, si l'axe du codeur est de 40mm, la résolution est de 360/(3,14*40) pts/mm.

Pensez à initialiser les valeurs de l'entraxe des roulettes codeur, le diamètre de l'axe du codeur, afin que le programme calcul la fonction de transfert du codeur.

LES CAPTEURS OPTOÉLECTRONIQUES PORTÉE 80MM

Le règlement interdit tous contact avec le ou les robots adverses. Lors du recul, il est donc conseillé de détecter le RA.

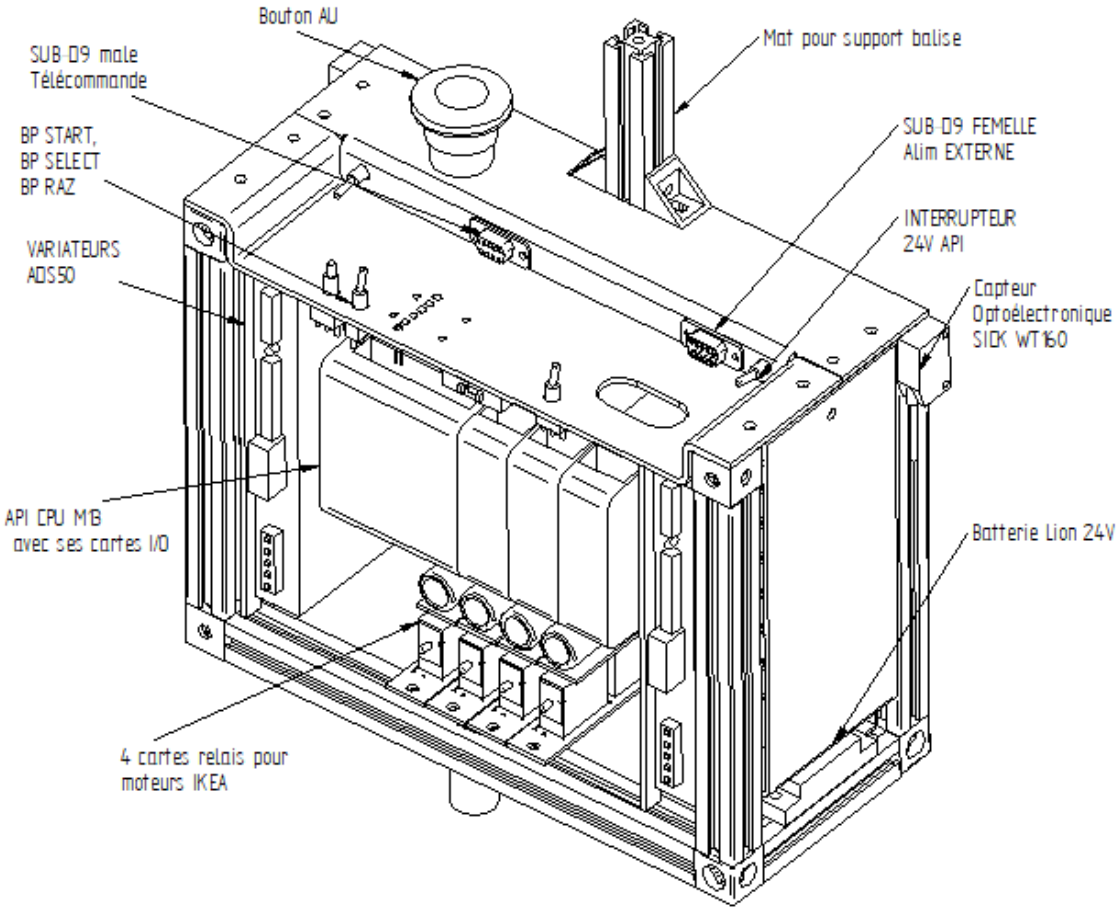
LES CAPTEURS AR

Deux boutons poussoirs peRMETTENT la détection des bordures de la table de jeux. Ils peuvent être utilisés pour la relocalisation du RM.

LA PARTIE COMMANDE

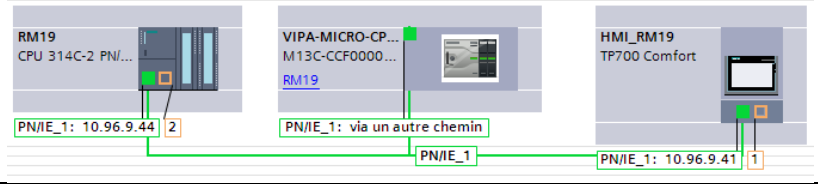
La partie commande contient :

- L'Automate Programmable Industriel CPU M13
- Les variateurs ADS50 pour les moteurs des roues
- Les deux batteries 24V pour les variateurs et pour l'API.
- Les 4 cartes relais avec batteries pour les moteurs IKEA
- La télécommande pour le déplacement manuel
- Deux capteurs WT160 pour détecter la présence à l'avant du robot adverse
- Le bouton AU pour couper l'alimentation des variateurs
- La plaque supérieure à boutons "*BP Start*", "*BP Select*", "*BP RAZ*", une roue codeuse pour indiquer la vitesse.



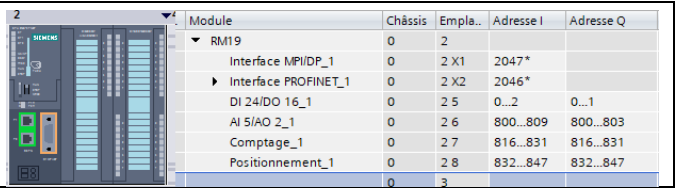
L'AUTOMATE PROGRAMMABLE INDUSTRIEL CPU M13

Le fichier PDF nommé HB400E_CPU_M13-CCF0000_17-48 donne les informations sur le câblage et l'utilisation des CPU M13VIPA. Cet API se programme soue TIA Portal ou bien sur STEP7.

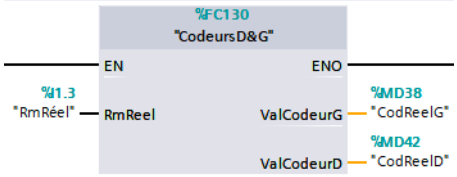
L'API est un élément sur le réseau, une CPU de type 314C est utilisé pour recevoir le programme.	

ADRESSAGE I/O DE LA CPU 314C

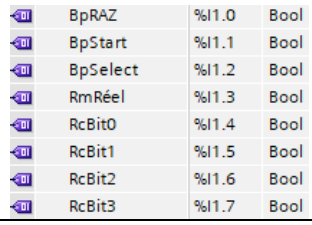
Une CPU 314c de chez Siemens est utilisée pour la CPU M13. Les adresses utilisées sont :

0..2 : pour les entrées rapides 0..1 : pour les sorties PWM (commande de servo moteurs) 816 pour le comptage des impulsions codeurs	

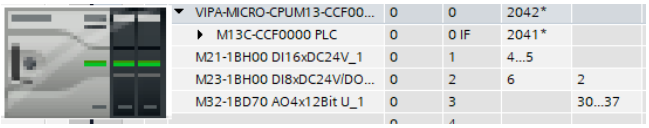
Les adresses I0.0 àI0.4 reçoivent les pistes A et B des deux codeurs droite et gauche qui roulent sur la table pour le calcul de la localisation. Ces entrées sont rapides. La fréquence maxi est de 100KHz.	**Variables API** Nom | Adr... | Type d 1 PisteACodD %I0.0 Bool 2 PisteBCodD %I0.1 Bool 3 I0.2NC %I0.2 Bool 4 PisteACodG %I0.3 Bool 5 PisteBCodG %I0.4 Bool

La fonction FC130 nommée CodeursD&G permet avec la fonction COUNT_300, de compter les impulsions des codeurs.
 
Les mots en DINT %MD38 et %MD42 donnent le nombre d'impulsions reçues.
	IF #RmReel THEN // Si mode RM réel (entrée RMreel
	"COUNT_300C_Cod_DB"(LADDR := W#820, //Adresse
	CHANNEL := 1,
	SW_GATE := NOT "BpRAZ",
	CTRL_DO := #CTRL_DO,
	SET_DO := #SET_DO,
	JOB_REQ := #JOB_REQ,
	JOB_ID := #JOB_ID,
	JOB_VAL := #JOB_VAL,
	STS_GATE => #STS_GATE,
	STS_STRT => #STS_STRT,
	STS_LTCH => #STS_LITCH,
	STS_DO => #STS_DO,
	STS_C_DN => #STS_C_DN,
	STS_C_UP => #STS_C_UP,
	COUNTVAL => #ValCodeurG,
	LATCHVAL => #Latch_VAL,
	JOB_DONE => #JOB_DONE,
	JOB_ERR => #JOB_ERR,
	JOB_STAT => #JOB_STAT);
	(idem pour le codeur D)

COUNTVAL évolue tant que l'entrée SW_GATE reste à 1, si elle passe à 0, le mot passe à la valeur 0. Le bouton RAZ est renseigné sur SW_GATE, s'il passe à 1 le compteur passe à la valeur 0.

L'octet %IB1 reçoit les boutons et la roue codeuse situés sur la plaque supérieure. Attention : Bien que l'on puisse forcer les autres entrées dans une VAT ou par le HMI, il n'est pas possible de forcer les entrées du module M13 par une VAT ou par le HMI.	

ADRESSAGE I/O DES CARTES DE L'API VIPA M13

4..5 : pour les 16 entrées TOR de l'extension M21. (%I4.0 à %I5.7) 6 : pour les 8 entrées TOR de l'extension M23. (%I6.0 à %I6.7)	

2 : pour les 8 sorties TOR, pour les commandes moteurs IKEA, de %Q2.0 à %Q2.7.

30..37 : pour les deux sorties analogiques %QW30 et %QW32 pour les consignes de vitesse des variateurs (set value).

LES VARIATEURS ADS50 POUR LES MOTEURS DES ROUES

Les variateurs utilisés pour les moteurs de roues sont des ADS50 de MAXON. Ils peuvent être alimentés de 12V à 48V, pour fournir une tension proportionnelle à un signal de consigne de ±10V fournie par une carte de sortie analogique (M32-1BD70 AO4x12Bit U) permettant de faire tourner le moteur en sens AV et AR à la vitesse voulue.

LE CÂBLAGE DU VARIATEUR

Le variateur permet de fournir une tension de + ou – 18V aux deux moteurs de roues.

Borne 1 : Borne du moteur (fil rouge)

Borne 2 : Borne du moteur (fil Vert et jaune)

La batterie pour les variateurs fournit du 24V qui est coupée par le bouton d'arrêt d'urgence.

Borne 4 : +24V (fil marron rayé noir au feutre)

Borne 5 : Masse (fil bleu)

Attention à ne pas inverser la polarité de l'alimentation sous peine de détériorer gravement le variateur !

PARTIE SIGNAL

Ci-contre un exemple de câblage avec un potentiomètre pour la consigne

Borne 1 et 2 : Tension de consigne issue de la SM234 (6 et 7 pour le moteur Gauche et 8 et 9 pour le moteur droit)

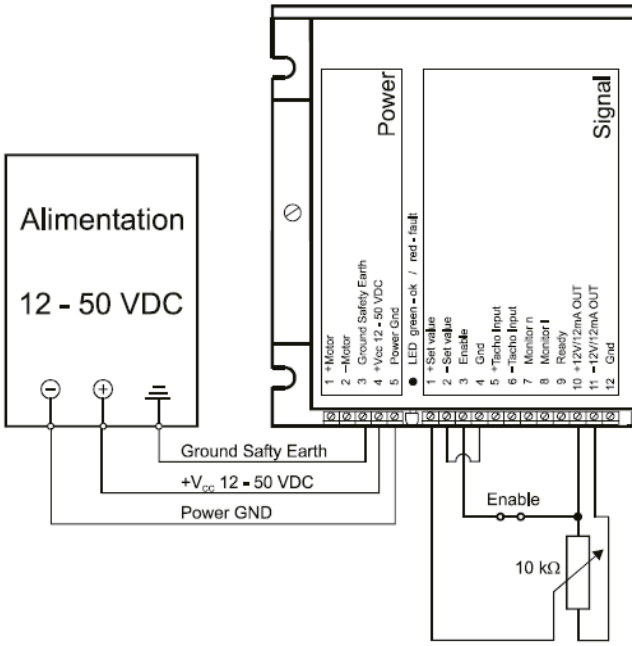
Couleurs de fils pour Set Value + et Set Value – autres que les couleurs utilisées ailleurs

Borne 3 : Enable câblé sur la sortie A4.1 (ou A4.2) Fil blanc. Aucune tension n'est envoyée au moteur si cette entrée n'est pas à 24V, le voyant clignote en vert, si 24V sur cette entrée le voyant est fixe en vert.

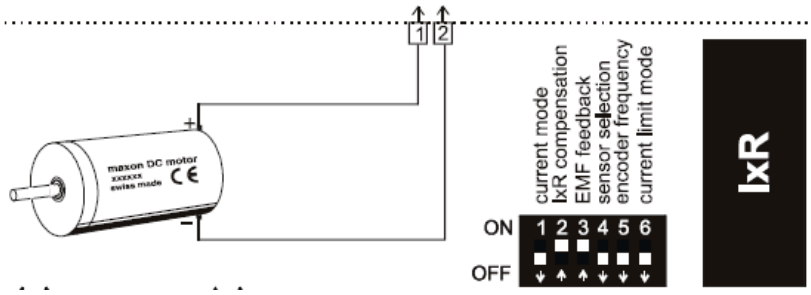
Borne 4 : Masse en bleu

Dans la VAT il est possible de commander les moteurs, ils doivent tourner en sens AV lorsque la consigne est positive.

Dans le cas contraire, coupez l'alimentation, et inversez les fils des bornes 1 et 2 coté POWER



Le mode utilisé est le IxR, les switches doivent être positionnés comme indiqué ci-contre.



L'ALIMENTATION DU RM PAR BATTERIE

LES DEUX BATTERIES 24V POUR LES VARIATEURS ET POUR L'API

Deux blocs de batteries sont placés sur les côtés du robot, elles fournissent les différentes tensions utilisées par le robot. L'utilisation des batteries Li-On impose des précautions. (Voir le schéma de câblage des alimentations en annexes).

