PONT EN H

Esteban Chevalier-Drevon
Matyas Ricci
Groupe F

Exercices : Le pont en H

Exercice 1 : Calcul de valeur moyenne et rapport cyclique

Soit un moteur commandé par le signal PWM suivant :

Question 1 : Quelle est la valeur moyenne aux bornes du moteur pour un rapport cyclique de 40% ?

L'expression de la tension moyenne V_{moy} en fonction du rapport cyclique α est : $V_{moy} = V_{max} * \alpha$

A.N : Vmoy = 12*0,4 = 4,8V

Question 2 : Quel doit être le rapport cyclique pour avoir une tension de 8v aux bornes du moteur ?

Pour avoir une tension de 8V aux bornes du moteur, le rapport cyclique doit être de 66,67%.
(100*8)/12

Question 3 : Si la fréquence du signal est de 4 kHz, quelle est la durée de tH ?
L'expression du Temps haut tH est : tH = T * Rapport cyclique

Pour un rapport cyclique de 66,67% et T = 1/f, on a :
tH = 1/4000 * 0,6667 = 200µs

La durée de tH est de 200µs pour 4kHz

Exercice 2 : Table de Vérité

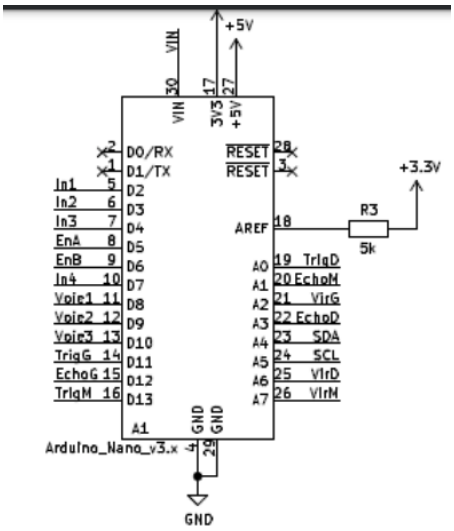
Chaque moteur sera piloté à l'aide de trois signaux de commande : Avant, Arrière et Pwm.

Ces signaux seront générés par l'Arduino pour faire varier

- Le sens de rotation du moteur (suivant l'état (0 ou 1) d'Avant / Arrière)
- La vitesse de rotation du moteur (suivant le rapport cyclique de Pwm)

Question 1 : En vous aidant du « Arduino NANO Pinout Diagram », faire figurer les numéros de broche sur la carte Arduino que vous choisirez. **Attention : les signaux PWM ne sont pas disponibles sur toutes les broches**.

Pour le pont en H, nous choisirons les broches D2, D3, D4 ET D7 pour les sorties TOR et les pins D5 et D6 pour les sorties PWM.



Attribution des broches de l'Arduino Nano

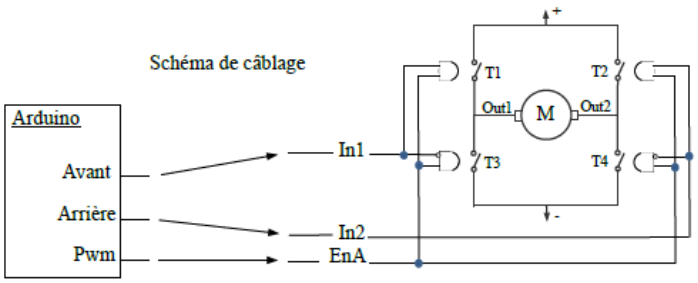
Question 2 : Remplir la table de vérité ci-dessous en indiquant pour chaque combinaison d'entrée :

- L'état (0 ou 1) de la commande des différents transistors du pont en H :

  « 0 » → le transistor est bloqué (interrupteur ouvert)

  « 1 » → le transistor est passant (interrupteur fermé)

- L'état du moteur → trois possibilités : Sens+, Sens- et Arrêt.



