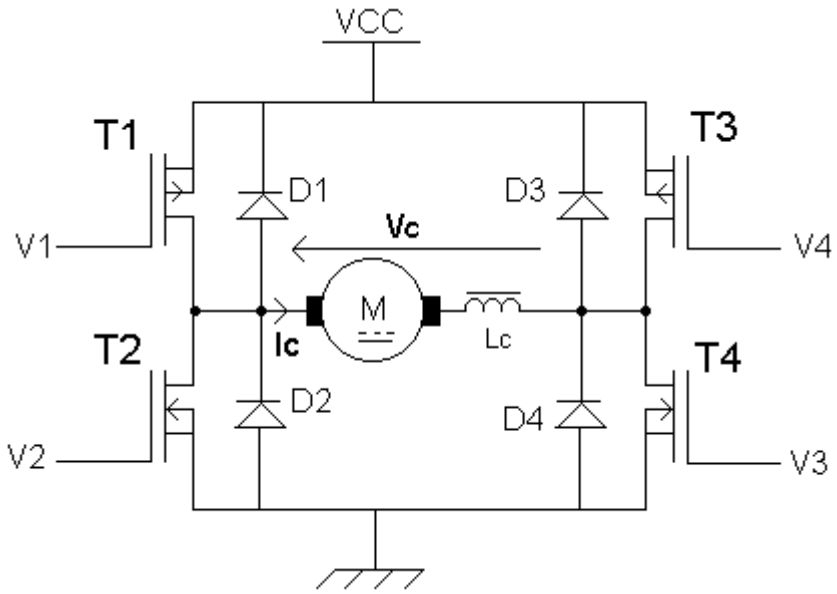
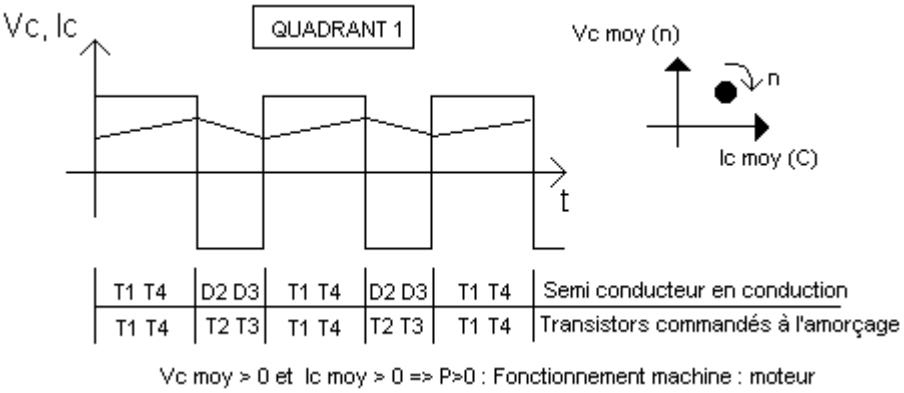
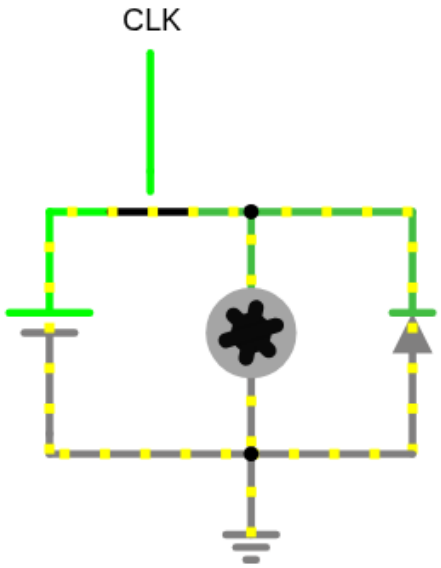
TP Arduino – Commande d’un moteur CC par pont en H (hacheur)