LES DIFFERENTES TENSIONS UTILISÉES PAR LE ROBOT

Deux blocs de batteries fournissent une tension de 24V pour les variateurs et de 24V pour l'API et les capteurs. 7 éléments de 3,6V en série permettent d'assurer toutes le 24V. Les alimentations 24V pour les variateurs et l'API sont séparées (pour éviter une chute de tension de l'alimentation sur l'API, suite un appel de courant des variateurs). La consommation de l'API est plus faible (environ 1A) que celle des variateurs. Attention : Toutes les masses doivent être reliées au bâti.

PRÉCAUTIONS À PRENDRE POUR LES BATTERIES LI-ION

Le règlement oblige de couper toutes les alimentations pouvant provoquer des mouvements par un bouton d'arrêt d'urgence AU. Ceci oblige de couper l'alimentation 24V des variateurs. L'alimentation de l'API est coupée par un petit interrupteur ce qui permet de ne pas consommer de courant lorsque le robot ne se déplace pas.

- ! Attention à ne pas mettre en court-circuit des éléments de batterie.
- ! Attention à ne pas relier des alimentations (les +) différentes (par exemple les 24V venant des batteries 24V API et 24V Variateurs)

Depuis 2013 le robot utilise des batteries de type 3,6V Li-Ion. Ces batteries sont très performantes au regard des batteries Cadmium/Nickel, d'un point de vue capacité/encombrement et de tenue de la charge. Jusqu'à présent l'association achète des tournevis IKEA (9€) pour les batteries mais aussi pour les motoréducteurs et les chargeurs. Les batteries Li-Ion présentent l'inconvénient d'être fortement inflammables. Lors d'un chargement non contrôlé (fait par un chargeur non adapté) ou bien suite à un court-circuit prolongé, la batterie prend feu. Ces batteries doivent impérativement être chargées individuellement par un chargeur du commerce. Ce dernier teste la température de la batterie pendant sa charge. Un chargeur composé de 7 chargeurs permet de charger la batterie de 7 éléments d'un seul coup. Attention : Les batteries Li-Ion ne doivent jamais être en court-circuit. Un court-circuit sur un élément Li-Ion le détruit et peut provoquer un incendie (incident qui s'est produit pour la coupe 2014). Le règlement impose des fusibles le plus près possible des batteries. On veillera au remplacement facile des fusibles.

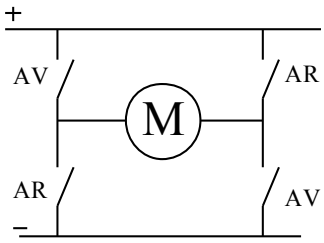
LES 4 CARTES RELAIS AVEC BATTERIES POUR LES MOTEURS IKEA

La commande par relais permet de fournir un courant assez important pour les motoréducteurs qui peuvent consommer, en pleine charge, un courant de 10A. Les cartes de sorties fournissent un courant maximum de 0,5A ce qui est largement insuffisant. Un schéma en H permet la commande en sens avant (AV) et arrière (AR) du motoréducteur. La tension d'alimentation des moteurs IKEA est de 3,6V, mais peut atteindre, sans risque pour le moteur, 10V. Une tension de 7,2V (2x3,6V) permet d'utiliser les variateurs DRI018.

LA COMMANDE EN H D'UN MOTEUR

Ci-contre le schéma en H pour faire tourner le moteur dans un sens ou dans l'autre (AV et AR)

Attention à ne pas commander le sens AV et le sens AR en même temps. La fonction FC114 permet de commander en toute sécurité le moteur.



LES INCONVENIENTS DE LA COMMANDE EN H

Le principal inconvénient de la commande par relais d'un moteur est l'impossibilité de moduler la vitesse. Il n'est donc pas possible d'assurer un démarrage et un arrêt progressif. Ceci occasionne des chocs lors des fins de course. La seule solution qui permet d'obtenir un arrêt correct et la mise en place d'amortisseurs pour freiner l'actionneur avant sa fin de course. Les amortisseurs industriels sont très chers, il est conseillé d'utiliser des amortisseurs utilisés pour des tiroirs ou des portes de placards (IKEA) d'un prix très abordable.

LA TÉLÉCOMMANDE POUR LE DÉPLACEMENT MANUEL

Manu : Permet le déplacement du RM par les quatre boutons AVD, AVG, ARD, ARG. Le passage en manu permet de faire une pause du cycle (arrêt du RM).

Power : Enclenchement de la puissance

LES RADARS AV POUR LA DÉTECTION DU RA

La partie commande accueille deux capteurs optoélectroniques WT160 pour la détection du Robot Adverse. La localisation du RA est utile pour changer de stratégie (ou de trajectoire) selon la position du robot adverse (RA) Le principe très simple, deux capteurs WT160, sont placés à droite et à gauche du robot. L'altitude des capteurs doit permettre de détecter le RA. La hauteur de visée des capteurs devra, par conséquent, être supérieure à la hauteur du plus grand objet, et inférieure à la hauteur du plus petit des RA. Cette solution permet aussi de savoir si le robot est à droite ou à gauche.

La portée des capteurs doit être préréglée en fonction de la couleur du RA. Si le RA est foncé, le capteur passe à 1 à une distance plus petite que s'il est clair. Les capteurs permettent de savoir si la distance entre RA et RM est plus petite que la portée préréglée, mais le RA hors de portée, n'est pas localisé. Solution pour la détection est très simple est fiable. Solution à garder en redondance avec un éventuel autre système.

Les capteurs optoélectroniques en mode reflex, permettent la détection d'objets éloignés, entre 100mm et plusieurs mètres. Deux types sont disponibles, la lumière est créée soit à partir d'une source LASER soit à partir de photo diodes. Attention le règlement n'autorise pas tous les classes de LASER. Bien vérifier avec la documentation construction de la classe de LASER avant son utilisation dans le RM. En 2012, les étudiants ne savaient pas quelle était la classe LASER de leurs capteurs et ont eu beaucoup de difficultés pour homologuer leur robot.

Attention, ce type de capteur réagit selon le pouvoir de réflexion de l'objet (albédo). Si l'objet est sombre, la distance de détection va diminuer, si l'objet est clair la distance augmente. Si ces capteurs sont utilisés pour

la détection du robot adverse (RA) il faut régler le capteur avec le RA (ou une pièce de sa couleur) pour obtenir une distance de détection cohérente.

4. LE PROGRAMME RM

L'OB1 (Main) appelle la fonction FC1 qui exécute les opérations qui doivent être exécutées en permanence et le bloc GRF7 qui traite le grafcet pour les déplacements et les commandes d'actionneurs.

LES ACTIONS FAITES PAR LA FC1

- Image du chiffre de la roue codeuse hors simulation (RmReel), bit0, 1, 2 et 3 de la RC.
- Deux bits "RA Devant" et "RA Derrière" passent à 1 quand le RA est détecté par un des deux capteurs à l'avant ou à l'arrière. RA devant peut être inhibé par un bit nommé "DésactivationWT160" et RA Derrière par "DésactivationRadarsAR", ceci pour ne pas prendre un objet du jeu pour le RA.
- Gestion de la temporisation du match qui force "DisablePower" au bout de 90 secondes
- Le seuillage des variables utilisées (Nobut, NoPoint)

Puis le lancement des fonctions suivantes :

- La fonction InitVar&Points (FC100) pour initialiser les coordonnées des objets et les variables, lorsque BP RAZ est à 1.
- Le calcul du score final
- Le calcul de la distance qui sépare le RM d'un point.
- La fonction Localisation (FB120) qui donne la position courante du robot (Cour, Centre et Tcour) selon les valeurs des codeurs
- La fonction Déplacement (FB110) qui commande les variateurs
- La fonction Simulation qui simule les impulsions des codeurs selon la vitesse des roues.
- La fonction pour afficher un objet sur le HMI (changement d'échelle)
- La fonction de commande d'un moteur (soit 4 au maximum)

LA FONCTION INITVAR&POINTS (FC100)

La fonction permet l'introduction des coordonnées et la couleur des objets.

Les coordonnées et la couleur des objets sont stockées dans le DB_Objets. Le point N°0 ("DB_Objets".X[0], "DB_Objets".Y[#I]) correspond à la position initiale du RM au départ.

La fonction calcule les coordonnées des objets en jeu inverse quand BP Select=1, symétriquement par rapport à un axe à X=1500.

La fonction initialise les coordonnées du RA pour la simulation.

La fonction force à zéro des bits de commande de déplacement, qui permet, lors d'un arrêt en cours de cycle, de ne pas avoir de bit de commande à 1 à la remise en route.

La fonction permet l'introduction des durées de déplacement des organes déplacés par les moteurs IKEA (M1, M2, M3 et M4)

LA FONCTION DÉPLACEMENT (FB110)

MODIFICATION DE LA VITESSE DE DÉPLACEMENT

La vitesse de déplacement est calculée par SpeedPotar * SpeedProg. La roue codeuse permet de moduler la vitesse globale et force la variable Speed Potar de 5% à 100% selon le chiffre sélectionné.

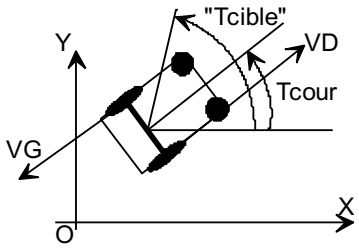
DEPLACEMENT CONSIDÉRÉ TROP LONG (CHIEN DE GARDE)

Il peut arriver que le robot ne puisse par atteindre le point, soit parce qu'il est contre une bordure, ou parce qu'un objet l'empêche d'avancer. Une temporisation est lancée au début de déplacement. La fin de déplacement est forcée si le bit #DeplVersNpTropLong passe à 1.

Attention : Il convient de donner une durée supérieure au temps de déplacement

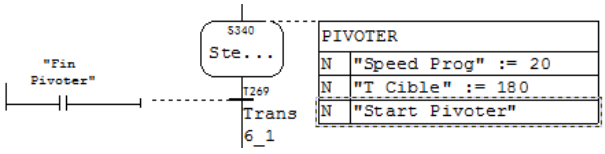
PIVOTEMENT D'UN ANGLE SELON LE REPÈRE DE LA TABLE

- **StartPivoter** ("Start Pivoter") : Le RM pivote autour du point CENTRE (milieu de l'entraxe des roues) pour atteindre l'orientation définie par "T cible" en degrés. Les vitesses VD et VG sont calculées pour pivoter, le bit **"Fin Pivoter"** passe à 1 lorsque le robot est orienté de "T Cible".



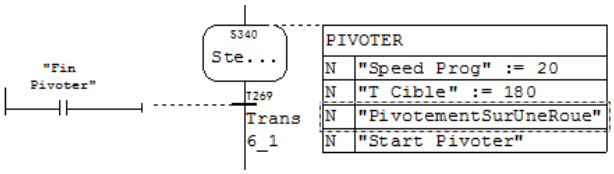
Nota : L'entrée **JeuInverse** indique le côté de départ, en effet les angles de pivotement et courant sont calculés par rapport à l'axe X de la table.

Exemple d'utilisation en S7GRAPH



OPTION : PIVOTEMENT SUR UNE ROUE

Dans certain cas, lorsque le RM est contre la bordure par exemple, il est possible de pivoter autour d'une roue. Pour cela il suffit de forcer le bit `"PivotementSurUneRoue" à 1`

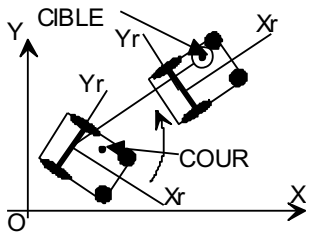


DÉPLACEMENT VERS UN POINT

- **StartDeplVersNp** *("Start DeplVersNp" OR "BpTc Depl Vers Pt")* : Lorsque cette entrée passe à 1 le RM se dirige vers le point indiqué par la variable "Npoint. Les coordonnées de chaque point sont à définir dans le FC100.

Le point COUR est un point du RM situé à "X Jauge" et "Y jauge" portés sur les axes X et Y dans le repère du RM.

Le RM s'oriente d'abord vers le point Cible avant d'avancer pour placer le point COUR (qui se déplace) sur le point CIBLE fixe.



Lorsque le point COUR entre dans un petit cercle (variable "Rayon Sur Cible" initialisé dans FC100) centré sur le point CIBLE, le RM s'arrête et le bit "Fin DeplVersNp" passe à 1 et reste à 1 tant que le bit "Start DeplVersNp" ou"BpTc Depl Vers Pt" restent à 1. Un mauvais réglage des variateurs ou un rayon sur cible trop faible, risque un dépassement du point cible, dans ce cas le robot tourne sur place à la recherche du point cible.

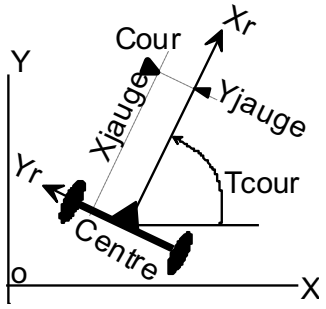
Le mouvement s'arrête aussi si le temps indiqué pour le chien de garde est dépassé. La durée est initialisée dans le FC100. Cette fonction est souvent utilisée lorsque l'on souhaite s'appuyer sur la bordure, ce qui correspond à donner un point dont les coordonnées sont en dehors de la table, ou bien pour éviter lors d'une collision avec un objet ou avec la bordure d'attendre une fin de déplacement qui ne peut jamais arriver.

UTILISATION DES JAUGES

On désigne par jauge (en X et en Y) la distance entre un point piloté du robot et son centre en X et en Y. Par défaut les jauges sur X et sur Y sont égales à zéro. Dans ce cas le point piloté est le centre de l'entraxe des deux roues motrices.

Mais il est souvent utile de placer une jauge qui correspond, en fin de déplacement, à la position du point cible par rapport au robot.

Par exemple pour prendre une pièce avec une pince éloignée de 200mm sur l'axe x du robot, on placera une jauge de 200mm sur X.



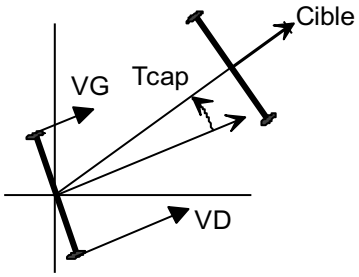
Il est possible aussi d'affiner la position de fin de déplacement en plaçant une jauge sur X (en augmentant la jauge, le robot s'arrêtera avant le point programmé)

Avec le simulateur, donnez le N° du point (MW200) et remarquez que les mots MW110 et MW112, correspondant aux coordonnées de ce point (cible). Appuyer sur M30.1, le RM doit se déplacer vers le point CIBLE, En fin de déplacement le bit M31.1 doit passer à 1. Il est possible de modifier les jauges par les mots MW124 et MW126.

