Classe de problèmes MECATRO-2

Asservir en rotation un moteur à courant continu

Analyser la structure d'une chaîne fonctionnelle
Générer un programme et l'implanter dans le système cible
Modifier un programme pour faire évoluer le comportement du système

Modéliser - Simuler - Expérimenter

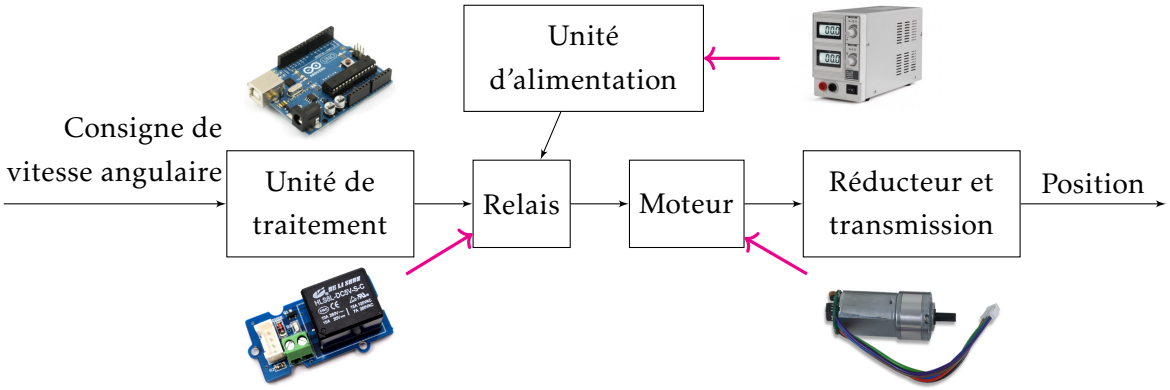
L'objectif de ce Tp est :

- d'acquérir des informations via l'interface homme machine pour élaborer une consigne
- d'acquérir des informations via les capteurs situés sur les éléments de la chaîne d'énergie
- d'asservir les mouvements de l'actionneur.

1 Présentation

1.1 Permier Tp

Le premier Tp a permis d'établir la connexion entre l'unité de traitement et le préactionneur pour moduler l'énergie vers l'actionneur.



1.2 Objectifs

Dans ce Tp, on s'intéresse dans un premier temps à l'acquisition d'information via les broches analogiques.

L'acquisition d'information permet :

- d'une part, de régler la consigne via un bouton poussoir (commande TOR) ou via un potentiomètre (commande analogique numérisée sur 2^{10} niveaux).
- d'autre part, de réguler l'actionneur en maîtrisant ses paramètres cinématiques

Dans un second temps, l'unité de traitement envoie un ordre quantifié à destination du préactionneur.

1.3 Rappel sur la carte *Arduino-Uno*

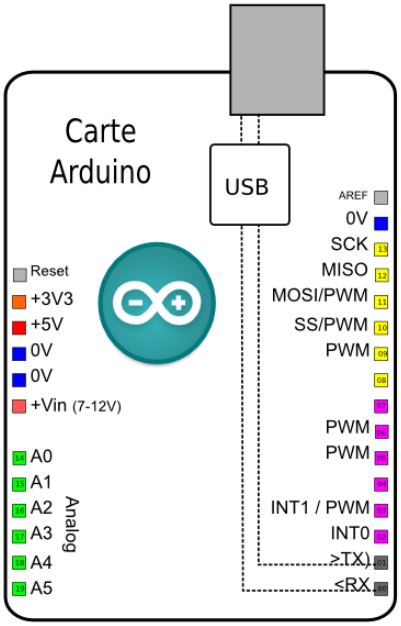
La carte **Arduino-Uno** est un exemple d'unité de traitement. L'élément central de la carte est le microcontrôleur. Son rôle est de traiter les données et d'assigner les sorties.
Les autres éléments principaux sont le connecteur USB, les 14 broches **numériques**, les 6 broches **entrées analogiques** et de 4 broches **sorties de tensions de référence**.

