TP2
Application sur robot mobile

1 Introduction

Objectif pédagogique : programmer une loi de commande sur un microcontrôleur en garantissant le fonctionnement temps réel.

Application : asservissement de vitesse d'un robot mobile télécommandé depuis un smartphone via une communication WiFi. Ce TP réutilise directement les résultats du TD2.

Livrables : un compte-rendu synthétique sera à rendre 1 semaine au plus tard après le TP. Les programmes seront à transmettre **en fin de séance** sous la forme d'une **archive ZIP à déposer sur Moodle**. **Pour le compte-rendu**, **ajoutez-le à votre archive et redéposez-la au même endroit**. Si vous le souhaitez, vous pourrez améliorer vos programmes (les commenter par exemple) à ce moment là.

2 Réalisation progressive

2.1 Génération des signaux PWM

Utilisez le fichier base_IF4_TP2-BO.zip.

On utilise pour cela 4 unités *Motor Control Pulse Width Modulator* (MCPWM) de l'ESP32 (cf doc complète Espressif[1] et doc de mise en œuvre[2]). Les signaux PWM prennent les valeurs 0 ou 3,3V. L'ESP32 possède 16 canaux PWM indépendants qui peuvent être assignés à quasiment n'importe quel pin GPIO. La configuration de la PWM peut être réalisée très simplement sur l'ESP32 avec l'environnement de développement Arduino. Vous utiliserez le module *ledc* (disponible par défaut) avec seulement 2 fonctions : `ledcAttach(), analogWrite()`.

Pour générer un signal PWM, par exemple sur la broche GPIO23, il faut :

- au moment de la configuration :
  - choisir la broche GPIO (*pin*) 23 pour générer le signal PWM ;
  - choisir la fréquence de la PWM ;
  - choisir la résolution de la largeur d'impulsion entre 1 et 16 bits ;
- pendant l'application :
  - appliquer un rapport cyclique.

1 Voir https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/peripherals/mcpwm.html
2 Voir https://docs.espressif.com/projects/esp-idf/en/v4.3/esp32/api-reference/peripherals/mcpwm.html

Un exemple de code de configuration et d'usage est donné en figure 1 :

```
/**
 * Initialise une PWM
 * cf https://github.com/espressif/arduino-esp32/blob/master/docs/en/migration_guides/2.x_to_3.0.rst#ledc
 */
void init_motor_pwm(uint8_t pin) {
  ledcAttach(pin, PWM_FREQ, PWM_RESOLUTION);
  analogWrite(pin, 0);
}
```

Figure 1: exemple de code pour gérer une sortie PWM (mise ici à 0)

Sur le robot, **les 4 sorties GPIO associées aux 2 moteurs sont** (inversion des sorties entre les deux moteurs pour que les roues tournent dans le même sens en marche avant) **:**

- **Moteur droit : MRF = 33; //Right Forward & MRB = 32; //Right Backward**
- **Moteur gauche : MLF = 26; //Left Forward & MLB = 25; // Left Backward**

Veillez à laisser le signal dans le sens opposé (avant/arrière) à 0.

Avant de lancer vos programmes, mettez le robot sur cales ! (pour éviter qu'il tombe de la table)

Proposez et testez un programme qui **:**

- attend 1 seconde ;
- fait avancer le robot pendant 500 ms ;
- attend 500 ms ;
- fait reculer le robot pendant 500 ms ;
- s'arrête définitivement.

Ce programme pourra être monotâche ou, mieux, bi-tâches (dans ce cas, ne vous préoccupez pas de la protection des données inter-tâches à ce stade).

Expliquez dans votre rapport votre approche.