Les consignes de vitesse des roues D et G sont calculées, à chaque période de scrutation, en fonction de l'écart angulaire, nommé Tcap.

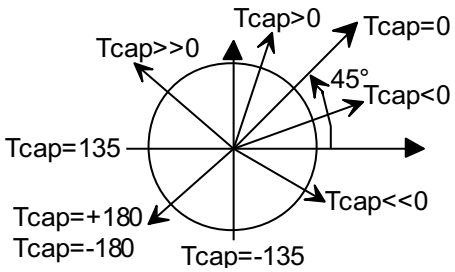
Tcap est l'angle entre la position angulaire courante (Tcour) et la direction correspondant à l'angle de cap

Pour un Tcap=0 il convient de donner une consigne équivalente sur les deux roues.



Prenons le cercle trigonométrique ci-contre, on s'aperçoit que pour un angle de Tcap+45°, la valeur du sinus et du cosinus est la même. Appliquons :

$$\begin{cases} VG := VMax.\cos(Tcap + \frac{\pi}{4} + \text{Tmodif}) \\ VD := VMax.\sin(Tcap + \frac{\pi}{4} + \text{Tmodif}) \end{cases}$$

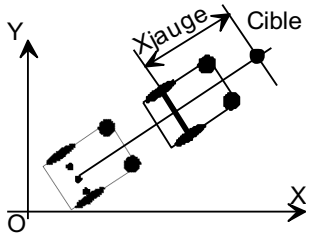


Lorsque l'angle Tcap augmente son cosinus diminue et son sinus augmente. Dans la figure 1 on voit que lorsque Tcap augmente le robot tend à s'éloigner il faut diminuer VG et augmenter VD. L'angle Tmodif calculé oblige le robot à se déporter, selon son signe, vers la gauche ou bien vers la droite.

APPROCHE D'UN POINT

Il est possible en fixant une valeur non nulle sur la jauge en X de s'approcher d'un point à une distance voulue.

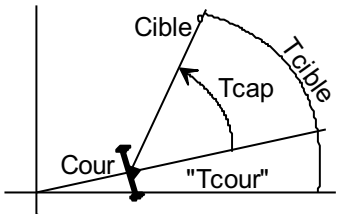
Attention à conserver une distance de déplacement supérieure à 0



CALCUL DE TCAP

On note Tcible, l'angle entre l'axe des x et la droite définie par les points Cour et Cible

$$Tcible = \arctan\left(\frac{Ycible - Ycour}{Xcible - Xcour}\right)$$



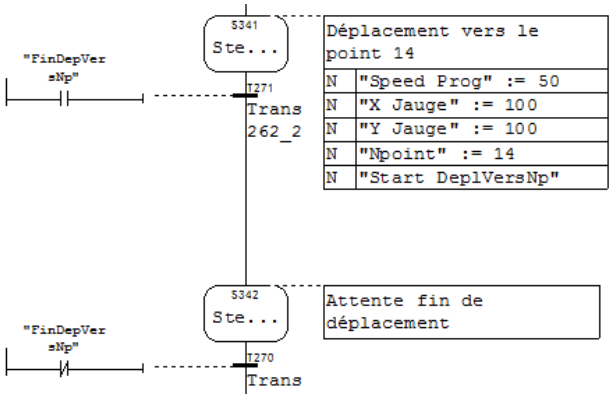
$$\text{Si on pose : } YCourCible = Ycible - Ycour$$
$$\text{et : } XCourCible = Xcible - Xcour$$
$$Tcible = \arctan\left(\frac{YCourCible}{XCourCible}\right)$$

Nota : La fonction Arctan n'accepte que des variables en REAL, il faut donc convertir les coordonnées du point Cible de INT en REAL

Il est nécessaire de calculer Tcible pour les 4 quadrants, c'est-à-dire lorsque XCourCible>0, <0 et =0 et dans chaque cas lorsque YCourCible>=0 et >0

L'angle Tcap est la différence entre l'angle Tcible et l'angle courant variant de 0 à 360° et non de -180° à +180°. Il convient de donner l'angle Tcap le plus petit entre les deux solutions pour tourner au plus court et non pas toujours dans le sens trigo.

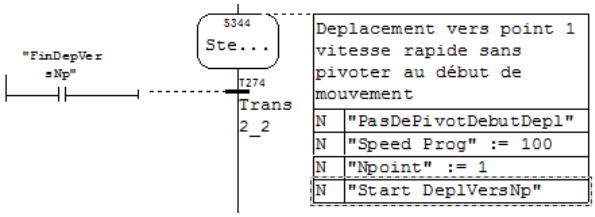
Exemple d'utilisation du Déplacement vers un point sous S7GRAPH



Attention, il faut insérer une étape entre deux déplacements consécutifs, pour tester le front sur le bit de fin de déplacement.

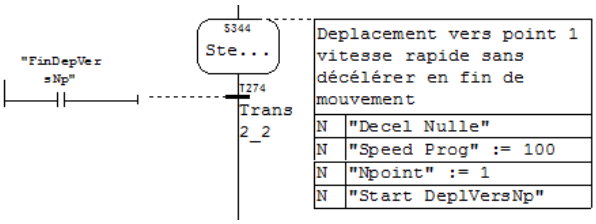
OPTIONS POUR NE PAS PIVOTER AU DÉBUT DE DEPLACEMENT

PasDePivotDebDepl : Si true le robot ne pivote pas au début de déplacement. Cette option est intéressante au départ lorsque l'arrière du robot est contre la bordure.



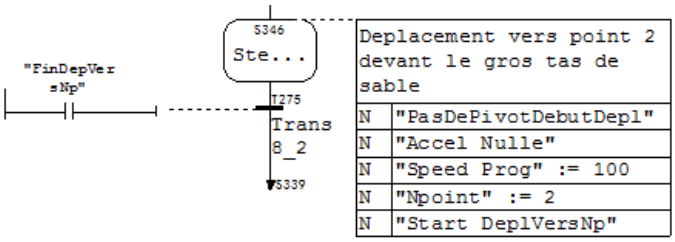
OPTION POUR DE PAS DECÉLÉRER EN FIN DE MOUVEMENT

Le bit **DecelNulle** lorsqu'il est à true, permet de ne pas décélérer en fin de déplacement, ceci permet de gagner du temps entre deux mouvements consécutifs.



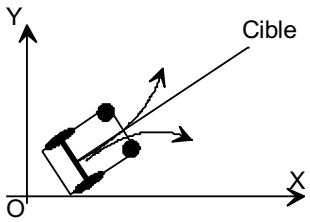
OPTION POUR DE PAS DECÉLÉRER EN DÉBUT DE MOUVEMENT

Le bit AccelNulle permet d'éviter de décélérer en début de mouvement. (option qui doit être combiné avec la précédente)

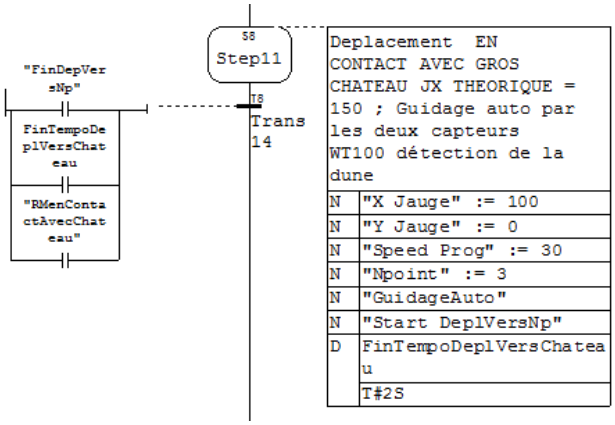


OPTION : GUIDAGE AUTO PAR DEUX CAPTEURS

Lorsque que le bit *"GuidageAuto"* est true, le robot est guidé en fonction de l'état de deux capteurs indiqués dans les entrées de la FB *CapteurAVG et CapteurAVD*. Le capteur *CapteurAVG* fait dévier le robot vers la gauche et le capteur *CapteurAVD* vers la droite. Ainsi si le robot se dirigera sur l'objet détecté par les deux capteurs.



Cette fonction peut être utilisée pour engranger des objets mobiles sur la table. Le guidage auto peut être aussi utilisé pour corriger la trajectoire lors d'un déplacement vers un objet fixe sur la table

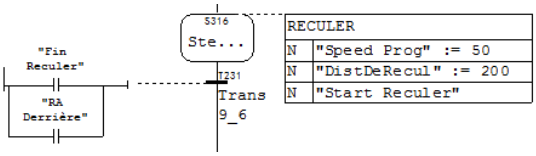


RECUL ET AVANCE D'UNE DISTANCE FIXE

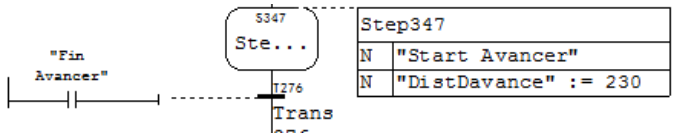
- **StartReculer** ("Start Reculer") : Lorsque cette entrée passe à 1 le robot recule tout droit de la distance spécifiée.

Exemple d'un recul de 200 mm à la vitesse 50%, il convient de vérifier la non présence du RA, bit qui

passe à 1 lorsqu'un des deux capteurs opto à l'arrière du RM passe à 1.



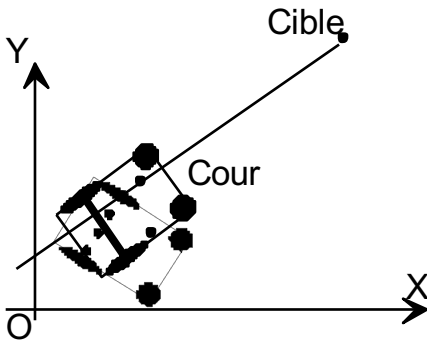
- **StartAvancer ("Start Avancer") :** identique à la fonction précédente, mais permet d'avancer tout droit sans changer d'orientation d'une distance voulue



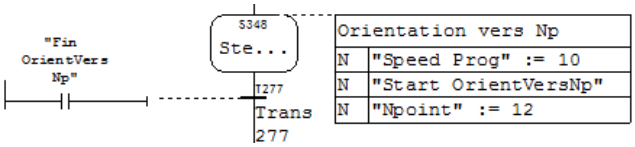
S'ORIENTER VERS UN POINT CIBLE ELOIGNÉ

- **StartOrientVersNp** ("Start OrientVersNp") : Permet d'orienter le RM vers un point CIBLE éloigné défini par le numéro de point.

Ce mode de déplacement est très utile lorsque l'on souhaite faire un tir vers un objet dont les coordonnées, par rapport à la table, sont connues.



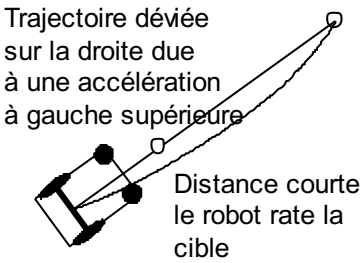
Exemple : Orientation vers le point N°12 (sans y aller)



Attention à ne pas demander l'orientation vers un point situé trop près du centre de rotation, dans ce cas le RM tourne en rond indéfiniment.

CERCLE DE TOLÉRANCE

La fin de déplacement est considérée lorsque le point Cour du robot entre dans un cercle de tolérance de rayon constant centré sur le point cible. Lorsque les variateurs sont parfaitement réglés, c'est-à-dire lorsque, pour une même consigne sur les deux variateurs, le robot avance tout droit, le rayon de tolérance peut être très petit, ce qui permet au robot de s'arrêter très près du point Cible.



Si le point Cour passe à côté du cercle le robot continue ! La chance de s'arrêter diminue avec le rayon "RayonSurCible" du cercle de tolérance. A noter que plus la distance entre le point de départ et le point cible

est grande, plus il y a de chance de rentrer dans le cercle.

DISTANCES D'ACCÉLERATION ET DE DECÉLÉRATION

Les coefficients d'accélération et de décélération (entre 0.1 et 1.0) sont dépendant des distances d'accélération et de décélération constantes. Ceci évite des pics trop importants d'intensité demandée par les variateurs (ce qui augmente fortement la durée de vie de la batterie)

Les variables DistDeDecel_100 (= 300.0 par défaut) et DistAccel_100 (= 150.0) correspondent aux distances de décélération et d'accélération pour une vitesse de 100% ("Speed Potar" * "Speed Prog" = 100%)

VITESSE MINIMUM DE CONSIGNE

Un potentiomètre sur chaque variateur nommé *Nmax* permet de régler le gain entre la valeur de consigne et la vitesse du robot. Pour une consigne faible les frottements entre les différents organes mécaniques engendrent une résistance, qui empêche les roues de tourner. Il faut, par conséquent, lorsque l'on souhaite faire avancer le RM donner une consigne minimum. Branchez la télécommande et passer en manuel. Sans modifier la vitesse par le bouton SP+, faite avancer et reculer le RM (Speed=5%). A cette vitesse, le robot doit avancer. Si ce n'est pas le cas il faut augmenter Nmax des variateurs. Attention à équilibrer les deux variateurs, appuyer sur les deux boutons en AV, le robot doit avancer tout droit.

Il est possible d'augmenter la vitesse en mode manuel, mais attention les accélérations ne sont pas prises en compte (TOR) le robot risque de basculer, la mécanique souffre à cause des couples instantanés importants, les batteries se décharges très vite (le courant étant proportionnel au couple) Il est recommandé de ne pas dépasser le 30%.

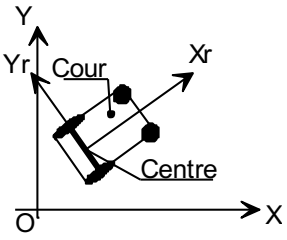
LA FONCTION DE LOCALISATION (FB120)

La fonction de localisation FB120 permet de donner les coordonnées du centre de RM et son orientation par rapport au repère de la table sur les variables globales ; "X centre", "Y centre", "TcourDeg" et d'un point Cour sur les variables globales : X cour", "Y cour", "TcourDeg" situé à une distance sur x et sur y sur les entrées Xjauge et Yjauge, après mesure du déplacement des deux roues par les deux codeurs incrémentaux. Cette fonction force aussi 4 bits qui indiquent la proximité des coins de RM aux bordures de la table.