In1	In2	EnA	T1	T2	T3	T4	Etat moteur
0	0	0	0	0	0	0	Arrêt
0	0	1	0	0	1	1	Arrêt
0	1	0	0	0	0	0	Arrêt
0	1	1	0	1	1	0	Sens -
1	0	0	0	0	0	0	Arrêt
1	0	1	1	0	0	1	Sens +
1	1	0	0	0	0	0	Arrêt
1	1	1	1	1	0	0	Arrêt

Question 3 : En déduire les équations booléennes des trois commandes en fonction de In1, In2 et EnA :

Sens+=In1.$\overline{IN2}$.EnA

Sens-=In2.$\overline{IN1}$.EnA

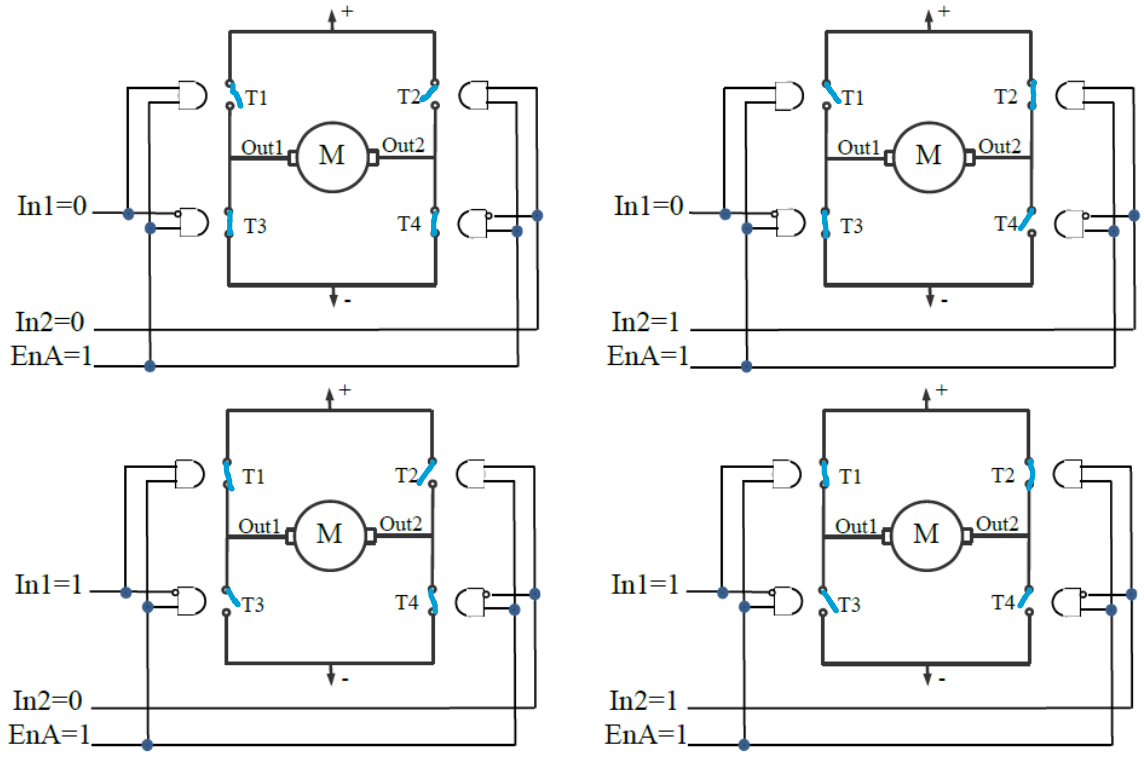
Pour la commande Arrêt, on vous demande d'utiliser un tableau de Karnaugh :

EnA\In1In2	00	01	11	10
0	*1*	*1*	**1**	**1**
1	1	0	1	0

Arrêt = $\overline{ENA}$+IN1.IN2+$\overline{IN1}$.$\overline{IN2}$ = IN1 ʘ IN2

Exercice 3 : Etats transistors

Question 1 : État ouvert ou fermé de chaque transistor en fonction des commandes appliquées.



Exercice 4 : Programmes "Commande Moteur"

Les programmes doivent être enregistrés dans un dossier Commande_Moteur en précisant le numéro du programme : Ils doivent être copiés dans ce document et les instructions et la structure du programme doit être expliqués.