Ce programme sera à transmettre **en fin de séance de TP dans une archive à déposer sur Moodle. Il faudra l'appeler «BO_noms_binome».**

2.2 Mesure de la vitesse de rotation

Les broches des signaux des 2 canaux de chaque codeur sont lisibles sur :

- Encodeur droit : **SRA = 35; SRB = 34;**
- Encodeur gauche : **SLA = 14; SLB = 27;**

1. En vous basant sur l'approche proposée au TD2 et ce que vous avez vu au TP1, **réalisez un programme multitâche** qui détermine toutes les **100 ms** la vitesse de rotation de la **roue gauche** moteur et l'envoie au PC sur le port série pendant 10 secondes.
2. Complétez votre programme pour que la vitesse des 2 roues s'affiche simultanément.

Si besoin de protection de ressources partagées, validez expérimentalement d'abord votre programme sans protection, puis avec...

Expliquez dans le rapport votre approche.

Ce programme sera à ajouter à l'archive précédente, à déposer sur Moodle en fin de séance. Il faudra **l'appeler « BO_Vitesse_noms_binome ».**

2.3 Asservissement de vitesse

En vous basant sur le TD2, programmez l'asservissement de la vitesse de la roue gauche.

Commencez par un correcteur PROPORTIONNEL, avant de passer au P.I.

Quand vous lancez le programme mais **faîtes attention à laisser suffisamment de longueur de câble USB** pour que le robot ne tire pas dessus.

Pour le tester, appliquez une consigne variant par pallier de **500 ms :**
0, puis 2.5 rad/s, puis -2.5 rad/s puis 0.

Affichez sur le port série la consigne de vitesse, la vitesse mesurée et le signal de commande : une ligne par période, valeurs séparées par une tabulation. Enregistrez ces informations dans un fichier au format TSV. Affichez-le sous forme graphique dans EXCEL ou MATLAB.

Une fois l'asservissement validé expérimentalement, complétez ce programme pour asservir les 2 roues. Expliquez dans le rapport votre approche et vos résultats.

Si besoin de protection de ressources partagées, validez expérimentalement d'abord votre programme sans protection, puis avec...

Ce programme sera à ajouter à l'archive précédente. Il faudra l'appeler « BF_noms_prenoms ».

2.4 Intégration de la communication WiFi

Utilisez le fichier base_IF4_TP2-WiFi.zip.

1. Reprenez la tâche de contrôle en boucle fermée du programme précédent.
2. Pour configurer la communication, modifiez le SSID de votre robot et le mot de passe (constantes en début de programme) pour éviter que vos voisins vous *hackent* votre robot. L'initialisation du hotspot WiFi de votre robot est réalisé par la fonction `wifi_start(ssid, password)` appelée dans la fonction `setup()`.
3. Créez une fonction (`update_desired_speeds()`) qui modifie les vitesses désirées des moteurs pour permettre au robot d'aller en avant/en arrière/à gauche/à droite et de s'arrêter en fonction de la valeur réceptionnée par la fonction de communication WiFi `communicate_with_phone()` (voir TD2).
3. Dans votre tâche de communication, appelez en permanence la fonction `communicate_with_phone()` pour répondre aux requêtes du navigateur de votre téléphone.
4. Compilez et *flashez* l'ESP32 puis ouvrez votre téléphone, connectez-vous sur le réseau WiFi qui aura le nom et le mot de passe que vous lui aurez attribué à l'étape 1.
5. Ouvrez un navigateur en utilisant l'adresse IP (w.x.y.z) **indiquée dans la console série** (http://192.168.**X**.1), X=4 la plupart du temps, sinon regarder dans les paramètres de votre connexion WiFi.
6. Testez et déboguez votre programme.

Si besoin de protection de ressources partagées, validez expérimentalement d'abord votre programme sans protection, puis avec...

Ce programme sera à ajouter à l'archive précédente. Il faudra l'appeler « Remote ».

3 Aller plus loin

Quelques idées (par ordre de difficulté croissante) :

- ajouter un bouton STOP ;
- afficher sur le téléphone la vitesse des roues ;
- régler la vitesse d'avance depuis le téléphone ;
- programmer des manœuvres (1/2 tours, créneaux …) ;
- ...