Projet de
Systèmes Bouclés (Partie 1)

ROBOT : le contrôleur de moteur DC

Présentation Lors de l'ECE Cup, vous avez à gérer un robot qui est dirigé par deux moteurs DC qui entrainent les roues de ce robot. Le but de ce projet est d'utiliser vos connaissances en Systèmes Bouclés afin de gérer la commande en vitesse des moteurs et position : lorsque la consigne de vitesse de rotation des moteurs est envoyée, les moteurs du robot doivent atteindre la bonne valeur de façon précise et rapide, avec peu d'oscillations. Pour cela, nous coderons et utiliserons un correcteur PID. Ainsi, si les deux moteurs sont bien réglés, on peut envoyer une consigne unique de vitesse de rotation des roues, et le robot devrait avancer de façon (quasi) parfaite en ligne droite avec la bonne vitesse demandée !

Travail préliminaire demandé : Partie 1

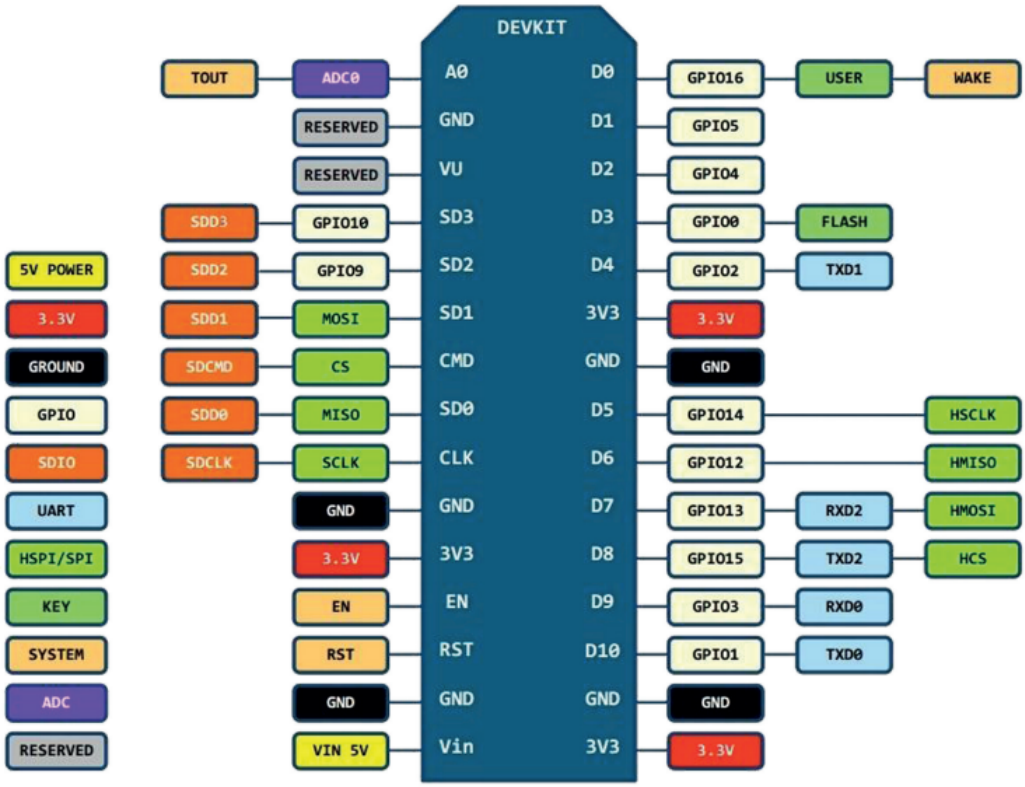
Dans un fichier robot.ino (IDE Arduino) :

1. **Coder le démarrage / l'arrêt de chaque moteur, dans un sens ou l'autre.**
2. **Fixer les roues codeuses sur les axes des moteurs et fixer les fourches avec de la colle forte ou tout d'abord avec de la patafixe afin de faire des essais.**
3. **Coder alors le fichier robot.ino de façon à mesurer la vitesse angulaire (en tour/seconde par exemple) de rotation de chaque roue. On pourra procéder par étapes, tout d'abord visionner dans le port série les différents « ticks » ; réfléchir à la façon d'utiliser éventuellement une fonction d'interruption etc… La librairie SimpleTimer peut vous être utile, et étant données les difficultés de trouver la bonne version (ne pas utiliser celles fournies de base sur IDE Arduino), on vous donne le fichier SimpleTimer.zip (à installer vous-même dans IDE Arduino) avec ce document.**

Conseils :