Les mesures faites par odométrie se dégradant rapidement, l'utilisation de fonctions de relocalisation du RM semblent utiles. Une solution, qui semble performante, est celle qui consiste à recalculer la position du RM en fonction de position réelles (le robot en contact sur une bordure par exemple) pour cela les entrées RAZPosX, RAZPosY et RAZTcour permettent d'initialiser la position calculée par rapport à une position réelle.

Le déplacement des deux roues est mesuré par deux codeurs incrémentaux couplés à la carte de comptage rapide. Les déplacements *dD* et *dG* des roues D et G sont mesurés, pendant une période de scrutation dt, puis l'angle courant *Tcour* et la variation de la position du robot, *dx* et dy sont calculés. La nouvelle position du centre (*Xcentre*, *Ycentre)* est déduite. La FB120 donne aussi les coordonnées du point, nommé Cour (*Xcour*, *Ycour)*, décalé du point centre, des jauges sur X et Y par rapport au repère robot. La FB120 gère aussi le réajustement de la position courante (relocalisation)

Le principal objectif de la FB120 est de donner les coordonnées des points Centre et Cour et de l'orientation du robot par rapport au repère de la table.

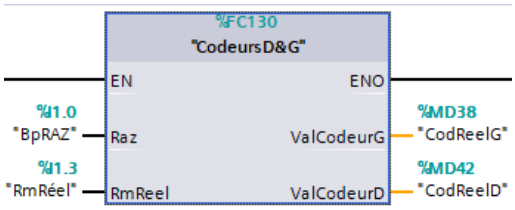


$$\overrightarrow{OCour} = \begin{bmatrix} Xcour \\ Ycour \end{bmatrix} \text{ et } \overrightarrow{OCentre} = \begin{bmatrix} Xcentre \\ Ycentre \end{bmatrix} \text{ et Tcour/X}$$

LE PRINCIPE DE LA MESURE DU DÉPLACEMENT DES CENTRES DES ROUES

On parle d'odométrie lorsque l'on mesure indirectement une distance par comptage des tours d'une roue roulant sans glisser sur le sol. Dans notre cas il est difficile de ne pas avoir de glissement des roues lors d'accélération ou de freinage.

Pour cela la roulette liée à l'arbre d'un codeur roule directement sur la table. Ce codeur envoie des impulsions vers des entrées rapides de la CPU.

La fonction ‘’Codeurs D&G’’ lancée dans l’OB1 donne, lorsque ‘’RmRéel’’ est à 1, sur les sorties ValCodeurG et D ; le nombre d’impulsions comptées issues des codeurs.	

C’est la fonction "COUNT_300C_Cod_DB" exécutée dans FC130 assure le comptage rapide.

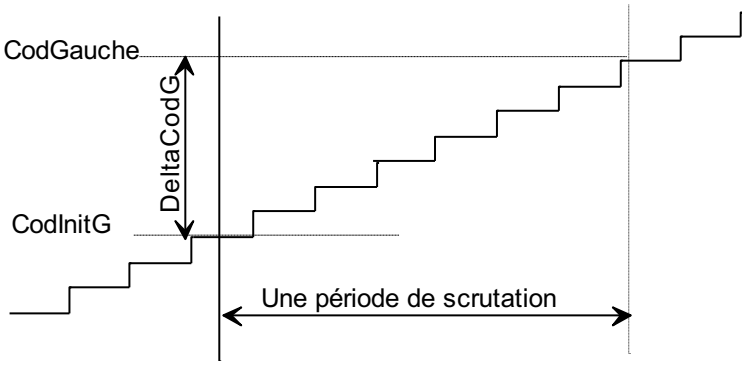
CALCUL DE LA FONCTION DE TRANSFERT DES CODEURS

Il est impératif de mesurer le diamètre des galets et de l'entraxe (en mm). Ces constantes sont définies dans FB100 par "DiamGaletsRoues" (= 34.8 par défaut) et "EntrAxeRobot" (= 233.5 par défaut). La fonction de transfert du codeur, qui correspond au nombre de points par mm, pour un codeur de 360 pts par tour est calculée :

```
#FdTCod := 360 / (#PI * #DiamGaletsRoues);
```

LE CALCUL DES PETITS DÉPLACEMENTS SUR LES DEUX ROUES

A chaque lecture du programme, le nombre d'impulsions dans une période "dt" est égal à la différence entre la valeur en cours du codeur et celle acquise au cycle précédent actualisée après le calcul. On nomme "dG" et "dD" le petit déplacement au cours de la période "dt"

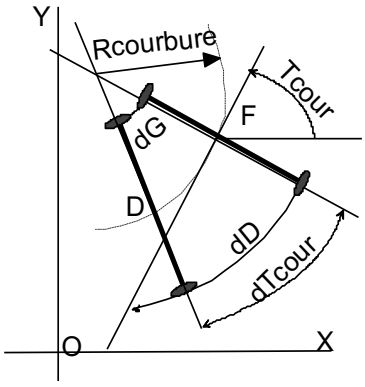


Nota : Les valeurs de "dG" et "dD" donne l'image de la vitesse de la roue gauche et de la droite

LE CALCUL DE L'ANGLE TCOUR

$$\begin{cases} dG = dTcour.Rcourbure - dTcour.\dfrac{EntraxeRobot}{2} \\ dD = dTcour.Rcourbure + dTcour.\dfrac{EntraxeRobot}{2} \end{cases}$$

$$\begin{cases} dG = dTcour.\left(Rcourbure - \dfrac{EntraxeRobot}{2} \right) \\ dD = dTcour.\left(Rcourbure + \dfrac{EntraxeRobot}{2} \right) \end{cases}$$



On note :

dG et dD : petits déplacement en points codeur pendant "dt"

dTcour : Petite orientation selon X pendant "dt"

Rcourbure : Rayon de giration du RM

EntraxeRobot : distance en mm entre les deux galets codeur

On rappelle que :

La longueur d'un arc est égale à l'angle en radian multiplié par le rayon.

$$dG - dD = -2.dTcour.\frac{EntraxeRobot}{2} \Rightarrow dTcour = \frac{dD - dG}{EntraxeRobot}$$

$$\Rightarrow Tcour = \frac{CodDroite - CodGauche}{EntraxeRobot}$$

LE CALCUL DE L'ANGLE TCOUR (SUITE)

Tcour, qui varie entre 0° et 2PI (+360°), doit prendre, quelle que soit sa valeur, une valeur angulaire entre –

PI et +PI. Il faut distinguer si sin(Tcour) est >0 ou <0.

De plus Tcour est converti en degrés et en INT pour un affichage plus simple.

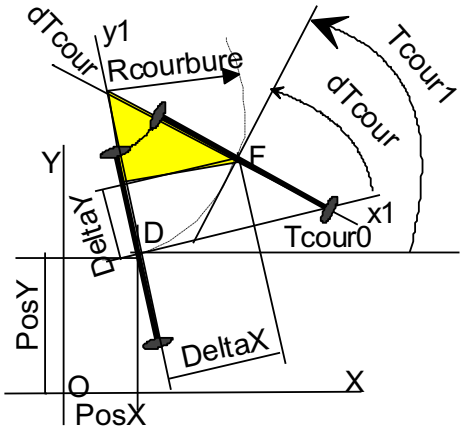
LE CALCUL DE LA VARIATION DX ET DY

On considère que la période de scrutation est assez courte pour que le mouvement s'apparente à une trajectoire circulaire.

Dans le triangle grisé :

$$DeltaX = Rcourbure.\sin(dTcour)$$
$$DeltaY = Rcourbure.[1-\cos(dTcour)]$$

$$ArcMoyen = \frac{DeltaCodD + DeltaCodG}{2}$$
$$\Rightarrow ArcMoyen = dTcour * Rcourbure$$
$$\Rightarrow Rcourbure = \frac{(DeltaCodD + DeltaCodG)}{2.dTcour}$$



LE CALCUL DE LA NOUVELLE POSITION DU CENTRE DU ROBOT EN MM SELON REPÈRE TABLE

Position initiale du robot : $\overrightarrow{OD} = PosX.\vec{x} + PosY.\vec{y}$ $\overrightarrow{DF} = DeltaX.\vec{x1} + DeltaY.\vec{y1}$

$$\overrightarrow{OF} = \overrightarrow{OD} + \overrightarrow{DF} = PosX.\vec{x} + PosY.\vec{y} + DeltaX.\vec{x1} + DeltaY.\vec{y1}$$

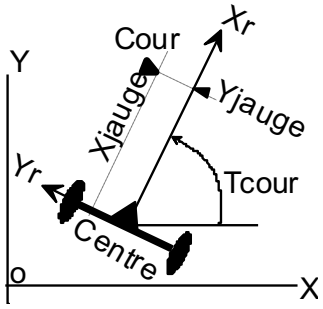
La rotation sur z du repère 1 de Tcour donne : $\begin{cases} \vec{x1} = \vec{x}.\cos(Tcour) + \vec{y}.\sin(Tcour) \\ \vec{y1} = -\vec{x}.\sin(Tcour) + \vec{y}.\cos(Tcour) \end{cases}$

La nouvelle position de RM est $\overrightarrow{OF} = \begin{bmatrix} PosX + DeltaX.\cos(Tcour) - DeltaY.\sin(Tcour) \\ [PosY + DeltaX.\sin(Tcour) - DeltaY.\cos(Tcour)] \end{bmatrix}_R$

LA POSITION COURANTE DU POINT COUR DECALÉ DES JAUGES SUR X ET Y

Le point Cour est le point piloté, c'est le point centre du robot décalé de Xjauge et Yjauge.

$$\overrightarrow{x_r} = \vec{x}.\cos(Tcour) + \vec{y}.\sin(Tcour)$$
$$\overrightarrow{y_r} = -\vec{x}.\sin(Tcour) + \vec{y}.\cos(Tcour)$$



$$\overrightarrow{OCour} = \overrightarrow{OCentre} + \overrightarrow{CentreCour}$$
$$\overrightarrow{OCour} = Xcentre.\vec{x} + Ycentre.\vec{y} + Xjauge.\vec{x}_r + Yjauge.\vec{y}_r$$

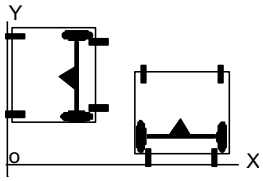
$$\begin{aligned} \mathrm{X}Cour.\vec{x} + \mathrm{Y}Cour.\vec{y} &= Xcentre.\vec{x} + Ycentre.\vec{y} \\ &+ Xjauge.\cos(Tcour).\vec{x} + Xjauge.\sin(Tcour).\vec{y} \\ &- Yjauge.\sin(Tcour).\vec{x} + Yjauge.\cos(Tcour).\vec{y} \end{aligned}$$

$$\boxed{\begin{aligned} Xcour &= Xcentre + Xjauge * \cos(Tcour) - Yjauge * \sin(Tcour) \\ Ycentre &= Ycentre + Xjauge * \sin(Tcour) + Yjauge * \cos(Tcour) \end{aligned}}$$

LES ERREURS SUR LA POSITION CALCULÉE ODOMÉTRIE SONT DUES :

- A la présence d'éléments entravant la rotation des galets des codeurs, le galet saute.
- Au décalage entre l'axe du galet du codeur et l'axe des roues. Lors d'une rotation du RM, le glissement axial est obligatoire. Les calculs considèrent l'axe de codeurs sur l'axe des roues, si ce n'est pas le cas les erreurs sont inévitables.
- A une variation de l'entraxe entre les deux roues, la valeur de l'entraxe intervient dans les calculs. En premier la valeur de Tcour ne sera pas conforme à la réalité.
- Et puis toutes les autres sources à trouver …

LE RÉAJUSTEMENT DE LA POSITION COURANTE (RELOCALISATION)



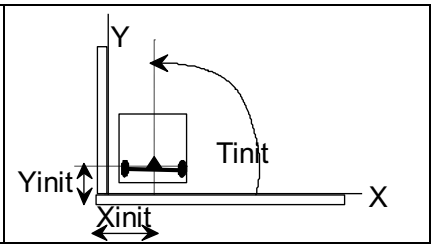
Nous avons vu que le défaut de position par odométrie est d'autant plus grand que le déplacement est important. Il est donc possible, qu'à partir d'une certaine distance parcourue, de procéder à un réajustement de la position courante

LE PRINCIPE PAR CONTACT SUR LA BORDURE

Si l'avant ou l'arrière du robot vient toucher la bordure, il est facile de modifier la position courante calculée par la valeur de la distance entre l'axe des roues et le bord arrière ou avant. Mais seule la position sur un seul axe est correcte, par exemple si la position courante calculée est de (78, 665, 178°) lorsque l'avant du robot touche la bordure gauche, la valeur en X doit être remplacée par la valeur constante et mesurable de la distance entre l'axe des roues et le bord avant, il est possible de modifier le Tcour par 180°, mais la valeur de Y ne doit pas changer.

LA REMISE A ZÉRO DES MOTS CODEURS

Lors de la RAZ, les deux mots des codeurs sont forcés à la valeur L#0. Les variables images de la position prennent les valeurs de la position indiquées par Xinit, Yinit et Tinit.

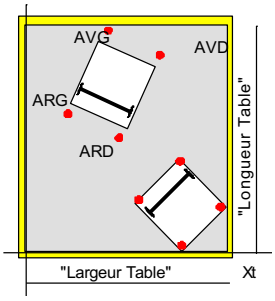


Les autres entrées RAZ suivantes permettent d'initialiser la position en X ou bien en Y ou bien en orientation

- Au front montant de RAZPosX, la position selon X (Xpos) prend la valeur de Xinit.
- Au front montant de RAZPosY, la position selon Y (Ypos) prend la valeur de Yinit.
- Au front montant de RAZTcour, l'orientation selon X (Tcour) prend la valeur de Tinit.