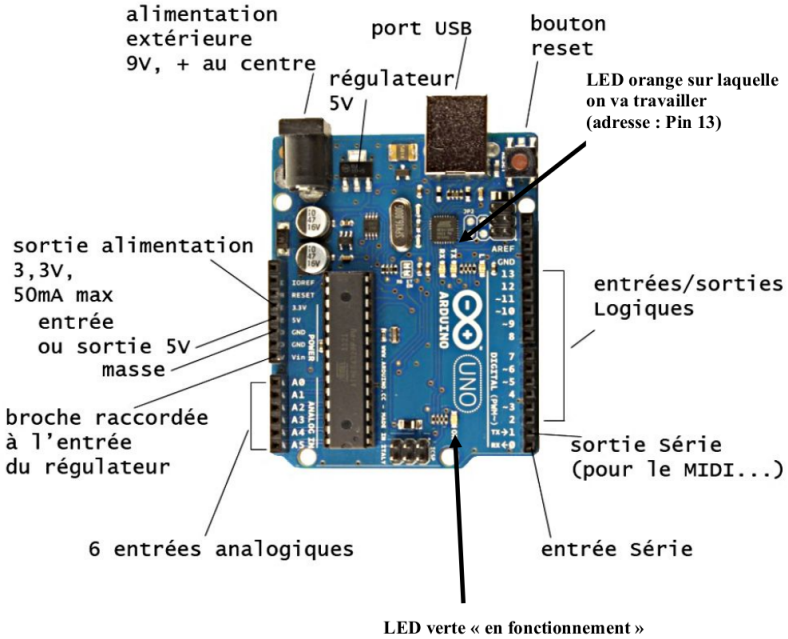
Le connecteur USB assure la communication de la carte avec le micro-ordinateur, et permet son alimentation en l'absence d'une autre source d'alimentation.

Les broches numériques peuvent être configurées **entrée** ou **sortie**. Lorsqu'elles sont configurées **entrée**, ce sont obligatoirement des entrées TOR (Tout Ou Rien, 1 ou 0). Pour être à 1, la broche doit être au potentiel 5 V et pour être à 0, au potentiel 0 V. Pour une **sortie**, il y a deux configurations possibles : soit elle est TOR, soit elle est numérique et codée sur 8 bits (1 octet), soit 256 possibilités (de 0 à 255).

REMARQUE : seules les broches repérées par le symbole ~ (3, 5, 6, 9, 10 et 11) peuvent être numériques, les autres broches ne peuvent être que TOR.

Les broches **entrées analogiques** permettent de mesurer à la cadence du quartz de cadencement l'évolution de grandeurs physiques. Une entrée analogique de la carte Arduino est capable d'enregistrer un signal de 0 à 5 V délivré par un capteur ou par un élément d'interface homme machine (par exemple un potentiomètre). Ce signal est numérisé sur 10 bits (de 0 à 1023). Les broches **sorties de tensions de référence** permettent d'alimenter des composants extérieurs (par exemple des capteurs) en 3,3 ou 5V.





2 Commande simple d'un moteur avec bouton poussoir

2.1 Objectifs

L'objectif de cette activité est de réaliser la commande en rotation d'un moteur en appuyant sur un bouton poussoir avec une alimentation externe .

2.2 Première étape : acquisition tout ou rien de l'état d'un bouton poussoir

- En transformant, le `pinMode` d'une broche en `INPUT` au lieu d'`OUTPUT`, acquérir son niveau avec la commande `digitalRead`
- Écrire un programme qui permet d'allumer/éteindre la LED avec le même bouton poussoir : un impulsion sur ce bouton fait passer la LED d'un état allumer à éteint et inversement.

```
// Affectation des broches
const int pin_LED = 12;
const int pin_BP = 13;

// Gestion des variables
int etat_bouton = LOW;
int etat_LED = LOW;


void setup() {
  // put your setup code here, to run once:
  pinMode(pin_LED, OUTPUT);
  pinMode(pin_BP, INPUT);
  }

void loop() {
  etat_bouton = digitalRead(pin_BP);
  if (etat_bouton){
    etat_LED = !etat_LED;
    digitalWrite(pin_LED, etat_LED);
  }
  delay(200);
}
```

Ce code est disponible **ici**.