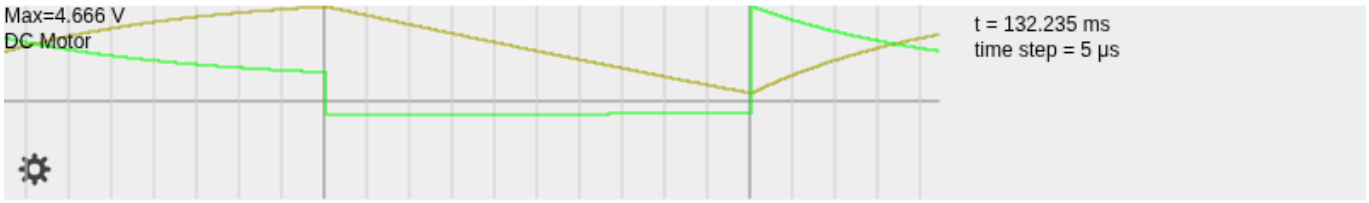




À retenir
Le hacheur permet de contrôler la vitesse du moteur. Il utlise la modulation de largeur d'impulsion (MLI ou PWM). Le rapport cyclique α(%) vaut $\alpha = \frac{t_{ON}}{T}$.

Ouvrir dans le navigateur

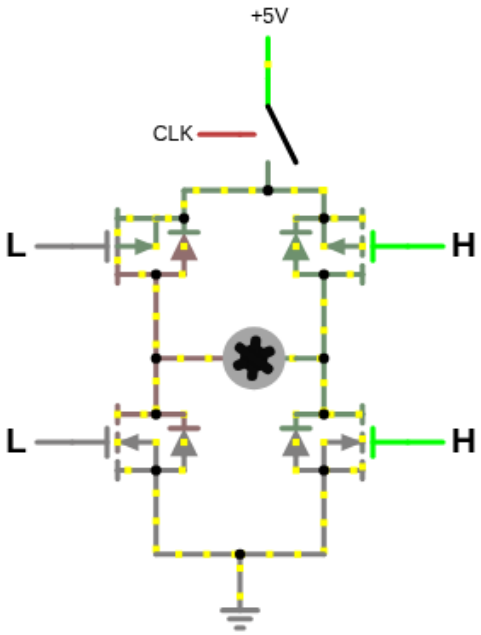


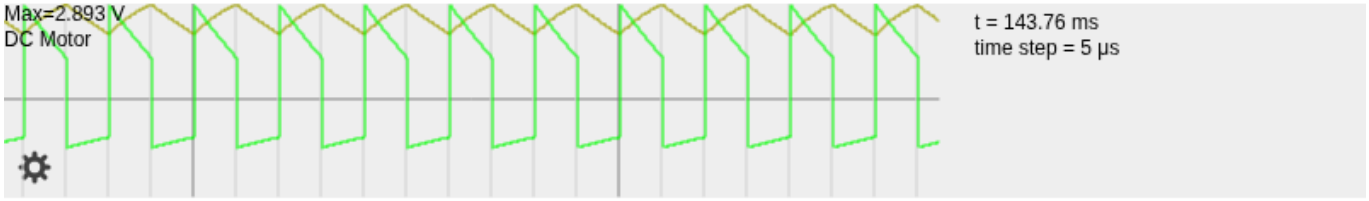


⚡ Ouvrir dans Falstad

Lecture seule (Circuit de référence : mistert2_hacheur1.txt) – connectez-vous pour créer ou enregistrer votre propre version.

https://mistert.freeboxos.fr/circuit/proxy.php?id=mistert2_hacheur1&mode=open





⚡ Ouvrir dans Falstad

Lecture seule (Circuit de référence : mistert2_hacheur2.txt) – connectez-vous pour créer ou enregistrer votre propre version.

https://mistert.freeboxos.fr/circuit/proxy.php?id=mistert2_hacheur2&mode=open

A) Câblage

Sur le matériel fourni :

- Connecter le moteur CC à la sortie du **pont en H**.
- Relier les broches de commande du pont en H aux sorties numériques de l’Arduino (ici D3 et D6).
- Vérifier l’alimentation.
- Objectif : faire tourner le moteur dans les deux sens.