- Attention à ne pas fixer les fourches perpendiculairement aux moteurs, mais bien parallèlement, sinon en replaçant le robot sur la bonne face, il ne pourra plus rouler à cause des fourches qui toucheraient le sol…
- Utiliser les PIN VCC, GND et D0 de chaque fourche.
- Brancher la fourche 1 sur D8 de l'ESP 8266 par exemple (ce qui correspond à la PIN 15 dans IDE Arduino) , cf annexe plus bas; et la fourche 2 sur D9.
- Inclusion de la bibliothèque au projet sur l'IDE Arduino : Tout d'abord supprimer dans /Arduino/Library/ tout répertoire /SimpleTimer qui pourrait déjà exister. Puis, depuis le menu : Croquis > Inclure une bibliothèque > Ajouter la bibliothèque .ZIP. Fermer puis redémarrer l'IDE pour valider cette inclusion.

<u>Annexe 1</u> : Schématique ESP8266 – équivalence avec Arduino



Par exemple, si on utilise la PIN D1 de l'ESP8266, comme elle correspond à GPIO 5, on utilisera donc le numéro de PIN 5 dans le code Arduino IDE (pinMode(5,INPUT ou OUTPUT) dans la fonction setUp() etc…)

<u>Annexe 2</u> : Fonctions d'interruption

Les fonctions d'**interruption** permettent la suspension temporaire de l'exécution d'un programme afin d'exécuter une tâche prioritaire. Elles sont déclenchées à l'intérieur du microcontrôleur par des modules internes hardware dédiés. Dans un code "classique", le déclenchement d'une fonction se fait lorsqu'un test (if(...)) renvoi une valeur positive. Cependant, avant d'arriver à ce test if(...), il faut parfois attendre plusieurs lignes de code, ce qui implique un retard de traitement, comme illustré sur la figure ci-dessous.

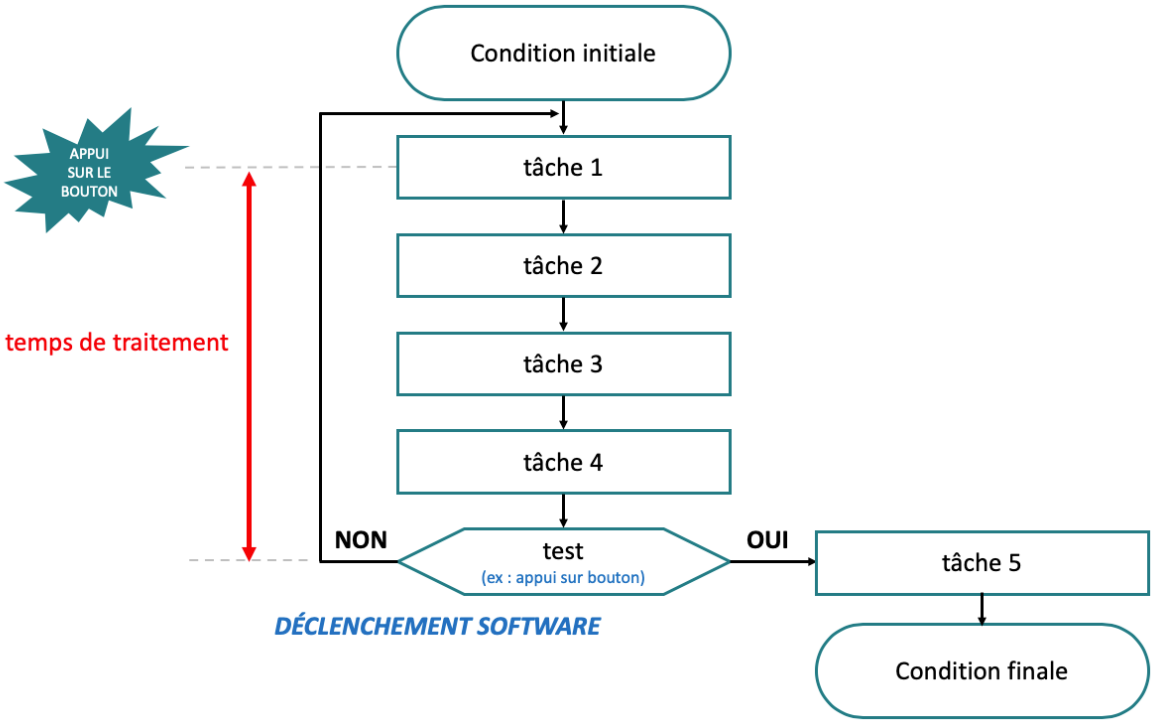


Figure 1 : Exemple d'algorigramme de test d'appui sur bouton classique : le traitement ne se fait qu'une fois la ligne de code de test atteinte.