2.3 Deuxième étape : Commande du relais avec le bouton poussoir

Après avoir réussi à allumer/éteindre une LED sur la breadbord en fonction des actions sur le bouton poussoir, la LED est remplacée par un relais.

- Relier le relais à la carte Arduino FIG 1 (ex : 11)
- Relier un côté d'un bouton poussoir à la tension 5 V et l'autre côté du bouton à une résistance de 10 kΩ. Relier l'autre côté de la résistance au GND.
- Relier une broche entrée numérique (ex : 8) entre le bouton poussoir et la résistance. Ainsi, si l'interrupteur est fermé, on mesure 5 V, s'il est ouvert en mesure 0 V. L'interrupteur ne fait pas court-circuit, grâce à la résistance.
- Écrire le code correspondant (téléchargeable **ici**). Pour cela :
  - Définir deux broches numériques en tant que variables `com_moteur` et `appui_bouton`
  - Définir une variable entier `etat_bouton` et une autre `etat_moteur`
  - Configurer la broche correspondante à la variable `com_moteur` en tant que sortie et la broche correspondante à la variable `appui_bouton` en tant que entrée ;
  - Dans la fonction `loop`, écrire le code correspondant au fonctionnement souhaité (disponible **ici**)

    La variable `etat_bouton` permet la `lecture` de l'état de la broche numérique liée à la variable `appui_bouton`.

    Si l'entrée `etat_bouton` est à 1, on met la sortie `com_moteur` à 1, sinon on la met à 0.

```
1  const int com_moteur = 12;
2  const int appui_bouton = 8;
3  int etat_bouton = LOW;
4  int etat_moteur = LOW;
5
6  void setup() {
7    pinMode(com_moteur, OUTPUT);
8    pinMode(appui_bouton, INPUT);
9  }
10
11 void loop() {
12   etat_bouton=digitalRead(
       appui_bouton);
13   if (etat_bouton){
14     etat_moteur = !etat_moteur
15     digitalWrite(com_moteur,
         etat_moteur);
16     delay(200)
17   }
18 }
```

- Compiler et téléverser le code dans la carte
- Tester le fonctionnement du système.
- Réaliser le câblage du moteur avec une alimentation extérieure en réglant la bonne tension (voir tension nominale du moteur - 9V) ;