B) Premier programme Arduino

Entrer le code suivant et observer le comportement du moteur :

Programme de commande de moteur sur Arduino

```
// Commande d'un moteur CC avec PWM et pont en H
// Tout ou rien (100 %), mais map() est déjà en place pour préparer la suite

const int sortie_commande_AV_pont_H = 3;
const int sortie_commande_AR_pont_H = 6;

void setup() {
  pinMode(sortie_commande_AV_pont_H, OUTPUT);
  pinMode(sortie_commande_AR_pont_H, OUTPUT);
}

// Marche avant (par défaut 100 %)
void moteurAvant(int alpha_percent = 100) {
  int alpha_pwm = map(alpha_percent, 0, 100, 0, 255);
  digitalWrite(sortie_commande_AR_pont_H, LOW);
  analogWrite(sortie_commande_AV_pont_H, alpha_pwm);
}

// Marche arrière (par défaut 100 %)
void moteurArriere(int alpha_percent = 100) {
  int alpha_pwm = map(alpha_percent, 0, 100, 0, 255);
  digitalWrite(sortie_commande_AV_pont_H, LOW);
  analogWrite(sortie_commande_AR_pont_H, alpha_pwm);
}

void loop() {

  moteurAvant();     // avant 100 %
  delay(2000);


}
```

C) Marche avant à 50 %

- Modifier l'appel dans `loop()` en utilisant `moteurAvant(50);`
- Observer le moteur : vitesse plus faible.
- Avec un multimère mesurer la tension moyenne sur le moteur.
- Avec l'oscilloscope :
    1. Capturer la tension aux bornes du moteur.
    2. Relever la largeur d'impulsion et le rapport cyclique (~50 %).
    3. Calculer la tension moyenne et vérifier que U_moy = 0,5 × U_alim.

D) Marche arrière à 70 %

- Modifier l'appel dans `loop()` en utilisant `moteurArriere(70);`
- Observer le moteur : marche arrière avec une vitesse plus élevée que pour 50 %.
- Avec un multimère mesurer la tension moyenne sur le moteur.
- Avec l'oscilloscope :
    1. Capturer la tension aux bornes du moteur.
    2. Relever le rapport cyclique (~70 %).
    3. Calculer la tension moyenne et vérifier que U_moy = -0,7 × U_alim.

E) Observation du courant haché

- Modifier l'appel dans `loop()` en utilisant `moteurAvant(50);`
- Avec l'oscilloscope et la sonde de courant :
    1. Capturer la forme du courant moteur.
    2. Comparer avec le cas où le moteur est alimenté en continu.
    3. Expliquer pourquoi le courant est ici de forme quasi triangulaire : lien avec la PWM et l'inductance du moteur.
    4. Exercer un couple résistant sur le moteur (freinage à la main par exemple).

Que se passe-t-il sur l'amplitude moyenne et l'ondulation du courant ? Conclure sur l'intérêt du hacheur.

F) Programmation d'une rampe de vitesse

Lors d'un démarrage direct en PWM, le moteur reçoit immédiatement sa consigne (exemple : 100 %). Cela provoque :

- un appel de courant important,
- une accélération brutale,
- un risque de perte d'adhérence (patinage d'un robot, glissement d'un convoyeur),
- une usure plus rapide du système mécanique.

Pour éviter cela, on programme une **rampe de vitesse** : la consigne de PWM augmente progressivement.

Exemple de code :

```
void loop() {
  // Rampe de vitesse en marche avant
  for (int alpha = 0; alpha <= 100; alpha += 5) {
    moteurAvant(alpha);      // vitesse croissante
    delay(200);              // attendre 200 ms entre chaque pas
  }

  delay(2000); // maintenir pleine vitesse 2 s
```

```
  // Rampe de vitesse en marche arrière
  for (int alpha = 0; alpha <= 100; alpha += 5) {
    moteurArriere(alpha);
    delay(200);
  }

  delay(2000); // maintenir pleine vitesse 2 s
}
```

Travail demandé :

- Observer la différence de comportement avec et sans rampe.
- Mesurer le courant moteur avec l'oscilloscope.
- Expliquer en quoi la rampe améliore l'adhérence et limite les efforts mécaniques.

G) Marche avant arrière commandée par un potentiomètre

Règle:

- à position médiane du potentiomètre, le moteur est l'arrêt
- sur la partie gauche, le moteur tourne à gauche (vitesse max sur position max gauche)
- sur la partie droite, le moteur tourne à droite (vitesse max sur position max droite)

H) Commande par clavier

Objectif : Programmer la carte Arduino pour commander un moteur à courant continu via un **pont en H**, en saisissant les consignes directement depuis le **moniteur série**.