Le recours à des fonctions d'interruption permet alors de répondre à cette problématique. De façon automatique (test avec des circuits logiques internes au microcontrôleurs), des fonctions dites d'interruption peuvent être déclenchées en réponse à un signal électrique (hardware), comme illustré sur la Figure ci-dessous :

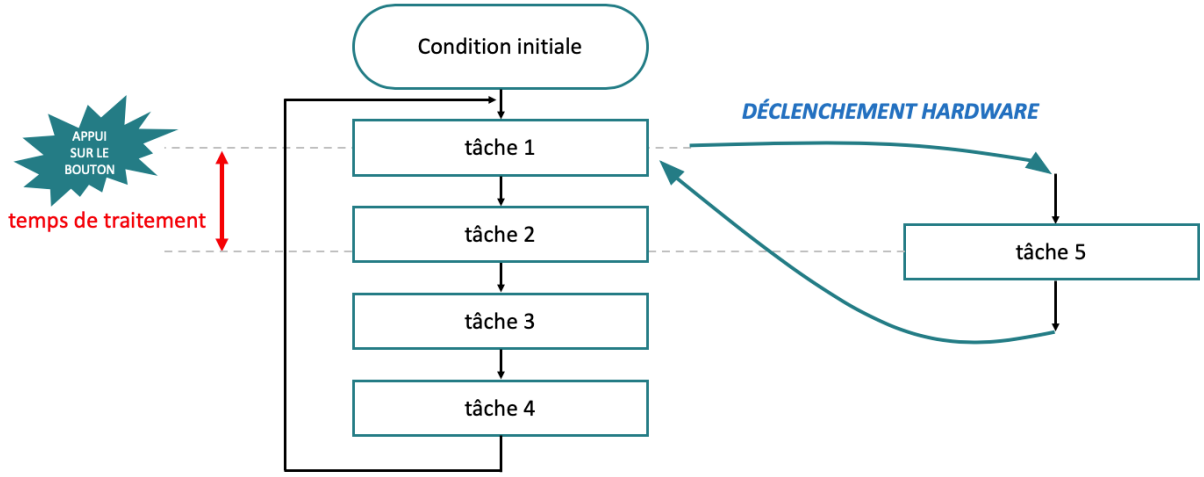


Figure 2 : Exemple d'algorigramme de test d'appui sur bouton avec fonction d'interruption : une fois le bouton appuyé, la tâche en cours est interrompue et le traitement est lancé. Le temps de traitement est donc optimisé.

Ce type de fonction est implémentable sur les microcontrôleurs dont l'ATMEGA328P, microcontrôleur de l'Arduino Nano. Un exemple de code permettant d'inverser l'état logique d'une pin est présenté sur

la Figure ci-dessous. Le code commence classiquement par paramétrer le microcontrôleur (procédure loop()) avant d'entrer dans une boucle infinie : la void loop(). Lorsqu'il y a un changement d'état sur la pin 0, la fonction d'interruption blink() est appelée et l'état logique de la variable state est inversé. Une fois la fonction d'interruption exécutée, le programme revient dans la void loop().

```
int pin = 13;
volatile int state = LOW; // déclartion d'une variable volatile

void setup()
{
  pinMode(pin, OUTPUT);
  attachInterrupt(0, blink, CHANGE); // attache l'interruption externe n°0 à la fonction blink
}

void loop()
{
  digitalWrite(pin, state); // la LED reflète l'état de la variable
}

void blink() // la fonction appelée par l'interruption externe n°0
{
  state = !state; // inverse l'état de la variable
}
```

Figure 3 : Code permettant d'inverser l'état logique d'une pin à l'aide d'une routine d'interruption.

Les fonctions d'interruption sont nécessairement déclenchées par une détection hardware (interne au microcontrôleur, non synchronisée sur l'exécution linéaire du code). Une des sources d'interruption possible est le timer : un compteur qui s'incrémente à chaque front montant d'un multiple du signal d'horloge du microcontrôleur, auquel cas on parle de fonction d'échantillonnage, dont un exemple est présenté sur la Figure ci-dessous :

```
#include <SimpleTimer.h>
SimpleTimer timer;
const int led = 2;
unsigned int time = 0;
const int frequence_echantillonnage = 20;

void setup()
{
  pinMode(led,OUTPUT); // la pin 2 est une sortie
  timer.setInterval(1000/frequence_echantillonnage, light); // appel de la fonction "light" toutes les 100 ms
}

void loop()
{
}

void light()
{
  digitalWrite(led,HIGH); // allumage de la LED
  delay(10); // attendre 10 ms
  digitalWrite(led,LOW); // éteindre la LED
}
```

Figure 4 : Code permettant d'allumer une LED pendant 10 ms toutes les 100 ms en utilisant une fonction d'échantillonnage.