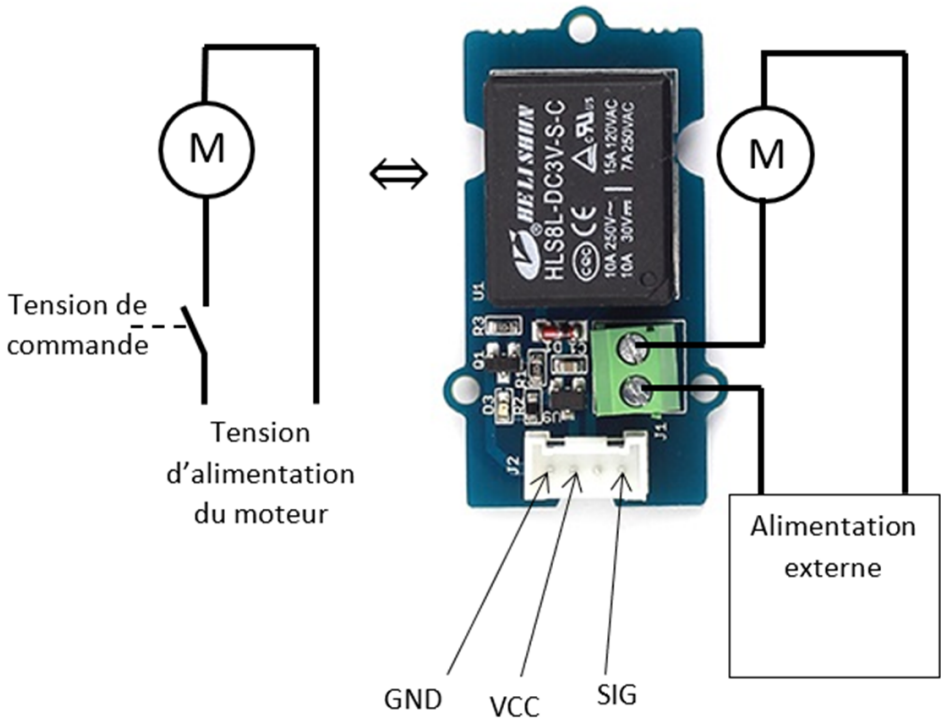
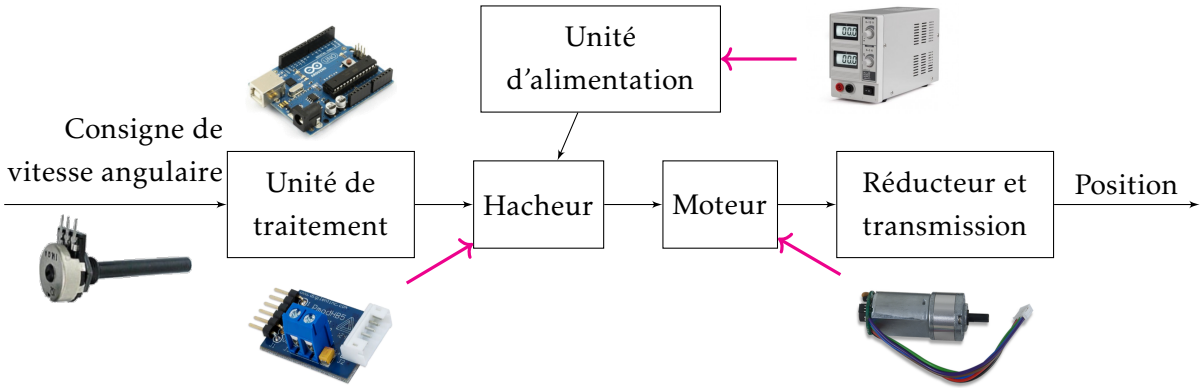


FIGURE 1 – Câblage du relais

3 Commande d'un moteur avec variation de vitesse

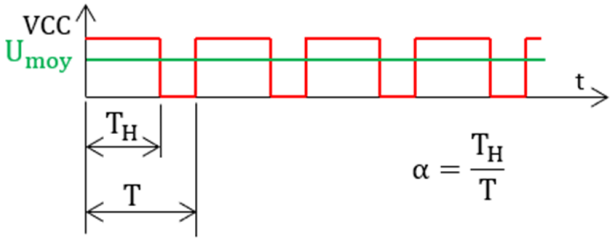
Dans l'activité précédente, la vitesse de rotation du moteur était fixe et proportionnelle à la tension d'alimentation. L'objectif de cette activité est de réaliser la commande d'un moteur de manière à faire varier sa vitesse de rotation avec un potentiomètre.

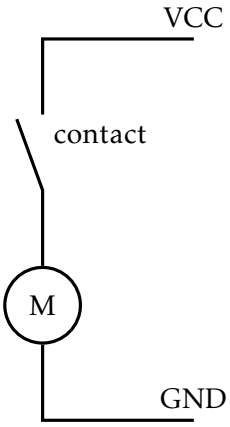


Pour cela on utilise un hacheur de tension en préactionneur. Son principe peut être décrit à l'aide du schéma électrique ci-contre :

- Lorsque le contact est fermé, le moteur est sous tension maximale VCC avec GND = 0 V.
- Lorsque le contact est ouvert, le moteur est hors tension.

En jouant sur les fréquences d'ouverture et de fermeture du contact (hautes fréquences), la tension d'alimentation du moteur peut être décrite par le graphe suivant :





On peut définir une période T, un rapport cyclique α (avec $0 \leq \alpha \leq 1$) et une tension moyenne U_{moy}.

Naturellement, cette tension en créneaux est filtrée par les bobines du moteur, ce qui équivaux à une alimentation à la tension moyenne. La vitesse de rotation du moteur est proportionnelle à la tension U_{moy} et donc au rapport cyclique α.