BITS HORS ZONE

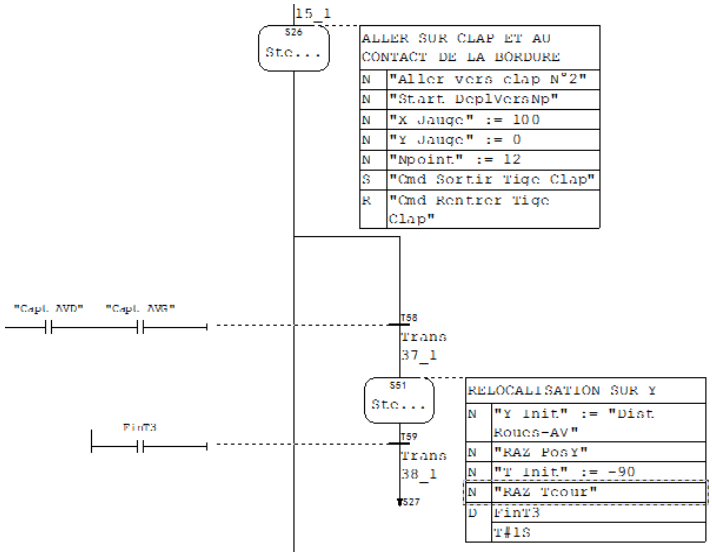
Traités auparavant dans une FC spéciale, les bits qui signalent les coins du robot en dehors des limites de la table sont traités dans la FB120. Ces bits sont nommés *AVGHorsZone, AVDHorsZone, ARGHorsZone et ARDHorsZone* et passent à 1 lorsqu'un coin AVG, AVD, ARD ou ARG du robot sort de la zone de la table. Ces bits sont très utiles en simulation, en effet ils sont testés dans la fonction de simulation et s'ils passent à 1 les impulsions simulées des codeurs de roues s'arrêtent, comme si le robot était stoppé par la bordure (la roue droite ou gauche ne tourne plus)



L'EXECUTION D'UNE ACTION DE RELOCALISATION DANS UN GRAFCET

La localisation par odométrie n'est pas très précise lorsque la distance parcourue est grande. Pour que la position calculée soit la plus proche de la position réelle, il faut périodiquement relocaliser le RM par rapport à un élément fixe. Cela se fait, par exemple, lorsque le RM est en contact avec une bordure.

Lors d'une action vers une bordure il suffit de demander la relocalisation selon Y, en forçant le bit "RAZ PosY" après avoir forcé "Y init" à la valeur correspondant à la distance roue-bord avant du RM.



AFFICHAGE DES POSITIONS DES POINTS SUR LE HMI

La fonction "AffSurHMI" permet de déplacer un petit cercle sur l'écran du HMI en respectant l'échelle entre le HMI et la table ainsi que la position initiale du cercle défini dans la vue modèle TableDeJeux du HMI. Ci-contre, le calcul de "AffXcentre" et "AffYcentre" en fonction de la position du point centre.	"AffSurHMI"(XenMM := #XcentreReal, YenMM := #YcentreReal, XposInitEnPixels := 15, YposInitEnPixels := 40, RayonCercleEnPixels := 4, XaffEnPixels => "AffXcentre", YaffEnPixels => "AffYcentre");

LIGNES DE LA FONCTION "AFFSURHMI"

```
#XaffEnPixels := REAL_TO_INT(IN := #XenMM * 0.15) - #RayonCercleEnPixels - #XposInitEnPixels;
#YaffEnPixels := REAL_TO_INT(IN := -1 * #YenMM * 0.15) - #RayonCercleEnPixels + 300 - #YposInitEnPixels;
```

LA FONCTION DE SIMULATION DU DÉPLACEMENT (FB130)

La fonction « simulateur de déplacement », activée lorsque l'entrée *"Entrée Simul" est à 0*, permet de donner deux mots images du comptage des impulsions issues de la rotation des codeurs incrémentaux en fonction des consignes de vitesse, de donner des bits images de l'état des radars avant et arrière selon la position du RA. Ceci permet de travailler sans le robot réel, en affichant le tracé de la trajectoire sur un HMI réel ou simulé par l'ordinateur.

L'ACTIVATION DE LA FONCTION DE SIMULATION

Une entrée nommée *"Entrée Simul"* est toujours branchée à l'alimentation 24V, par conséquent lorsque le robot réel est utilisé cette entrée est à 1, et lorsque PLCSIM est lancé cette entrée est à 0, la fonction de simulation est activée.

LA SIMULATION DES VARIABLES DES COMPTEURS DES CODEURS INCRÉMENTAUX DE DEPLACEMENT

Si ActSimul est à 1, les variables images des compteurs d'impulsions des codeurs incrémentaux de déplacement, nommées ImageCodG et ImageCodD sont calculées par la FB130 en fonction de la consigne de vitesse sur les deux roues CmdVG et CmdVD correspondant aux sorties CmdVarG %QW30 et CmdVarD %QW32 reçues sur les entrées SetValue des deux variateurs de roues et si EnableG et EnableD sont à 1. Le défaut de réglage des variateurs est simulé en donnant une valeur sur *GainVG* et *GainVD* différente de 1.0.

LA SIMULATION DES RADARS AV ET AR

Le simulateur donne aussi des bits images des capteurs WT100 et des radars de recul qui passent à l'état 1 lorsque le robot adverse, dont on a fixé ses coordonnées. La position du robot adverse doit être donnée en mm aux variables "XRASimulé" et "YRASimulé". Un obstacle peut être détecté par un des radars à l'avant ou à l'arrière. Les radars AVG et AVD ont une portée qui peut être réglée de 300mm à 3m. Les capteurs ARG et ARD ont une portée fixe de 30 à 80mm variant selon la couleur de l'objet détecté (un objet noir ne renvoie pas beaucoup de lumière. Le simulateur permet de simuler ces capteurs.

Il convient de donner aux constantes "PortRadAR" et "PortRadAV" une valeur qui correspond à la portée réelle des capteurs (qui change en fonction de la couleur du RA et de l'éclairage ambiant)

LA COMMANDE DES ACTIONNEURS TOR (FC112)

Permet de commander un moteur en sens AV ou en sens AR et de donner les états de l'actionneur selon ses fins de course ou en fin de tempo.

La FC114 stoppe le mouvement si les capteurs *FCfinAR* ou *FCfinAV* passe à 1 ou bien si le temps de fermeture défini par les variables *TV_CmdAR* et *TV_CmdAR* est écoulé.

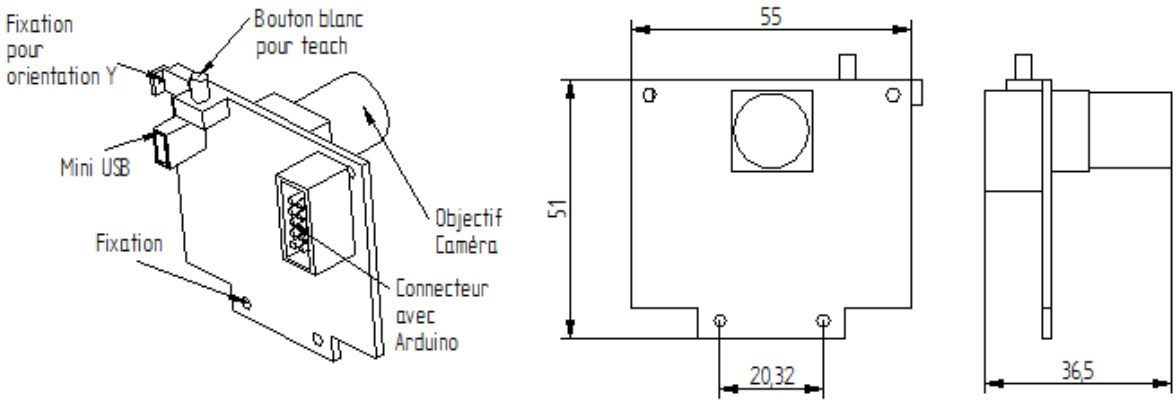
5. LE SYSTÈME DE VISION

Le système de vision peut permettre de localiser "de loin" la position des balises mobiles (posées sur les robots adverses), d'affiner la position du RM en mesurant les distances entre le RM et les balises fixes sur des supports placés aux coins de la table, et aussi de localiser des objets à saisir sur la table ou des obstacles mobiles. Le RA possède une plateforme de 80x80 (obligatoire selon règlement) située à 430mm du sol. Celle-ci et celle du RA peuvent recevoir une balise de forme et couleur au choix, de dimension 80x80x80 au maximum. Le règlement jusqu'à nos jours, autorise la mise en place de capteurs sous la balise dans un espace de 80x80 à une altitude comprise entre 350mm (hauteur maxi) et 430 (position de du support de balise).

Les balises peuvent être des sphères de gros diamètre tout en restant dans l'encombrement de 80x80. La couleur choisie doit permettre d'obtenir un bon contraste entre la balise et l'environnement, il est possible d''y loger un éclairage à l'intérieur.

LA CAMÉRA

La société LEXTRONIC commercialise (moins de 60€) une caméra intelligente nommée CMU CAM5 PIXY. De petite taille elle peut s'intégrer facilement dans le RM, deux trous de fixation permettent de l'installer facilement, une petite patte permet d'y loger une barre pour l'incliner. L'objectif de la caméra est à focale fixe, l'image possède 200x300 pixels, qui n'offrent pas une très bonne résolution artistique, mais largement suffisante pour la robotique. Un petit bouton blanc permet l'apprentissage des couleurs, un connecteur mini USB permet de voir l'image sur l'écran d'un PC, et un connecteur 10 points permet le dialogue avec une carte Arduino. Le logiciel PixyMoon, intégré fournit des informations pertinentes de l'image.



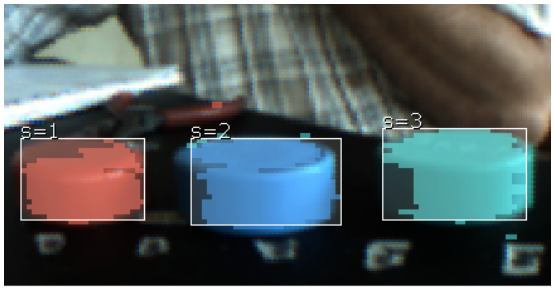
La PIXY fournit, pour chaque tache j de 7 couleurs différentes, trouvée dans l'image, ses coordonnées (pixy.blocks[j].x, pixy.blocks[j].y) sa couleur ((blocks[j].signature), sa largeur (pixy.blocks[j].width) et sa hauteur (blocks[j].height) Le programme de démo hello_world.ino permet de voir le fonctionnement de la PIXY.

RÉGLAGE DES SIGNATURES

La signature d'un objet est sa couleur. Le réglage des signatures se fait à l'aide de PixyMoon, la caméra pixy branchée sur le PC via la mini USB.

La fonction SetSignature permet de définir les différentes signatures en vérifiant que chacune d'entre elles sont bien identifiées.

Ci-contre trois couleurs (rouge, bleu et vert) sont identifiées et discriminées. Le numéro de signature apparait sur chaque forme. Chaque balise doit être d'une couleur discriminable des autres balises et du fond de l'image.



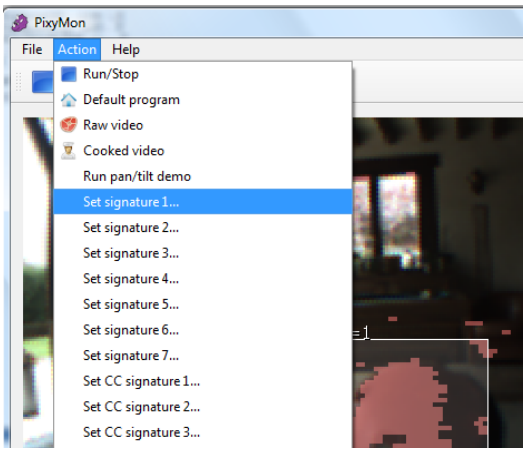
LE SUPPORT DE LA CAMÉRA

La société LEXTRONIC commercialise un support 2 axes nommé "Pan/Tilt" (orientation / inclinaison) pilotés par servos moteurs, un programme de démo dans PixyMoon permet de suivre un objet de couleur définie.

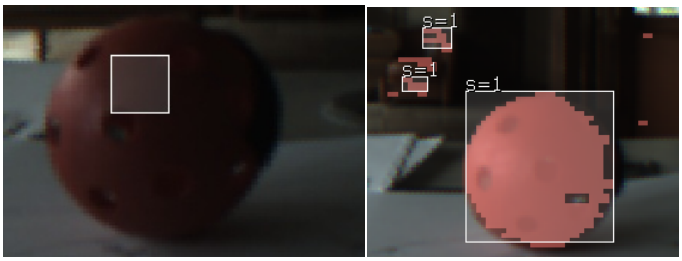
Les fils des deux servo moteurs sont connectés directement sur la PIXY les fils jaunes vers le connecteur 10pts, l'axe Z (Pan) sur le connecteur de gauche, l'axe Z (Tilt) sur celui de droite. Le programme de démo dans PixyMoon permet de suivre un objet de couleur définie par un set signature.

PROCÉDURE POUR RÉGLER LE SYSTÈME DE VISION

Brancher la pixy par le petit câble USB et lancer PIXYMOON. L'image vue par la PIXY est visible sur l'écran. Dans la rubrique Action, demandez le Set de la signature 1 (couleur N°1) encadrez par une petite fenêtre la zone qui correspond à la couleur recherchée.



En relâchant le bouton de la souris les zones reconnues sont identifiées. Il est possible que des zones de même couleur apparaissent. Si elles ne sont pas trop importantes, le programme ne les prendra pas pour la balise qui, pour qu'elle soit reconnue comme telle, doit être de taille importante.



Remarquez quand vous approcher la balise de la caméra, la diode de la pixy s'allume à la couleur recherchée.