Programme n°1 : Commande des deux moteurs avec un seul sens de rotation :

Ecrire un programme qui lance quatre séquences successives d'arrêt/démarrage (Arrêt de 1 sec et moteur vitesse max 1 sec) suivies d'un arrêt définitif.

```
#define In1 2
#define In2 3
#define In3 4
#define In4 7
#define EnA 5 // Moteur droit
#define EnB 6 // Moteur gauche

unsigned long int t0;     // Variable de temps afin de rempalcer le delay de manière non blocant.

void setup() {
    pinMode(In1, OUTPUT); // Tous les transistors de notre pont en H
    pinMode(In2, OUTPUT); // sont des transistors de sorties.
    pinMode(In3, OUTPUT); // Ont les définit donc comme tel dans notre programme;
    pinMode(In4, OUTPUT);
    pinMode(EnA, OUTPUT); // EnA et EnB sont nos pins générant notre PWM
    pinMode(EnB, OUTPUT); // afin de faire varier la vitesse des moteurs.

    digitalWrite(In2, LOW);
    digitalWrite(In1, HIGH); // Ont met les transistors 1 et 4 comme étant passant
    digitalWrite(In3, LOW);  // ce qui a pour effet de sélectionner le sens de rotation
    digitalWrite(In4, HIGH); // des deux moteurs en marche avant.
}

void loop() {
    static int counter = 0;                 // Compteur du nombre de boucles effectués.
    if (millis() - t0 > 1000) {             // Toutes les secondes, on rentre dans la boucle
      t0 = millis();                        // afin de mettre en marche ou non les moteurs.
        if (counter == 1 || counter == 3) { // Lors de la deuxième entrée dans la 1ère boucle
            analogWrite(EnA, 255);          // et de la 4ème, on met en marche les moteurs
            analogWrite(EnB, 255);
        } else {                            // Sinon, on met les rapports cycliques à 0 (Arrêt des moteurs)
            analogWrite(EnA, 0);
            analogWrite(EnB, 0);
        }
        counter++;                          // On incrémente notre compteur.

    }
}

```

Programme n°2 : Commande des deux moteurs et rotation dans les deux sens :

Ecrire un programme qui lance quatre séquences successives d'arrêt/démarrage (Arrêt pendant 1sec et moteur à la vitesse max pendant 1 sec) en changeant de sens à chaque arrêt, suivies d'un arrêt définitif.

```
#define In1 2
#define In2 3
#define In3 4
#define In4 7
#define EnA 5 // Moteur droit
#define EnB 6 // Moteur gauche

unsigned long int t0;     // Variable de temps afin de rempalcer le delay de manière non blocant.

void setup() {
    pinMode(In1, OUTPUT); // Tous les transistors de notre pont en H
    pinMode(In2, OUTPUT); // sont des transistors de sorties.
    pinMode(In3, OUTPUT); // Ont les définit donc comme tel dans notre programme;
    pinMode(In4, OUTPUT);
    pinMode(EnA, OUTPUT); // EnA et EnB sont nos pins générant notre PWM
    pinMode(EnB, OUTPUT); // afin de faire varier la vitesse des moteurs.
}

void loop() {
    static bool seDeplace = 0;                  // Booléen indiquant si oui ou non on restera à l'arrêt lors du passage dans la boucle.
    static bool sens = 0;                     // Booléen indiquant si le robot doit avancer ou reculer lors du passage dans la boucle.
    static int counter = 0;                   // Compteur du nombre de séquences effectués.
    if (millis() - t0 > 1000 && counter <=4) { // Ici, la fonction millis() - t0 nous permet d'avoir notre delay non bloquant de 1s.
                                              // On vérifie aussi par la même occasion le nombre de séquences effectués (On doit en faire 4)
      t0 = millis();                          // Reset du compteur de millisecondes permettant notre delay non bloquant.
        if (seDeplace) {                        // Si le booléen est à 1, alors on doit se déplacer.
            analogWrite(EnA, 255);            // Paramétrage de la vitesse du moteur droit à 100%.
            analogWrite(EnB, 255);            // Paramétrage de la vitesse du moteur gauche à 100%.
        } else {
            analogWrite(EnA, LOW);            // Sinon, si seDeplace est à 0 celà signifie que l'on doit s'arrêter lors de cette séquence.
            analogWrite(EnB, LOW);
            counter++;                        // On incrémente notre compeur signifiant que l'on a effectué une séquence.
        }
        if (sens && seDeplace) {                  // Si seDeplace est à 1 alors on doit faire bouger le robot, et si sens est à 1
            digitalWrite(In2, LOW);           // on déplace le robot en marche avant.
            digitalWrite(In1, HIGH);          // On paramètre le pont en H afin que les deux moteurs soient
            digitalWrite(In3, LOW);           // configurés de manière à avancer.
            digitalWrite(In4, HIGH);
            sens = !sens;                     // On inverse le sens de déplacement du robot pour la prochaine boucle.
        } else if (seDeplace) {                   // Sinon si seDeplace est à 1 mais que sens est à 0, alors le robot doit reculer.
            digitalWrite(In2, HIGH);          // On paramètre donc le pont en H afin que les deux moteurs soient
            digitalWrite(In1, LOW);           // configurés de manière à reculer.
            digitalWrite(In3, HIGH);
            digitalWrite(In4, LOW);
            sens = !sens;                     // On inverse le sens du déplacement du robot pour la prochaine boucle
        }
        seDeplace = !seDeplace;                       // On inverse le booléen seDeplace. Celà signifie que si lors du passage dans la boucle
                                              // le robot s'est déplacé, alors dans le prochain passage de boucle le robot sera dans
                                              // sa phase d'arrêt de 1seconde.
    }
}
```