3.1 Première étape : acquisition numérique sur 1024 niveaux de l'état d'un potentiomètre

On s'intéresse cette fois à une valeur analogique, codée sur 1024 niveaux. On utilisera donc les broches A_i, quatre fois plus précises que les broches i. On utilise alors la commande `pinMode(A0, INPUT)` pour sélectionner en acquisition la broche A0. Le code est disponible **ici**.

Pour afficher les valeurs de la broche, on peut écrire sur le moniteur série avec, comme déclaration dans le `setup`, `Serial.begin(9600)` pour écrire à une vitesse de 9600 bauds.

Si `valeur` est une variable de type `long`, on peut écrire la `valeur` sur le moniteur série avec `Serial.println(vale`

On cherche donc :

- dans un premier temps, à obtenir sur le moniteur les valeurs liées à l'état du potentiomètre
- dans un second temps , à régler l'intensité de la LED en fonction de l'état du potentiomètre

Nous sommes donc près à envoyer au hacheur un signal analogique proportionnel à l'état du potentiomètre.

```
1  // Affectation des broches
2  const int Potar =A0;
3
4  // Déclaration des variables
5  long niveau = 0;
6
7  void setup() {
8    pinMode(Potar, INPUT);
9    Serial.begin(9600);
10 }
11
12 void loop() {
13   niveau = analogRead(Potar);
14   Serial.println(niveau);
15 }
```

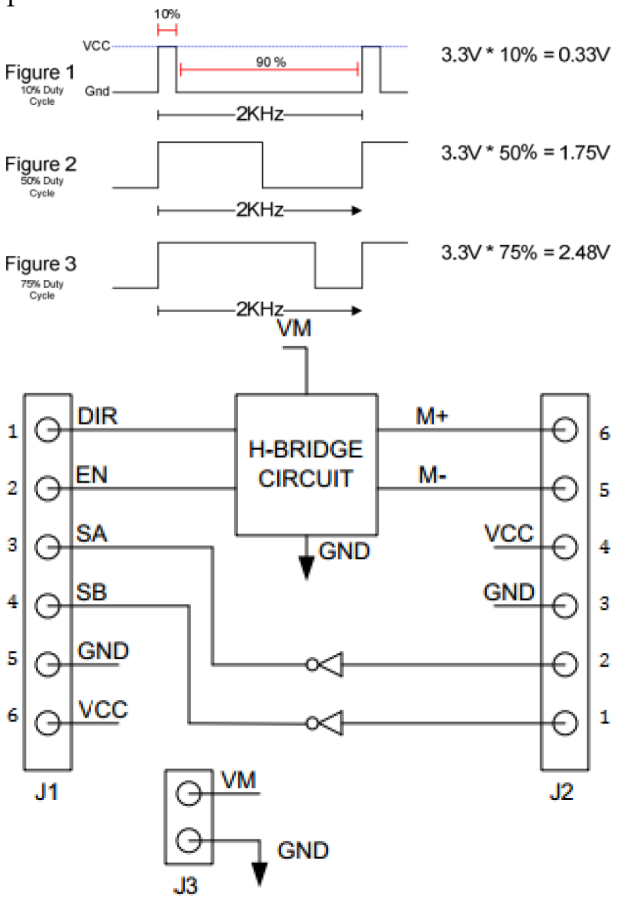
3.2 Deuxième étape : Commande du hacheur avec le potentiomètre.

Header J1 (pin 1 on the top)		
Pin	Signal	Description
1	DIR	Direction pin
2	EN	Enable pin
3	SA	Sensor A feedback pin
4	SB	Sensor B feedback pin
5	GND	Power Supply Ground
6	VCC	Positive Power Supply (3.3/5V)

Header J2 (pin 1 on the bottom)		
Pin	Signal	Description
1	SB	Sensor B feedback pin
2	SA	Sensor A feedback pin
3	GND	Power Supply Ground
4	VCC	Positive Power Supply (3.3/5V)
5	M+	Motor positive pin
6	M-	Motor negative pin

Pour cette activité on va utiliser le motoréducteur IG220053X00069R et le composant PmodHB5.