Pour plus d'informations sur la PIXY :

http://www.cmucam.org/projects/cmucam5/wiki/Assembling_pantilt_Mechanism

http://www.cmucam.org/projects/cmucam5/wiki/Run_the_Pantilt_Demo

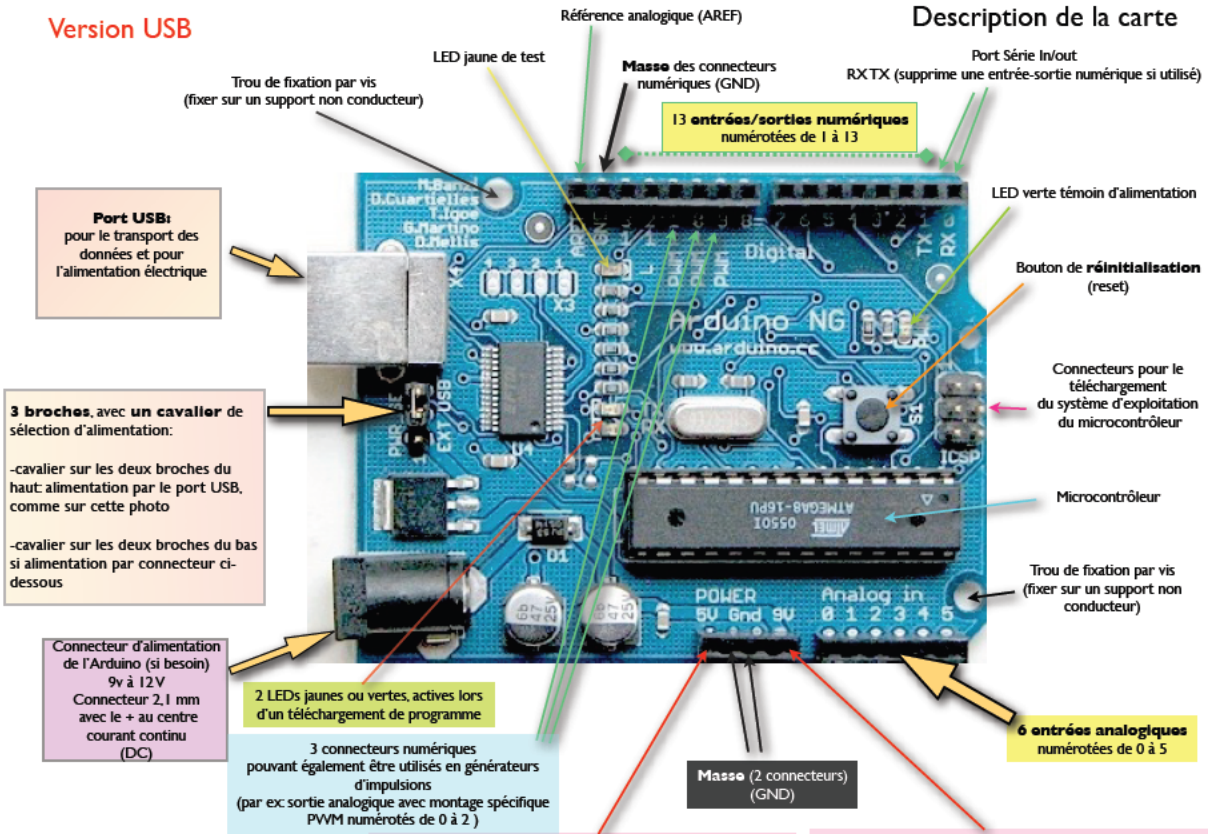
LA CARTE ARDUINO DU SYSTÈME DE VISION

Une carte arduino permet de faire l'interface et d'effectuer les calculs entre la PIXY et l'API.

La Pixy est branchée sur carte arduino par un connecteur 10 pins sur Pixy et 6 pins sur l'arduino, l'arduino reçoit les informations de la caméra, fait les calculs et envoies le résultat de ces calculs à l'API par le biais des entrées et des sorties TOR.

LES CONNECTEURS DE LA CARTE ARDUINO

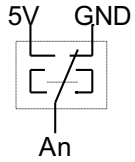
La photo ci-dessous décrit la fonction de chaque connecteur.



LE CÂBLAGE DES BOUTONS OU CAPTEURS TOR SUR ARDUINO

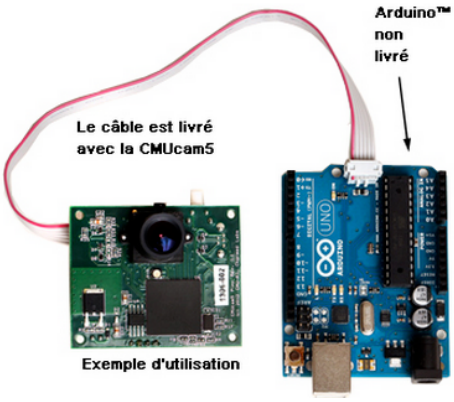
La carte arduino possède 6 entrées analogiques, il est par conséquent possible de lire 6 tensions variant de 0 à 5V. Mais il est possible aussi de les utiliser en TOR (0 ou bien 5V)

1. Brancher les boutons au 5V (rouge) et à la masse (GND Bleu)
2. Reliez les pattes milieu des boutons aux entrées notées A0 à A5.
3. Donner un nom au pin recevant chaque bouton : #define BP_RechRA A0
4. Dans le setup, indiquer le mode (entrée ou bien sortie) pour la pin : pinMode(BP_RechRA, INPUT) ;



LE CÂBLAGE ENTRE LA CAMÉRA PIXY ET L'ARDUINO

Une nappe 6 conducteurs relie le connecteur 10 pts de la PIXY au connecteur 6 pts de la liaison I2C de la carte Arduino.



LE CÂBLAGE ENTRE LES SERVOMOTEURS ET L'ARDUINO

Le servomoteur a besoin de l'alimentation 5V, celle-ci peut être pour un seul moteur prélevée sur la carte arduino, pour plusieurs moteurs il faut utiliser l'alimentation externe en reliant toutes les masses (GND). Le fil jaune de commande doit être relié sur un pin en PWM (pin 3 et 5 par exemple).

L'UTILISATION DE LA PIXY

Pour pouvoir utiliser les fonctions de la caméra PIXY il est nécessaire de charger les deux bibliothèques : SPI et Pixy : *#include <SPI.h> #include <Pixy.h>*, et de donner un nom à votre caméra Pixy pixy. Dans le setup, il faut initialiser la pixy par la fonction : *pixy.init(),* puis, dans la loop, l'acquisition de l'image et des blocks se fait par la fonction : *pixy.getBlocks()*. La récupération des coordonnées du block i se fait par les fonctions : *pixy.blocks[i].x* et ; *pixy.blocks[i].y,* la signature relative à la couleur de l'objet par : *pixy.blocks[i].signature.*

L'AFFICHAGE SUR LE PC DES VARIABLES INTERNES A L'ARDUINO

Lors de la mise au point il est souvent utile de visualiser les valeurs des variables calculées. La liaison série, entre la carte Arduino et le PC via la prise USB, est utilisée. Dans le setup, il faut préciser la vitesse de transmission des données (9600 bauds) : *Serial.begin(9600),* déclarez une chaine de n caractères : *char buf[80]*, puis demander l'affichage d'une ligne par les deux fonctions : *sprintf(buf, "\n Angle=%3d Dist=%3dcm", Angle, Distance) ;* et *Serial.print(buf);* les %3d vont être remplacés par la valeur des variables dans l'ordre où elles sont placées.

ESTIMATION DE LA DISTANCE DE LA BALISE FONCTION DE LA LARGEUR VUE

La taille du capteur CCD et la focale de la caméra sont fixes Par expérimentation, le rapport : $\frac{Focale_{mm}}{\text{LargCapteur}_{mm}}$ peut être fixé par la constante : CoefCam :=250

La distance est : $\frac{\text{LargObjet}_{mm} * Focale_{mm}}{\text{LargeurObjet}_{pixels} * \text{LargCapteur}_{mm}}$ => $Dist = 250\frac{DiamBalise}{Width}$

UTILISATION DE LA PIXY

LA LOCALISATION DE LA BALISE RA

Il est donc possible de déterminer la distance entre le RM et le RA qui possède une balise de diamètre connu. Ceci permet de décider des actions à faire (ou ne pas faire) en fonction de la proximité du RA. Permet de remplacer les deux WT100.

LA LOCALISATION D'OBJETS

Il est possible de localiser d'un objet d'une certaine couleur sur la table de jeux et de guider le RM pour aller chercher cet objet.

PROGRAMMES PIXY

DetectBalise

Le programme ci-dessous permet de forcer une entrée API si un objet, de couleur voulue, est détecté à moins d'une distance préréglée par un potentiomètre.

```
/*Fichier : DetectBalisePotar
Ce programme permet d'indiquer que la balise de signature préréglée est dans l'image
et à une distance inférieure à la valeur ajustée par un potentiométre.
Si la distance est inférieure à cette distance la sortie vers API passe à 1
*/
#define SortieVersApi  10    // Adresse pin Arduino pour relais pour fournir du 24V à une entrée API
#define SortieLED      3     // Adresse pin Voyant indiquant que RA est détecté
#define SortieBUZZER   5     // Adresse pin Pour buzzer PWM
#define EntreePotar    A0    // Adresse pin entrée ANA du potar réglage de la distance, dépend de l'éclaiage
#include <SPI.h>
#include <Pixy.h> // Charge la bibliothéque de fonction de la caméra PIXY
Pixy pixy;
char      buf[80] ; // Reserve une chaine de caractére pour affichage par sprintf(buf, "texte %3d", Var)
void setup() // Procédure appelée une seule fois à la mise en run (idem OB100)
{
pixy.init();
Serial.begin(9600)            ; // Conf liaison série pour affichage sur PC
pinMode (EntreePotar,  INPUT)    ; // Conf POTAR sur entrée ANA
pinMode(SortieVersApi, OUTPUT)   ; // Conf Sortie relais
pinMode(SortieLED,     OUTPUT)   ; // Conf en sortie pour LED
pinMode(SortieBUZZER,  OUTPUT)   ; // Conf en sortie pour Buzzer
} // de setup

//---------------------------------------------------------------
void loop()
{
int blocks = pixy.getBlocks()                    ; // Acqui image nombre de blocks trouvés dans l'image
int Larg = pixy.blocks[0].width                  ; // Largeur de la balise en pixels
int DiamBalise = 50                              ; // Diametre de la balise en mm
int Dist = 375 * DiamBalise / pixy.blocks[0].width     ; // Calcul de la distance selon largeur de la balise
int Xobjet = pixy.blocks[0].x                    ; // Donne la position de l'objet en X
int Yobjet = pixy.blocks[0].y                    ; // Donne la position de l'objet en Y
```

Nom du fichier Arduino

Définition des entrées et des sorties

Bibliothèques de fonctions pour la PIXY

Idem OB100, la fonction setup est exécutée une seule fois à la mise sous tension.

Affectation des pins aux entrées ou aux sorties

Programme principal, comme l'OB1

Prise des infos de l'image

Calcul de la distance en fonction de la largeur de l'objet

Coordonnées de l'objet

Valeur du potar de réglage

```
int SignRecher   = 1                          ; // Signature de la balise recherchée (à régler sur la pixy)
int ValPotar     = analogRead(EntreePotar)        ; // Lecture val ANA A0
int DistReglage  = map(ValPotar, 0, 1023, 100, 500) ; // Distance à partir de laquelle la sortie commute
int Buzzer       =  map(Dist, 50, 500, 255, 0)    ; // Pour buzzer

sprintf(buf, "Balise Distance=:%3d DistReglage : %3d\n", Dist, DistReglage );
Serial.print(buf);
 if (Dist<DistReglage )
   { digitalWrite(SortieLED,    HIGH)  ;
     digitalWrite(SortieVersApi, HIGH)  ;
     analogWrite (SortieBUZZER,  Buzzer) ;
   }
    else
  { digitalWrite(SortieLED,     LOW)   ;
    digitalWrite(SortieVersApi,  LOW)   ;
    analogWrite (SortieBUZZER ,   0 )   ;
 } // de if
} // de loop
```

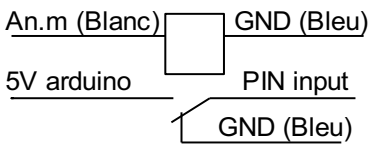
Pour affichage des variables sur écran du PC

Forçage des sorties selon la distance calculée et réglée par le potar

DIALOGUE ARDUINO API PAR E/S

Une sortie API en 24V est reliée à la bobine d'un relais 24V, ce dernier doit posséder un contact INV (1 NO et 1 NF) qui permet de donner du 5V à une PIN d'entrée de la carte Arduino. Attention le contact doit être relié aussi à la masse pour éviter que l'entrée passe à 1 sans le vouloir par un parasite. Ce qui permet, lorsque la sortie API passe à 1 de faire passer l'entrée de l'Arduino à 1.

De même pour forcer une entrée API en 24V depuis la carte Arduino, il suffit de relier une PIN de sortie de la carte Arduino en 5V à la bobine d'un relais 5V qui possède un contact NO alimenté en 24V et relié à la carte d'entrées de l'API.



6. LE GRAFCET PRINCIPAL DU RM

Fonction pour gérer les actions de déplacement et de commande des actionneurs, définies en S7GRAH, elle permet d'assurer les tâches à effectuer selon la stratégie employée. De gérer la présence du robot adverse (RA).

ACTIONS A FAIRE LORSQUE RA EST DEVANT LE RM

Le bit "*RA Devant*" est à 1 lorsque le RA est devant le RM. La distance entre le RA et le RM doit être la plus faible, mais pas à 0 pour éviter la collision. Les capteurs WT100 doivent être réglés de telle sorte qu'ils détectent le RA (attention sa couleur peut modifier la distance) suffisamment tôt pour s'arrêter le RA en vitesse maxi du RM. La détection par la PIXY, qui filme la balise du RA, doit permettre de s'affranchir de la couleur du RA. Reste à trouver une couleur différente des couleurs des pièces du jeu.

Trouver une solution universelle qui définit les actions à faire lorsque le RA est devant est très compliqué voire impossible. Il faudra donc traiter chaque cas particulier. Certains déplacements n'auront peut-être pas besoin d'être traités, par exemple de déplacement vers la bordure à moins de 400mm du RM.

Une solution assez simple est, lorsque le RA est détecté, est d'attendre qu'il s'en aille, et en fin d'attente si le RA est toujours là, abandonner la tâche engagée pour faire la suivante. Le principal défaut est qu'il est difficile de quantifier la durée de temporisation. Une observation des comportements des robots permettra peut-être de quantifier une durée optimale.

Une autre solution assez simple consiste, lors de la détection du RA, à reculer puis à se diriger vers un autre point.

TRAITEMENT DU CAS RA DEVANT DANS LE GRAFCET

Dans l'exemple ci-contre, pendant le mouvement vers le point 11, si le bit "RA Devant" passe à 1, l'étape S de droite s'active, une temporisation est lancée. Si le RA s'en va avant la fin de temporisation, l'étape où se trouve le déplacement s'active de nouveau, ce qui permet de continuer le mouvement vers le point.

