Communication vers carte « Asservissement-Positionnement »

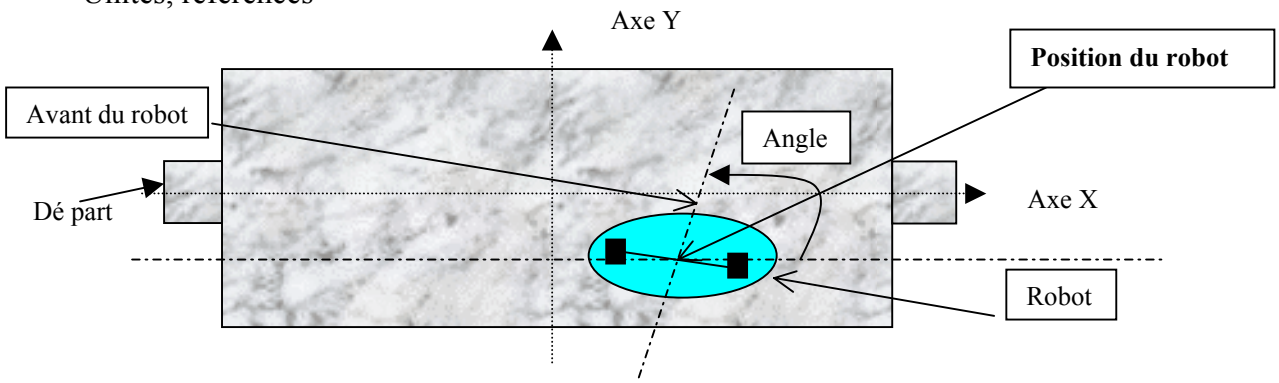
La carte garde la position courante du robot à jour. La carte exécute les ordres de trajectoires.
La communication est effectuée par le maître I2C (la carte principale).

Ici n'est défini que le contenu des messages, la correspondance entre ces messages et les trames I2C n'est pas encore définie
*toutes les **désignations** se font sur un **octet.***
*chaque **paramètre** fait par défaut un **octet***

*Le terme « **la carte** » désigne la carte asservissement*

Fonctionnement de la carte :

-La carte garde en permanence la position actuelle du robot .
-La carte reçoit comme infos materielles :
    -Nb d'impulsion des roues codeuses
    -Capteur(s) « panier »
    -Détection des lignes
-La carte utilise les roues codeuses comme info principale de position. La carte estime aussi un rayon d'erreur maximal
-Ces infos peuvent être « corrigées » suite à une demande externe
-La correction des infos de position par détection des lignes impose que la trajectoire suivent certaine règles (**non définies ici**)
-La carte exécute les déplacements d'un point à un autre en marche avant. La trajectoire est calculée en ligne droite (rotation, pour être dans la bonne direction, puis ligne droite).
-La carte peut exécuter des déplacements en marche arrière en ligne droite d'une certaine distance
-La carte peut arrêter une trajectoire si une ou plusieurs des conditions sont remplies:
    -elle est terminée
    -un capteur de « choc » à été activé
    -un pb moteur est détecté (surintensité, et/ou dérapage)
-Unités, références



Positions

Une position correspond au **milieu** de l'axe des roues codeuses. L'angle correspond à l'angle entre la droite perpendiculaire à l'axe des roues codeuses et l'axe des X.

Le point 0,0 est le centre du terrain.

Position X : entre –32000 cm et +32000 **mm (16 bits)**
Position Y : entre –32000 cm et +32000 **mm (16 bits)**
Angle : entre -90 et +90 par pas de **2°** (soit entre –180° et +180°)
Rayon erreur max : entre 0 et 255 **mm**
Vitesse : entre 0 et 2,55 m/s (en **cm/s**)
Accélération : entre 0 et 2,55 m/s² (en **cm/s²**)

-**Initialisation** de la carte
position : 0,_bout_du_terrain
angle : 0°
vitesse : 10cm/s
accélération : 40 cm/s²
filtre : P(à définir), I(à définir), D(à définir)

Messages de type ORDRE

Désignation : **GOTO**
Execute une trajectoire jusqu'au point
Paramètre(vers le slave) :
-Position X (8 bits de poids fort)
-Position X (8 bits de poids faible)
-Position Y (8 bits de poids fort)
-Position Y (8 bits de poids faible)
-Options (8 bits)
Bit 0 : recalage ligne
Bit 1 : recalage contact sur le bord du terrain (plus dur)

Désignation : **GOTO_REVERSE**
Recule d'une distance D
Paramètre(vers le slave) :
-Distance D (8 bits, en mm)

Désignation : **GOTO_PANIER**
Avance jusqu'à toucher le panier. Effectue les calculs de recalage
Paramètre(vers le slave) :
-Angle du panier(8 bits)

Messages de type ETAT

Désignation : **POSITION**
Retourne la position actuelle
Paramètre(vers le master) :
-Position X (8 bits de poids fort)
-Position X (8 bits de poids faible)
-Position Y (8 bits de poids fort)

-Position Y (8 bits de poids faible)
-Angle
-rayon erreur max

Désignation : **CHECK_BUSY**

Retourne l'état d'occupation de la carte

Paramètre(vers le master) :

BIT 7 :
BIT 6 :
BIT 5 : PB_MOTEUR
-erreur boucle d'asservissement (choc contre qqchose par ex,dérapage),surintensité
->implique un arrêt de la trajectoire (abandonnée)
->peut être dérapage total
->on peut repartir
BIT 4 : CHOC_BORD
BIT 3 : CHOC_PANIER
BIT 2 : BUSY_GOTO_PANIER *En trajet ver le panier*
BIT 1 : BUSY_TRAJET *Trajet en cours*
BIT 0 : PB_GRAVE *Incapable d'accepter d'autres ordre*