Le motoréducteur est équipé de 6 fils qui se connectent directement sur le composant PmodHB5 qui a 6 broches correspondantes au niveau du connecteur J2. Il y a aussi 6 broches au niveau du connecteur J1. Le connecteur J3 permet l'alimentation externe du moteur.




```arduino
const int com_moteur = 11;
const int appui_bouton = 8;
int etat_bouton = LOW;
int sens = LOW;
int vitesse =0;
const int consigne_vitesse = 0;
void setup() {
  pinMode(com_moteur, OUTPUT);
  pinMode(appui_bouton, INPUT);
  pinMode(consigne_vitesse,INPUT);
}

void loop() {
  vitesse = analogRead(consigne_vitesse)
     /4;
  etat_bouton=digitalRead(appui_bouton);
  if (etat_bouton){
    analogWrite(com_moteur, vitesse);
  }
  else{
    analogWrite(com_moteur, 0);
  }
}
```

La broche 1 (DIR) de J1 doit définir le sens de rotation du moteur, elle doit être reliée à une sortie TOR de la carte de commande. Si cette broche est à 0, le moteur tourne dans un sens, si elle est à 1, le moteur tourne dans l'autre sens.

La broche 2 (EN) de J1 doit définir la valeur de la vitesse de rotation du moteur, elle doit être reliée à une sortie numérique de la carte de commande. Les sorties numériques de la carte Arduino sont codées sur 8 bites soit 256 possibilités. Lorsque la sortie est à 0, le moteur ne tourne pas, lorsque la sortie est à 255, la vitesse du moteur est maximale, entre ces deux valeurs, la vitesse est proportionnelle.

Les broches 5 et 6 doivent assurer l'alimentation du composant sous une tension de 5 V, et doivent donc être reliées aux broches d'alimentation de la carte de commande.

Le code est disponible **ici**.

4 Asservissement en vitesse du moteur

4.1 Objectif

L'objectif de cette activité est d'évaluer la vitesse de rotation du moteur à l'aide des signaux fournis par les codeurs.

4.2 Fonction d'interruption

Pour réaliser l'asservissement en vitesse du moteur, il faut connaître la vitesse de rotation du moteur. On compare alors la consigne à la mesure de la vitesse de rotation du moteur et on élabore un signal écart. Cet écart est parfois corrigé/amplifié et donne l'ordre au préactionneur de moduler l'énergie à destination de l'actionneur.

Nous avons donc besoin dans un premier temps de mesurer la vitesse de rotation du moteur. Or le moteur possède deux codeurs magnétiques en quadrature de phase. Compter les passages devant le codeur magnétique permet d'obtenir une information sur la vitesse de rotation du moteur.

Un programme Arduino se compose de deux fonctions essentielles : `setup` et `loop`. La fonction `setup` est appelée une seule fois lorsque le programme commence et la fonction `loop` est appelée en permanence.

Il est possible d'arrêter brièvement et temporairement la fonction `loop` à chaque fois qu'un événement (changement d'état d'une entrée TOR) se produit. Pour cela il faut utiliser les broches numériques d'interruption, les broches 2 et 3, numérotées 0 et 1.

On propose **ici** un programme simple pour prendre en main la fonction `attachInterrupt(num, fonction, quand)`. Avec un bouton poussoir, on souhaite allumer ou éteindre une LED beaucoup plus simplement qu'au premier TP, avec l'aide de ce tuto :

```
1  int Led = 13; // Pin de la LED
2  int etat = LOW; // Etat de la LED
3
4  void setup()
5  {
6   pinMode(Led, OUTPUT);
7   attachInterrupt(0, clignote, RISING); // Lors d'un changement de son etat, on
       attache a la broche 2 la fonction clignote
8   Serial.begin(9600);
9  }
10
11 void loop() {
12   digitalWrite(Led, etat);
13   Serial.println(etat);
14 }
```

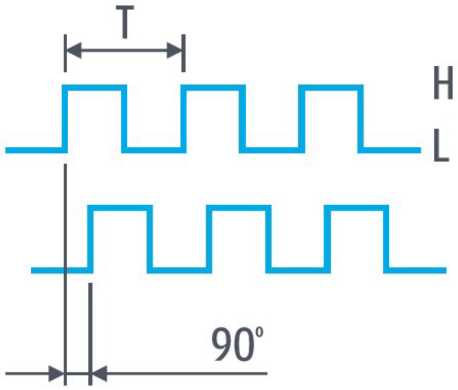
```
15
16 void clignote() // fonction liee a l'interruption externe 0
17 {
18   etat = !etat; // prend le complementaire
19 }
```

On note que la fonction `clignote` est définie en dehors du `setup` et du `loop`. Elle ne doit pas avoir d'argument et ne rien renvoyer.