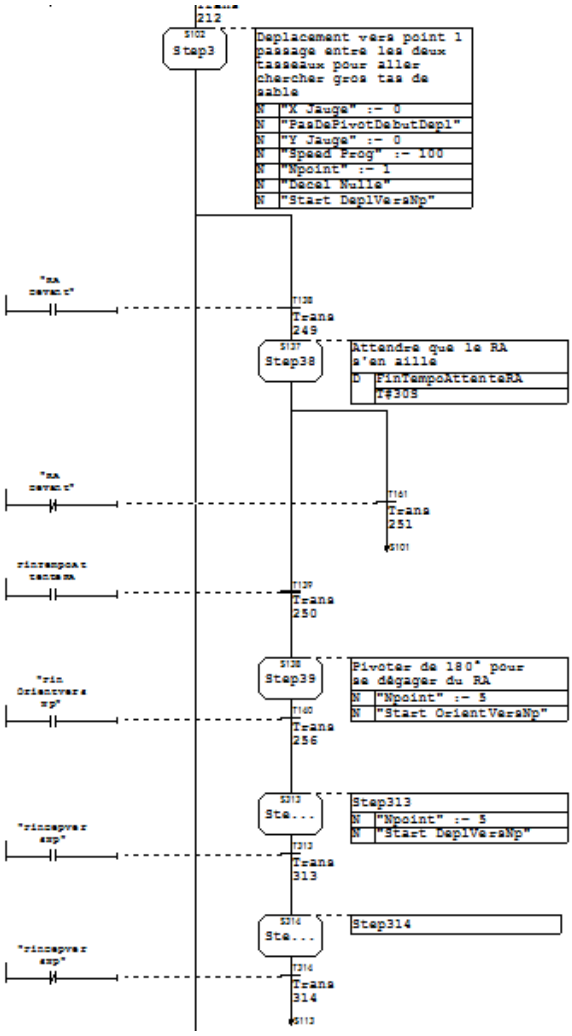
Si le RA est toujours devant en fin de temporisation, le RM doit pivoter vers un point opposé pour que les deux WT100 soient hors de portée du RA.

Abandonner la tâche en cours en faisant un saut d'étape pour faire les tâches suivantes.

Attention les tâches suivantes doivent être situées du côté opposé à RA. Dans le cas contraire il faut trouver un chemin qui permet de contourner le RA.

Nota : La durée de l'attente que RA s'en aille dépend de l'importante de la tâche en cours.

Nota : A chaque tâche il est judicieux de signaler qu'elle a été faite. Ceci doit permettre de refaire une tâche abandonnée.



STRUCTURE DU GRAFCET

Le bloc FBn est souvent composé d'un certain nombre de grafcets, le grafcet principal (ou maître) et des grafcets de tâches qui peuvent être considérés comme des macro-étapes. Le grafcet principal demande l'exécution des tâches qui sont faites par les grafcets de tâches, le grafcet maitre attend la fin d'exécution du grafcet macro-étape pour continuer. Le dialogue entre le grafcet maitre et les grafcets macro-étapes se fait par des bits internes forcé par les étapes des grafcets.

La numérotation des étapes des grafcets doit être faite en rapport avec le numéro du grafcet. On conseille vivement de faire les calculs importants en SCL dans le bloc FC1. Le grafcet principal est conçu en fonction de la stratégie permettant de gagner le maximum de points. C'est le grafcet principal qui va décider de l'ordre des tâches. Le nom du bloc FB15 est relatif à l'année de la coupe, par exemple en 2016 le FB15 doit s'appeler FB16.

LE DIALOGUE ENTRE LE GRAFCET MAITRE ET LES GRAFCETS MACRO-ETAPES

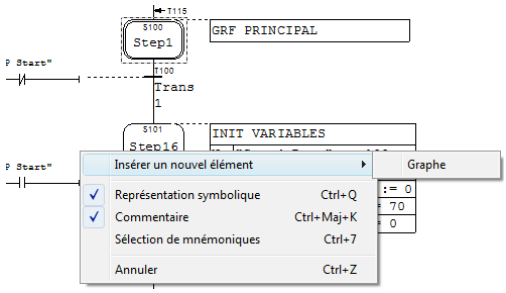
Bien que modifié tous les ans, on peut décrire la structure de ce bloc. Un grafcet principal force des bits internes (nommé DcyNomTache) à une étape et attend que le bit FinNomTache passe à 1. Une tâche peut

être constituée d'un ou plusieurs déplacements, des commandes d'actionneurs, ouverture d'une pince par exemple. Cette tâche est régie par un autre grafcet qui attend que le bit nommé DcyNomTache passe à 1, exécute les tâches dans l'ordre prévu par le grafcet, puis sa dernière étape force un autre bit nommé FinNomTache permettant par le fait au grafcet principal de continuer. Cette structure allège fortement la programmation surtout lorsque la même tâche doit être exécutée plusieurs fois.

L'INSERTION D'UN NOUVEAU GRAFCET SOUS S7GRAPH

L''insertion d'un nouveau grafcet dans le FB15, se fait simplement par un clic de gauche dans la fenêtre du grafcet.

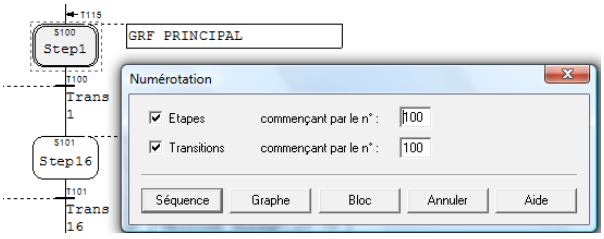
Une fois créé il faut modifier la première étape en étape initiale, ceci est obtenu par un clic droit sur l'étape, en sélectionnant Propriétés de l'objet et en cochant Etape initiale. Sans cette modification, le grafcet ne pourra pas être lancé autrement que par un forçage (solution déconseillée)



LA NUMÉROTATION DES ÉTAPES DU GRAFCET

Il n'est pas possible de renommer les différents grafcets, les grafcets sont nommés automatiquement, dans l'ordre des numéros des étapes par grafcet1, grafcet2…

Pour se repérer plus facilement il est conseillé de renuméroter les étapes des grafcets, de 200 à 299 pour le grafcet 2, de 300 à 399 pour le grafcet 3, etc. La renumérotation se fait en sélectionnant l'étape initiale et par Renumérotation / *Graphe* dans le menu Edition.



LES CALCULS

Bien qu'il soit possible de faire des calculs en S7GRAPH, il est fortement conseillé de faire ces calculs en SCL lorsqu'ils font appel à des boucles et des variables en tableau A[i]. Dans la FC1, bloc appelé par l'OB1, il est possible de faire des calculs et de récupérer sur des variables globales les résultats de ces calculs.

7. PARTIE ELECTRIQUE DU RM

LES COULEURS DES FILS

Le respect des couleurs permet de faciliter le câblage et le dépannage, par exemple, quel que soit l'état du RM, la présence du 24V doit être obligatoire présente entre un fil marron et un bleu.

- **Marron** : Alimentation 24V API
- **Rouge** : Alimentation 24V variateurs (coupée par le bouton AU)
- **Orange** : Alimentation 7,2V+des petits moteurs (coupée par le bouton AU)
- **Rouge** : Alimentation 5V+ pour Arduino
- **Bleu** : Masse commune à toutes les alimentations 24V 5V et 7,2V
- **Noir** : Pour les entrées TOR 24V (du capteur vers l'entrée API)
- **Blanc** : Pour les sorties TOR 24V (de la sortie TOR API vers la bobine d'un relais)
- **Vert et gris** : Pour les bornes des moteurs

Attention à faire des soudures correctes, pour cela il faut étamer au préalable le fil à souder et le support puis souder l'ensemble sans ajouter d'étain.

Isolez les fils avec de la gaines thermo rétractable (surtout pour les batteries). Bien attacher les fils avec des colliers rilsan.

Vérifiez avec un ohmmètre que les fils ne se touchent pas. Il est recommandé de passer un petit tournevis (hors tension) pour enlever des impuretés entre les bornes.

Toujours étamer les fils avant de les placer dans les connecteurs à vis (dominos).

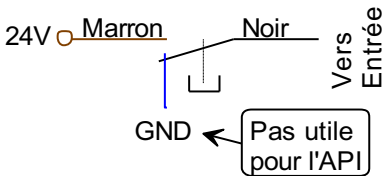
Attention la plaque supérieure doit pouvoir être soulevée pour un test d'alimentation, pour cela il est impératif que les fils soient placés de telle manière à faire une charnière.

Attachez les fils, un fil qui bouge c'est un fil qui se déconnectera.

LE CÂBLAGE DES BOUTONS, INTERRUPTEURS ET CAPTEURS

La carte d'entrée de l'API attend sur chaque borne une tension de 24V. Cette tension peut être issue d'un bouton poussoir, d'un interrupteur ou d'un capteur.

Nota : pour les entrées de la carte Arduino, il faut connecter la masse comme le montre le schéma.



En général les capteurs optoélectroniques ont besoin de la masse. La majorité des capteurs respecte les couleurs de fils (Bleu : Masse ; Marron : 24V+ ; Noir : Charge). Mais bien vérifier sur la documentation (ou directement sur le capteur) le brochage. Une inversion des trois fils occasionne la perte du capteur. Le prix des capteurs optoélectroniques oscille entre 50€ et 200€ !

LE CÂBLAGE DES MASSES

Toutes les masses doivent être reliées entre elles et sur le bâti du robot, tous les fils de masse sont de couleur bleue. Toutes les masses sont reliées, à savoir : la masse des batteries 24V API, 24V Variateurs, 7,2V et la masse Arduino.

(Voir dans Schémas Elec PARTIE COMMANDE.ppt)

8. ETUDES / AMELIORATIONS DU RM

LA COMMANDE DES MOTORÉDUCTEURS IKÉA PAR VARIATEUR DRI0018

Le variateur de moteur de tournevis IKEA a été étudié afin de remplacer les relais électromécaniques pour la commande des motoréducteurs issus de tournevis IKEA. L'objectif est de pouvoir faire tourner un motoréducteur, dans un sens ou dans l'autre, à une vitesse préréglée. Il est aussi possible de moduler la vitesse en cours de déplacement de l'actionneur, qui permet donc de ralentir avant l'arrêt et éviter l'utilisation d'amortisseur. La tension minimale fournie par le DRI018 est de 7V. Deux piles 3,6V en série peuvent être utilisées.

La carte DRI018 permet la commande de deux moteurs 15A selon deux consignes en PWM. Malheureusement il n'existe pas de carte chez VIPA qui fournit ce signal. Une carte Arduino, qui reçoit les informations de l'API, peut fournir des signaux en PWM aux deux entrées de la carte DRI018. Pour simplifier la mise en place des cartes (DRI018 et Arduino) il est fortement suggéré de faire un boitier, acceptant les deux cartes et conçu pour brancher les fils vers l'API et vers les deux moteurs. Sur ce boitier on trouvera deux potentiomètres pour régler les vitesses maxi souhaitées sur les deux moteurs.

CARACTÉRISTIQUES DE LA CARTE MOTEUR DRI018

Tension d'alimentation : 4,8 à 35V

Courant maxi : 15A/13,8V par canal, pic de 20A

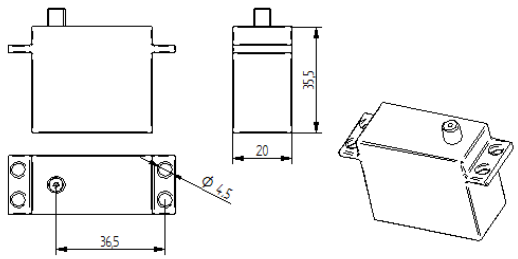
Interface pour la consigne : Signal PWM

Le variateur DRI0018 pour moteur IKEA est défini par l'assemblage qui se trouve dans Composants électriques \ Variateurs\. VariateurDRI0018&Arduino.asm

Il est constitué d'un bâti qui reçoit une carte DRI018 et une carte Arduino.

LES SERVOMOTEURS

Un servomoteur est un petit motoréducteur qui possède une électronique lui permettant d'être commandé en position très facilement par un signal PWM. Un servomoteur n'a pas beaucoup de couple, la vitesse de rotation est assez faible (de 0 à quelques tours par seconde).



La carte Arduino possède des sorties PWM ou MLI (Modulation de Largeur d'Impulsions). La commande des servomoteurs est très simple. Le programme *Test_Servo_2_GS9018* est un exemple pour la commande des servomoteurs. La bibliothèque chargée par *#include <Servo.h>* possède des fonctions qui permettent de positionner l'axe du servomoteur en indiquant l'angle en degrés. La commande *Servo ServoRotZ* permet de donner un nom au servomoteur. L'alimentation peut se faire directement sur la carte Arduino, le fil rouge sur +5V, le bleu (ou le marron) sur le GND et le jaune sur une pin de 3 à 13 (attention toutes ne sont pas en

PWM voir annotations sur la carte). La commande : *ServoRotZ.attach(3,544,2400)* attribue le le N° de pin sur laquelle le fil jaune du servomoteur est connectée (ici la pin 3) les deux paramètres permettent de configurer les angles mini et maxi. La commande *ServoRotZ.write(Angle)* permet de positionner l'axe du moteur.

SIGNAL PWM

La carte Arduino génère un signal carré d'une période de 50Hz de largeur d'impulsions variables. Selon la largeur des impulsions, le Servomoteur s'oriente de n degrés par rapport à une position initiale fixe.

Exemples : L = 1ms => -90° ; L=1,5ms => 0° ; L = 2ms => 90°

CARACTÉRISTIQUES CONSTRUCTEUR DU SERVO GS9018

Tension d'alimentation : 4,8V à 6V

Couple : 1,1 Kg.cm

Vitesse angulaire : 0,12/60°

Poids : 9g

Câblage : Alim+ : Rouge ; GND : Noir ; Fil de commande PWM : jaune

PROGRAMMATION D'UN SERVOMOTEUR EN CODE ARDUINO

La programmation d'un servomoteur est très facile, en effet la bibliothèque <servo.h> (en l'incluant dans le code par *#include <servo.h>* permet l'utilisation de fonctions très facile à utiliser. Indiquer un nom pour le servomoteur par la ligne :

Servo MonServo ;

Dans le programme setup(), configurer la pin utilisé pour le servomoteur, par la ligne :

MonServo.attach(2, 1000, 2000) ; // 2 numéro de la pin ou est câblé le servomoteur et les durées en microseconde pour les angles mini et maxi (2400 par défaut).

MonServo.write(Angle) ; // Demande à positionner le servomoteur à la valeur *Angle.*

LE SIMULATEUR DE LA CAMÉRA

LE CALCUL LES ANGLES ET LES LARGEURS DES BALISES

Le programme doit calculer, en fonction des coordonnées du RM (Xcentre, Ycentre, Tcour) et de la position des balises, les distances et les largeurs en pixels que calculerait le système de vision des balises. Dans la feuille Excel, le Robot Adverse est placé à un endroit ou l'on souhaite effectuer la simulation les variables "XRASimulé" et "YRASimulé" sont forcées à une valeur dans une VAT.