Programme n°3 : Une fonction pour commander les moteurs :

Ecrire une fonction générique permettant de piloter chaque moteur (droit ou gauche) dans n'importe quel sens (avant ou arrière) et en gradation de vitesse (0->100%)

```
void cmd_motor(bool motor, bool direction = 0, short power = 0) {
    float kfactor = 1;
    if (motor) { // Moteur droit
        power = (power * 2.55) * kfactor;
        if (direction) { // marche avant
            digitalWrite(In1, LOW);          // On paramètre le pont en H afin que le moteur droit
            digitalWrite(In2, HIGH);         // soit configuré de manière à avancer.
            analogWrite(EnB, power);         // On applique une PWM de rapport cyclique définit par la variable power pour le moteur droit.
        } else { // Marche arrière
            digitalWrite(In2, LOW);          // On paramètre le pont en H afin que le moteur droit soit
            digitalWrite(In1, HIGH);         // configuré de manière à reculer.
            analogWrite(EnB, power);         // On applique une PWM de rapport cyclique définit par la variable power pour le moteur droit.
        }
    } else { // Moteur gauche
        power = (power * 2.55);
        if (direction) { // Marche avant
            digitalWrite(In3, HIGH);         // On paramètre le pont en H afin que le moteur gauche soit configuré
            digitalWrite(In4, LOW);          // de manière à avancer
            analogWrite(EnA, power);         // On applique une PWM de rapport cyclique définit par la variable power pour le moteur droit.
        } else { // Marche arrière
            digitalWrite(In3, LOW);          // On paramètre le pont en H afijn que le moteur gauche soit configuré
            digitalWrite(In4, HIGH);         // de manière à reculer.
            analogWrite(EnA, power);         // On applique une PWM de rapport cyclique définit par la variable power pour le moteur gauche.
        }
    }
}
```

Le prototype de la fonction est :

void cmd_motor (bool motor, bool direction, short power);

Avec :

motor : le numéro du moteur
Direction : avant ou arrière
Power : le rapport cyclique en %

Pour clarifier l'écriture du code concernant la commande des moteurs on peut définir les constantes nommées suivantes :

#define Left 0
#define Right 1
#define Forw 1
#define Backw -1
#define Stop 0

Par exemple, les deux appels suivants feront tourner le robot modérément sur la droite ! :

cmd_motor (Left, Forw,85);
cmd_motor (Right, Forw,50);

Les deux appels suivants feront quant à eux tourner le robot sévèrement sur la gauche ! :

cmd_motor(Left, Stop,0) ;
cmd_motor(Right, Forw,80) ;

TEST D'INTEGRATION N°1

FAIRE AVANCER LE ROBOT TOUT DROIT DANS UN COULOIR DEFINI PAR L'ENSEIGNANT.

Le robot a avancé tout droit dans un couloir défini par l'enseignant.