4.3 Codeurs magnétiques et compteur

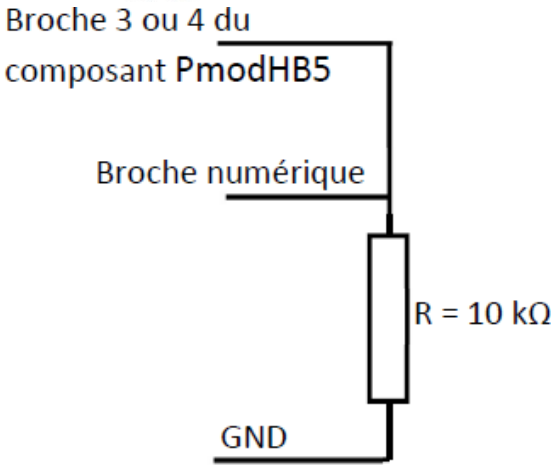
Le motoréducteur est équipé de deux codeurs magnétiques qui délivrent chacun un signal de 0 ou 5 V.

Ces codeurs sont décalés de 90°, le graphe ci-contre donne le profil des signaux délivrés lorsque le moteur tourne.



Les broches 3 et 4 délivrent ces signaux. On cherche ici à réaliser un compteur à partir des codeurs magnétiques sur l'arbre moteur.

Les broches 3 et 4 du composant PmodHB5 délivrent chacune un signal de 0 ou 5 V. Il faut donc relier chacune de ces broches à une broche entrée TOR de la carte Arduino de manière à pouvoir compter le nombre d'impulsions à des intervalles de temps réguliers.



De cette façon, on peut déterminer la vitesse de rotation du moteur. Pour un seul codeur, si on compte les fronts montants et descendants du signal, et que l'on a évalué 6 impulsions en 10 ms, cela veut dire que le moteur effectuerait 3 tours en 10 ms et donc qu'il tournerait à 300 tr/s soit 18 000 tr/min. Pour 2 codeurs, il y aura 4 impulsions par tour.

A chaque changement d'état des signaux des codeurs, on va incrémenter un compteur.

REMARQUE : Le fait d'avoir deux codeurs peut permettre de détecter le sens de rotation. On peut aussi décrémenter le compteur.

À chaque fois que la fonction `comptageA` est appelée, le niveau (haut ou bas) de la voie A sera comparé au niveau de la voie B. En fonction de cela, on détectera le sens de rotation et on incrémentera ou décrémentera le compteur d'impulsion. On peut faire alors de même avec une fonction `comptageB` sur le niveau de la voie B. On gagne ainsi en résolution.

- Effectuer le câblage des broches 3 et 4 du composant PmodHB5 avec les broches d'interruption de la carte ;
- Définir les broches numériques 2 et 3 en tant que variables `signA` et `signB` .
- Définir une variable entière `Nimpulsion` et l'initialiser à 0 .
- Écrire deux fonctions `comptageA` et `comptageB` permettant d'incrémenter ou de décrémenter un compteur d'impulsion (ces fonctions seront placées avant la fonction `setup` et seront appelées à chaque interruption) .

On peut obtenir le code ci-dssous en **là**.

```
#include "digitalWriteFast.h"

// Affectation des broches
const int Pin_EN = 10;
const int Pin_DIR = 8 ;
const int Pin_Cap_A = 2;
const int Pin_Cap_B = 3;


// Variables
long vitesse = 255 ;
long Nimpulsion = 0;


void setup() {
  pinMode(Pin_EN, OUTPUT);
  pinMode(Pin_DIR, OUTPUT);
  pinMode(Pin_Cap_A, INPUT);
  pinMode(Pin_Cap_B, INPUT);
  Serial.begin(115200);
  attachInterrupt(0, comptageA, CHANGE);
  digitalWrite(Pin_EN, vitesse);
}

void loop(){
  Serial.println(Nimpulsion);
}


void comptageA(){
  if (boolean digitalReadFast(Pin_Cap_A) == boolean digitalReadFast(Pin_Cap_B)){
    Nimpulsion = Nimpulsion - 1;
  }
  else {
    Nimpulsion = Nimpulsion + 1;
  }
}
```

4.4 Mesure numérique de la vitesse

Il faut maintenant calculer la vitesse de rotation du moteur.