Connaissant la position de la balise, la position du point Centre (situé au milieu de l'entraxe des roues) l'orientation Tcour du robot, il faut calculer l'angle Béta et la distance DistRA qui sont calculés par le système de vision.

$$\overrightarrow{OB} = \overrightarrow{OC} + \overrightarrow{CB}$$

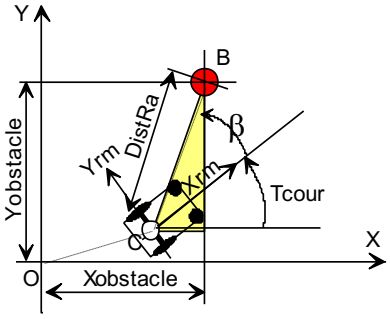
$$Xbalise.\vec{x} + Ybalise.\vec{y} = Xcentre.\vec{x} + Ycentre.\vec{y} + DistBalise.\vec{x1}$$

Avec :

$$\vec{x1} = \vec{x}.\cos(Tcour + \beta) + \vec{y}.\sin(Tcour + \beta)$$

$$\vec{y1} = -\vec{x}.\sin(Tcour + \beta) + \vec{y}.\cos(Tcour + \beta)$$

f



Dans le triangle grisé :

$$\beta + Tcour = Arc\tan\left(\frac{Ybalise - Ycentre}{Xbalise - Xcentre}\right)$$ et

$$\text{DistanceBalise} = \sqrt{(Xbalise - Xcentre)^2 + (Ybalise - Ycentre)^2}$$

CALCUL DE L'ANGLE BETA ENTRE LA DROITE PASSANT PAR LE CENTRE RM ET LA BALISE

$$\beta = Arc\tan\left(\frac{Ybalise - Ycentre}{Xbalise - Xcentre}\right) - Tcour$$

On pose pour éviter trop de calcul : *Xbalise_Xcentre* la distance sur X entre les points Centre et l'obstacle (RA). Idem pour *Ybalise_Xcentre* sur Y.

L'Arc tangente ne donne pas directement l'angle, il faut analyser les tous les cas selon les quadrants, et analyser les cas de division par 0. Voir le programme FB135.

LA LARGEUR EN PIXELS QUE LA CAMERA EST CENSÉE DONNER

La largeur dépend de la valeur de *DistanceBalise* de la distance de la focale et de la dimension de la balise (constante)

La largeur en pixels peut être calculée par l'expression déjà vue dans le paragraphe : **Erreur ! Nous n'avons pas trouvé la source du renvoi.**

$$\text{LargeurObjet}_{pixels} = \frac{\text{LargObjet}_{mm} * f_{mm}}{\text{DistanceBalise}_{mm} * \text{LargCapteur}_{mm}} = \frac{12000}{\text{DistanceBalise}_{mm}}$$

Le calcul est fait à partir de la distance calculée précédemment. La fonction de transfert qui lie la dimension en pixels avec la distance est la fonction inverse déjà expliquée précédemment. La largeur en pixels de la balise de diamètre *DiamBalise* en mm, peut être exprimée par :

$$\text{LargeurObjet}_{pixels} = \frac{CoefCam * \text{DiamBalise}_{mm}}{\text{DistanceBalise}_{mm}} \text{ Avec } CoefCam = \frac{f_{mm}}{\text{LargCapteur}_{mm}} = 240.$$

FC153 - LA FONCTION DE CONVERSION BIN_TO_GRAY

La fonction FC153 permet de convertir un octet en Gray. La mnémonique "BIN_TO_GRAY" a été affectée à cette fonction.

Exemple : titi:= "BIN_TO_GRAY"(ByteEnBin := toto") ; Si toto est un octet = 0000_1110 (14 en dec) ; titi prend la valeur 0000_1001

Ci contre la table de vérité pour convertir un nombre de 0 à 15 en Gray. Par exemple le nombre 14 est 1001 en Gray (14 en binaire : 1110).

Si on note : en[3], en[2], en[1], en[0], les 4 bits d'un mot en binaire à convertir, les 4 bits de sortie S[3], Sn[2], Sn[1],Sn[0] du mot en Gray.

Pour 4 bits les équations pour les bits sn sont :

$$Sn[3] = en[3] \qquad Sn2 = en[3] \oplus en[2]$$
$$Sn1 = en[2] \oplus en[1] \qquad Sn0 = en[1] \oplus en[0]$$

Ceci peut être étendu, par récurrence sur 8 bits :

$$Sn[7] = en[7]; Sn[6] = en[7] \oplus en[6]; \; Sn[5] = en[6] \oplus en[5]; Sn[4] = en[5] \oplus en[4];$$
$$Sn[3] = en[4] \oplus en[3]; \; Sn[2] = en[3] \oplus en[2]; \; Sn[1] = en[2] \oplus en[1]; \; Sn[0] = en[1] \oplus en[0]$$

Binaire - Gray

	en3	en2	en1	en0	sn3	sn2	sn1	sn0
0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1
2	0	0	1	0	0	0	1	1
3	0	0	1	1	0	0	1	0
4	0	1	0	0	0	1	1	0
5	0	1	0	1	0	1	1	1
6	0	1	1	0	0	1	0	1
7	0	1	1	1	0	1	0	0
8	1	0	0	0	1	1	0	0
9	1	0	0	1	1	1	0	1
10	1	0	1	0	1	1	1	1
11	1	0	1	1	1	1	1	0
12	1	1	0	0	1	0	1	0
13	1	1	0	1	1	0	1	1
14	1	1	1	0	1	0	0	1
15	1	1	1	1	1	0	0	0

LA FONCTION DE RELOCALISATION PAR LES CAPTEURS AV

Cette fonction permet de forcer les valeurs de la position courante lorsque le robot touche une bordure par l'avant ou par l'arrière.

Nota : Par soucis d'économie de mémoire, bien que pratique cette fonction est en commentaires dans le code, donc pas compilée ni chargée dan l'API et par conséquent inutilisée.

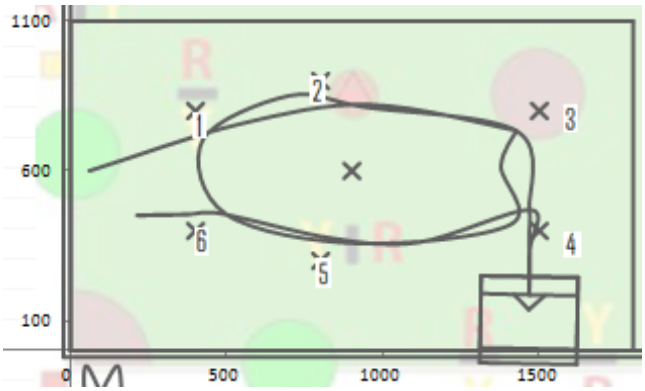
Lorsque les deux capteurs (boutons poussoirs) sont à 1 tous les deux, selon l'angle d'orientation (Tcour) que le robot possède à cet instant, les bits "RAZ PosX" ou "RAZ PosY" ou "RAZ Tcour" passent à 1, les valeurs qui faut donner à Xinit, Yinit sont dépendante des distances entre les bords et l'axe du robot et de la largeur et de la longueur de la table, constantes modifiable dans FC100 ("Dimension Table EN X", "Dimension Table EN Y", "Dist Roues-AR", "Dist Roues-AV")

Il est donc possible, qu'à partir d'une certaine distance de déplacement ("Dist De Relocal") avant relocalisation sur X et sur Y, variables calculées dans FB120 ("Dist SurX Parcourue" et "Dist SurY Parcourue" d'enclencher un cycle composé d'une orientation vers la bordure, d'une avance plus loin que la bordure tant que les deux capteurs ne sont pas à 1. Si la fonction FC127 est exécutée, la relocalisation sera effectuée.

L'inconvénient de cette solution est la perte de temps due au pivotement, à l'avancée et au recul.

Ci-contre un exemple de relocalisation sur la bordure basse.

Lorsque "Dist SurX Parcourue" ou "Dist SurY Parcourue sont > (Dist De Relocal" et que le robot est à proximité de la bonne bordure, le robot pivote pour être perpendiculaire à la bordure, avance tant que les deux capteurs ne sont pas à 1 puis recule. La fonction FC127 gère l'initialisation.

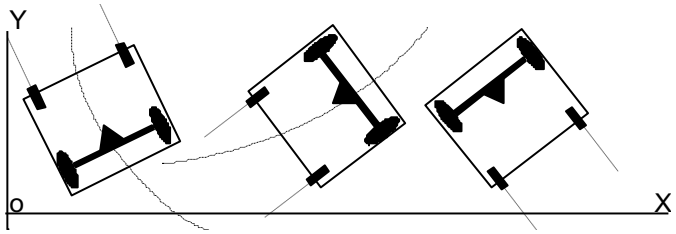


PAR DÉTECTION D'UN ÉLÉMENT FIXE DE LA TABLE DE JEUX

Selon le règlement il est possible que la table contienne des lignes qui peuvent être décelable par un capteur de couleur. Bien que cette méthode permette un temps de relocalisation très court (instantané), elle n'est pas universelle et dépend du règlement qui change tous les ans.

PAR DÉTECTION DE LA BORDURE À DISTANCE

La FC127 permet aussi de modifier la position courante du robot lorsque celui-ci est à proximité d'une bordure. Deux capteurs optoélectroniques de portée constante peuvent être installés à l'avant à droite et à gauche ("Capt Opto AVD" et "Capt Opto AVG")



L'angle Tcour lorsqu'un des capteurs passe à 1 nous indique quelle est bordure détectée, les valeurs de Xinit, Yinit sont calculées par rapport à l'angle de Tcour (non modifié).

Cette solution est très rapide (comme la précédente) mais beaucoup moins précise, elle dépend fortement de la couleur de la bordure (une couleur claire renvoie la lumière) de la vitesse du RM à l'instant t (pour une vitesse de 1m/s si "dt"=10ms, le RM a avancé de 10mm).

Attention à ne pas pouvoir prendre un objet pour une bordure. Il faut jouer sur la hauteur des capteurs dans le cas où les objets sont plus petits que la bordure. Il est possible aussi de doubler les capteurs en plaçant un en bas à 60mm et un plus haut à 80, ainsi la bordure est détectée seulement si le capteur à H60 est à 1 et celui à H80 est à 0.

LA CONVERSION DE GRAY EN BINAIRE PUR

Cette fonction permet de convertir la valeur de l'adresse envoyée par la carte Arduino de GRAY en binaire pur. Cette fonction n'existant pas en SCL, il faut la construire.

La conversion d'un nombre défini en Gray en un nombre défini en binaire pur correspond à un TD fait en semestre A. Les équations pour n bits sont ci-contre. Le tableau permet de retrouver la valeur en GRAY d'un nombre de 0 à 15. Par exemple 9 est 1101 en Gray, 15 est 1000, etc.

$$S_0 = e_0 \oplus e_1 \oplus ... \oplus e_n$$
$$S_1 = e_1 \oplus ... \oplus e_n$$
$$S_{n-1} = e_{n-1} \oplus e_n$$
$$S_n = e_n$$

	00	01	11	10
00	0	1	2	3
01	7	6	5	4
11	8	9	10	11
10	15	14	13	12

La fonction FC152 (IN en BYTE) retourne un BYTE en binaire pur d'un BYTE nommé IN d'entrée. La FC152 est affectée de la mnémonique "GRAY _TO_ BIN", pour convertir un octet TITI en GRAY dans un autre octet TOTO :

TOTO := "GRAY _TO_ BIN"(ByteEnGray:=TITI) ;

Les bits du BYTE IN sont calculés en faisant une conversion de BYTE en BOOL (ce qui correspond à prendre le bit de poids faible), du BYTE décalé du nombre de bits du poids du bit. Les bits notés S[de i=0 à i=7] correspondent aux 8 bits du BYTE

S[7] := BYTE_TO_BOOL (SHR (IN:=IN, N:=7)) ;

Puis, pour les autres bits de 6 à 0 :

S[6] := S[7] XOR BYTE_TO_BOOL (SHR (IN:=IN, N:=6)) ;

Le calcul du BYTE de la variable de retour est fait par la ligne :

FC152 := INT_TO_BYTE (BOOL_TO_INT (S[0]) + BOOL_TO_INT (S[1])*2 + BOOL_TO_INT (S[2])*4 + BOOL_TO_INT (S[3]) * 8 + BOOL_TO_INT (S[4])*16 + BOOL_TO_INT (S[5])*32 + BOOL_TO_INT (S[6])*64 + BOOL_TO_INT (S[7])*128);

(Ce qui correspond à la conversion d'un nombre en base binaire vers un nombre en décimal).

9. BIBLIOGRAPHIE / SITES INTERNET

DOCUMENTATION SIEMENS ET VIPA

S7-GRAPH - Mise en route S7-GRAPH.pdf

S7-SCL pour S7-300 et S7-400 – manuel.pdf

STEP 7 - CONT pour S7-300 et S7-400.pdf

STEP 7 - Fonctions standard et fonctions système pour S7-300 et S7-400.pdf

DOC_VIPA.pdf

LA DOCUMENTATION DE LA CARTE ARDUINO

Logiciel de programmation : http://arduino.cc/en/Main/Software

Documentation sur la carte : Arduino UNO.pdf

Utilisation de la carte Arduino : LivretArduinoCRAS.pdf

Dossier d'exemple de programme arduino : Prog Aduino de Test

http://www.dfrobot.com/wiki/index.php/DC_Motor_Driver_2x15A_Lite_(SKU:_DRI0018)

http://www.arduino.cc/en/Reference/HomePage (Instructions arduino°

LA DOCUMENTATION DES VARIATEURS ET MOTEURS

Variateur MAXON 4-Q-DC - ADS50.pdf

DC Motor Driver 2x15A_lite SCH.pdf

Servomoteur Futaba S2003.pdf

LA DOCUMENTATION DES ÉLÉMENTS MECANIQUES

HPC-T3-2013-ElementsDeGuidage.pdf

HPC-T4-2013-RouesDentees.pdf

LA DOCUMENTATION DE LA CAMÉRA PIXY

http://www.mon-club-elec.fr/pmwiki_reference_arduino/pmwiki.php?n=Main.Reference

http://www.lextronic.fr/P30095-module-de-reconnaissance-video-pixy.html

LA DOCUMENTATION DES CAPTEURS

SICK Photoelectric proximity switch WT100 Laser.pdf