RoBoNav - Étalonnage des moteurs

Jean Fruitet

Introduction

Pour se déplacer et s'orienter, chaque bouée dispose de deux moteurs immergés qui tournent en sens contraire l'un de l'autre.
Pour un déplacement en ligne droite ces moteurs doivent être parfaitement équilibrés ce qui n'est jamais le cas en sortie d'usine.
Il convient donc de passer par une phase d'étalonnage lors de toute mise en œuvre car il est à craindre que celui-ci ne soit pas conservé.
Nous proposons un algorithme assez sommaire pour réaliser cet équilibrage ; ce n'est pas un PID.

Algorithme

Après la mise à l'eau, par construction la bouée s'oriente face au vent, dans la direction `COG0`, fournie par le compas embarqué.

L'action sur le commutateur **SWB** de la radiocommande fait passer la bouée en mode étalonnage des moteurs.

On cherche à déterminer les compensations `COMPMTR_G` et `COMPMTR_D` qu'il faut ajouter aux commandes assignées aux moteurs pour maintenir un cap lors d'un déplacement en ligne droite.

Les compensations s'expriment en microseconde puisque le protocole est PWM.
On va appliquer une vitesse lente identique aux moteurs et mesurer le changement d'orientation par unité de temps, en commençant par une mesure tous les 1/100ème de seconde.

Soit `COG` l'orientation mesurée par le compas de bord ;

```
// Initialisation
INCMTR0 = 1500; // en microsecondes la valeur de la poussée initiale sur chaque moteur ;
```

```
INCMTR = 200; // en microsecondes l'incrément initial de poussée ;
SIGNMTR = 0; // l'orientation initiale appliquée ;
MAXMESURE = 100; // le nombre maximal de mesures par étalonnage ;
COMPTMESURE = 0; // le compteur de mesures ;
MAXSENSIBILITE = 2; //  la valeur finale de sensibilité en degré
SENSIBILITE = 8; // en degré, la valeur initiale de la sensibilité ; c'est le demi cône autour
de la direction COG0

COMPMTR_D = 0;  // Aucune compensation initiale à droite
COMPMTR_G = 0;  // ni à gauche

// Mesure de la direction opposée au vent
COG0 = MesureCompas();

// Poussée lente identique sur chaque moteur
AvanceMoteurGauche(INCMTR0);
AvanceMoteurDroite(INCMTR0);

// Boucle de test
Tant que (SENSIBILITE > MAXSENSIBILITE) {
   Tant que ( COMPTMESURE < MAXMESURE)     {
           COG = MesureCompas();
           Si (COG < COG0 - SENSIBILITE) { // On compense vers la droite
                SIGNMTR = 1;
           }
           Sinon si (COG > COG0 + SENSIBILITE) { // On compense vers la gauche
                SIGNMTR = -1;
           }
           Sinon {
                SIGNMTR = 0;
           }
           COMPMTR_D = COMPMTR_D + SIGNMTR * INCMTR;
           COMPMTR_G = COMPMTR_G - SIGNMTR * INCMTR;
```

```
            AvanceMoteurGauche(INCMTR0+COMPMTR_G);
            AvanceMoteurDroite(INCMTR0+COMPMTR_D);
            delay(100); // Pas de temps entre les mesures en millisecondes
      }
      // On augmente la sensibilité
      SENSIBILITE = SENSIBILITE / 2;
      COMPTMESURE = 0;
      // On diminue la poussée de compensation
      INCMTR = INCMTR / 2;
}
// En sortie les compensations COMPMTR_D  et COMPMTR_G sont positionnées.
```

Désormais ces valeurs sont systématiquement ajoutées aux valeurs affectées aux moteurs.

SWB = OFF // Fin de calibration

Implantation en C++

```
// Initialisation
int INCMTR0 = 1500; // microsecondes la valeur de la poussée initiale sur chaque moteur ;
int INCMTR = 200; // microsecondes l'incrément initial de poussée ;
int SIGNMTR = 0; // l'orientation initiale appliquée ;
int MAXMESURE = 100; // le nombre de mesures pour l'étalonnage ;
int COMPTMESURE = 0; // le compteur de mesures ;
int MAXSENSIBILITE = 2; //  la valeur finale de sensibilité en degré
int SENSIBILITE = 8; // en degré, la valeur initiale de la sensibilité ; c'est le demi cône autour de la
direction COG0

int COMPMTR_D = 0;       // Aucune compensation initiale à droite
int COMPMTR_G = 0;       // ni à gauche

float COG0;        // Cap initial
float COG;         // Cap mesuré
```

```
// Mesure de la direction opposée au vent
COG0 = MesureCompas();

// Poussée lente identique sur chaque moteur
AvanceMoteurGauche(INCMTR0);
AvanceMoteurDroite(INCMTR0);


// Boucles de mesure
while (SENSIBILITE > MAXSENSIBILITE) {     // de 8° à 2° de part et d'autre de la direction recherchée
   while (COMPTMESURE < MAXMESURE) {       // 100 Mesures espacées de 100 millisecondes
            // Mesure de l'orientation de la bouée
            COG = MesureCompas();
            if (COG < COG0 - SENSIBILITE) { // On compense vers la droite
                  SIGNMTR = 1;
            }
            else if (COG > COG0 + SENSIBILITE) { // On compense vers la gauche
                  SIGNMTR = -1;
            }
            else {
                  SIGNMTR = 0;
            }
            COMPMTR_D = COMPMTR_D + SIGNMTR * INCMTR;
            COMPMTR_G = COMPMTR_G - SIGNMTR * INCMTR;
            AvanceMoteurGauche(INCMTR0+COMPMTR_G);
            AvanceMoteurDroite(INCMTR0+COMPMTR_D);

            delay(100); // Micro secondes
      }
     // On augmente la sensibilité
      SENSIBILITE = SENSIBILITE / 2;
      // On diminue l'incrément
      INCMTR = INCMTR / 2;
      COMPTMESURE=0;
}
// En sortie les compensations COMPMTR_D  et COMPMTR_G sont positionnées.
```

Adaptations possibles

Jouer sur le nombre d'itérations `MAXMESURE`, les valeurs de poussées `INCMTR0` et `INCMTR` et sur le délai entre deux itérations.

Si l'algorithme ne converge pas (trop d'oscillations autour de la direction à suivre) on peut modifier la fabrication de la bouée en lui ajoutant un voile de quille profond dans le sens avant arrière et/ou implanter un algorithme de type PID.