Pour cela nous avons besoin d'un `timer` qui déclenchera un calcul toutes les 10 millisecondes. En effet, puisqu'on sait incrémenter/décrémenter un compteur grâce aux capteurs magnétiques, il est possible de compter le nombre d'incréments reçus sur un temps donné.

Pour un seul codeur, si on compte les fronts montants et descendants du signal, et que l'on a évalué 6 impulsions en 10 ms, cela veut dire que le moteur effectuerait 3 tours en 10 ms et donc qu'il tournerait à 300 tr/s soit 18 000

tr/min. Pour 2 codeurs, il y aura 4 impulsions par tour.

Le fabricant annonce une vitesse en sortie de réducteur de 150 tr/min à vide sous une tension de 6 V avec une réduction de 52,734 ce qui correspond à une vitesse moteur de 7910 tr/min soit 131,8 tr/s. Pour un ordre de commande de hacheur de 255 et une alimentation de 6 V le moteur tournera donc à 7910 tr/min.

Comme nous allons alimenter le hacheur avec 9 V, le moteur devrait pouvoir tourner à presque 200 tr/s. Nous essaierons de réguler la vitesse à 100 tr/s.

Il convient donc maintenant d'obtenir une mesure numérique de la vitesse en tr/s.

Chaque tour du moteur permet d'obtenir 4 incréments grâce aux fonctions d'interruption `ComptageA` et `ComptageB`. Ainsi `Nimpulsion/4` permet d'obtenir le nombre de tour de moteur sur la période de comptage.

Comme nous allons lancer une fonction `calcul_vitesse` toutes les 10 ms (tous les 100ièmes de seconde), il suffira de multiplier par `100 Nimpulsion/4` pour connaître la vitesse en tr/s.

```
#include <digitalWriteFast.h>
#include <MsTimer2.h>

// Affectation des broches
const int Pin_EN = 10;
const int Pin_DIR = 8 ;
const int Pin_Cap_A = 2;
const int Pin_Cap_B = 3;

// Variables
long mes_vit = 0 ;
long Nimpulsion = 0;


void setup() {
  pinMode(Pin_EN, OUTPUT);
  pinMode(Pin_DIR, OUTPUT);
  pinMode(Pin_Cap_A, INPUT);
  pinMode(Pin_Cap_B, INPUT);
  digitalWrite(Pin_EN, rap_cyc);
  Serial.begin(115200);
  attachInterrupt(0, comptageA, CHANGE);
  attachInterrupt(1, comptageB, CHANGE);
  MsTimer2::set(10, calcul_vitesse); // 500ms period
  delay(10);
  MsTimer2::start();
}

void loop(){
  analogWrite(Pin_EN, 255);
}


```

```
34 void comptageA(){
35   if (boolean digitalReadFast(Pin_Cap_A) == boolean digitalReadFast(Pin_Cap_B)){
36     Nimpulsion = Nimpulsion - 1;
37   }
38   else {
39     Nimpulsion = Nimpulsion + 1;
40   }
41 }
42
43 void comptageB(){
44   if (boolean digitalReadFast(Pin_Cap_A) == boolean digitalReadFast(Pin_Cap_B)){
45     Nimpulsion = Nimpulsion + 1;
46   }
47   else {
48     Nimpulsion = Nimpulsion - 1;
49   }
50 }
51
52
53 void calcul_vitesse(){
54   mes_vit = 100*Nimpulsion/4;
55   Serial.println(mes_vit);
56   Serial.println("␣");
57   Nimpulsion = 0;
58 }
```

Ce code est disponible **ici**.