Désignation : **INFO_CHOC**

Retourne la position actuelle

Paramètre(vers le master) :

BIT 7 : capteur 7 (a définir)
BIT 6 : capteur 6 (a définir)
BIT 5 : capteur 5 (a définir)
BIT 4 : capteur 4 (a définir)
BIT 3 : capteur 3 (a définir)
BIT 2 : capteur 2 (a définir)
BIT 1 : capteur 1 (a définir)
BIT 0 : capteur 0 (a définir)

Messages de type SET

Désignation : **SET_VITESSE_TRAJ**

Initilise la vitesse de la trajectoire

Paramètre(vers le slave) :

-Vitesse
-Accélération

Désignation : **SET_VITESSE_ROT**

Initialise la vitesse de rotation

Paramètre(vers le slave) :

-Vitesse
-Accélération

Désignation : **SET_FILTRE_RAW**

Envoie les param 'bruts' du filtre PID

Paramètre(vers le slave) :

-Param P (8 bits de poids fort)
-Param P (8 bits de poids faible)
-Param I (8 bits de poids fort)
-Param I (8 bits de poids faible)
-Param D(8 bits de poids fort)
-Param D(8 bits de poids faible)

Désignation : **SET_FILTRE**

Envoie les param du filtre PID

Paramètre(vers le slave) :

-N° de preset de filtre (à définir)

Désignation : **SET_NEW_POS**

Envoie les param d'une correction de trajectoire

Paramètre(vers le slave) :

-erreur X (unité mm) poids fort
-erreur X (unité mm) poids faible
-erreur Y (unité mm) poids fort
-erreur Y (unité mm) poids faible
-erreur angle (unité °) poids fort
-erreur angle (unité °) poids faible

```
// Robot 2002
// Communication I2C vers la carte asservissement positionnement
// ------------------------------------------------------
// Définition des constantes
//
// adresse de la carte sur le bus I2C
#define ADD_CARTE_ASSER      2

// désignation du message
#define GOTO                 0
#define GOTO_REVERSE         1
#define GOTO_PANIER          2
#define CHECK_BUSY           3
#define POSITION             4
#define INFO_CHOC            5
#define SET_VITESSE_TRAJ     6
#define SET_VITESSE_ROT      7
#define SET_FILTRE           8
#define SET_FILTRE_RAW       9
#define SET_NEW_POS          10
#define INFO_CHOC            11
#define DEBUG                200

// bit de l'octet d'état
#define PB_MOTEUR                    5
#define CHOC_BORD                    4
#define CHOC_PANIER                  2
```

```
#define BUSY_GOTO_PANIER        2
#define BUSY_TRAJET             1
#define PB_GRAVE                0

// bits des capteurs (constantes à définir)
#define CAPTEUR_7               7
#define CAPTEUR_6               6
#define CAPTEUR_5               5
#define CAPTEUR_4               4
#define CAPTEUR_3               3
#define CAPTEUR_2               2
#define CAPTEUR_1               1
#define CAPTEUR_0               0

// preset des filtres PID
#define FILTRE_STD              0
```

(amené à être séparé dans plusieurs fichiers)

Annexe : Exemples et possibilités actuelles (12/01)

Initialisation :

SET_VITESSE_TRAJ
100 (cm/s)
SET_VITESSE_ROT
50 (cm/s)

Trajet vers le point 200,100 (mm)

GOTO
0 (poids fort X)
200 (poids faible X)
0 (poids fort Y)
100 (poids faible Y)
0 (pas d'options)

Le robot va tourner sur place pour s'orienter vers son point de destination. Il va effectuer le trajectoire en ligne droite. Le bit BUSY_TRAJET sera à 1 pendant toute la durée

While (BUSY_TRAJET) ; // attente de fin de trajet

Aller vers un panier (100,50 représente un point DEVANT le panier, un peu avant quelques cm)

GOTO
0 (poids fort X)
100 (poids faible X)
0 (poids fort Y)
50 (poids faible Y)
0 (pas d'options)

Le robot va tourner sur place pour s'orienter vers son point de destination. Il va effectuer le trajectoire en ligne droite. Le bit BUSY_TRAJET sera à 1 pendant tte la durée

While (BUSY_TRAJET) ; // attente de fin de trajet

GOTO_PANIER
22 (angle /2 soit 45°/2 == 22)

Le robot va tourner sur place pour s'orienter vers le bon angle de destination. Il va effectuer le trajectoire en ligne droite jusqu'à ce qu'il rencontre le panier. Le bit BUSY_GOTO_PANIER sera à 1 pendant toute la durée

Performances :

Erreur de position inconnue (on ne sait pas où on est) : sur une distance droite de 1,8m (en X par ex) une erreur de +ou- 2cm quasi indépendante de la vitesse

Erreur de position connue (on ne va pas exactement où on veut) : sur une distance de 1,8m (en X par ex) pas d'erreur pour des vitesses < 0.2m/s , erreur de + ou – 7cm pour : vitesse 1m/s accélération 1m/s. Cette erreur peut être contournée en fractionnant la trajectoire (on oblige des points d'arrêt dans ce cas).

A priori meilleure utilisation :

-Fractionner les grandes trajectoires.(les eviter est encore mieux : on perd temps, précision,etc)

-Pour annuler l'erreur de position inconnue : faire une trajectoire allant un peu moins loin que l'objectif, faire une deuxieme trajectoire vers l'objectif.