—

Principe de fonctionnement 1. L'utilisateur entre au clavier :

1. le **sens de rotation** du moteur :

`F` pour *Forward* (horaire) ou `R` pour *Reverse* (antihoraire),

1. le **rapport cyclique** (en %),
2. la **position à atteindre** (en nombre d'impulsions du codeur).

2. Le programme :

1. commande le moteur à l'aide du **pont en H**,
2. fait varier la vitesse par **PWM**,
3. compte les impulsions du **codeur incrémental**,
4. **arrête automatiquement** le moteur lorsque la position demandée est atteinte.

—

Travail demandé

1. **Analyser** le programme fourni.

2. **Tester** son fonctionnement à l’aide du **moniteur série**.

3. **Observer** le comportement du moteur selon :

    1. le sens choisi,
    2. le rapport cyclique,
    3. la position demandée.

4. **Améliorer** le programme en ajoutant une rampe de vitesse pour la marche arrière.

```
// === Commande moteur CC avec pont en H et codeur incrémental ===
```

```
// Version optimisée avec fonction lireTexteSerie() pour lecture clavier

// --- Sorties PWM vers le pont en H ---
const int sortie_commande_AV_pont_H = 3;
const int sortie_commande_AR_pont_H = 6;

// --- Entrées codeur incrémental ---
const int canalA = 2;   // interruption 0
const int canalB = 11;

// --- Variables globales ---
volatile long compt = 0;     // compteur codeur (volatile car modifié par interruption)
String sens = "";
String consigne = "";
String position = "";

byte alpha = 0;          // consigne en PWM
byte alphafinal = 0;     // rampe progressive
int positionfinale = 0;

// --- Prototypes ---
void moteurAvant(int alpha_percent = 100);
void moteurArriere(int alpha_percent = 100);
void Reagir();
String lireTexteSerie(String messageInvite);
void choix_sens();
void choix_consigne();
void choix_position();

// === SETUP ===
void setup() {
  Serial.begin(9600);
  pinMode(sortie_commande_AV_pont_H, OUTPUT);
  pinMode(sortie_commande_AR_pont_H, OUTPUT);
  pinMode(canalB, INPUT);

  attachInterrupt(digitalPinToInterrupt(canalA), Reagir, RISING);

  Serial.println("Système prêt !");
}

// === BOUCLE PRINCIPALE ===
void loop() {
  choix_sens();

  if (sens == "F" || sens == "R") {
    choix_consigne();
    choix_position();
```

```
    alphafinal = 50;  // démarrage progressif

    if (sens == "F") {
      Serial.println("→ Moteur sens horaire (avant)");
      digitalWrite(sortie_commande_AR_pont_H, LOW);
      compt = 0;

      while (compt < positionfinale) {
        moteurAvant(map(alphafinal, 0, 255, 0, 100));
        if (alphafinal < alpha) alphafinal++;
        delay(4);
      }
      moteurAvant(0);
      Serial.print("Position finale atteinte : ");
      Serial.println(compt);
    }

    else if (sens == "R") {
      Serial.println("← Moteur sens antihoraire (arrière)");
      digitalWrite(sortie_commande_AV_pont_H, LOW);
      compt = 0;

      while (-compt < positionfinale) {
        moteurArriere(map(alphafinal, 0, 255, 0, 100));
        if (alphafinal < alpha) alphafinal++;
        delay(4);
      }
      moteurArriere(0);
      Serial.print("Position finale atteinte : ");
      Serial.println(-compt);
    }

    Serial.println("Cycle terminé !");
  }
}

// === Fonction générique pour lecture série ===
String lireTexteSerie(String messageInvite) {
  Serial.println(messageInvite);
  String texte = "";
  char car = 0;

  while (car != '\n') {
    if (Serial.available() > 0) {
      car = Serial.read();
      if (car != '\n' && car != '\r') {
        texte += car;
      }
    }
  }
  texte.trim(); // Supprime espaces et retours éventuels
```

```
  return texte;
}

// === INTERRUPTIONS CODEUR ===
void Reagir() {
  if (digitalRead(canalB) == HIGH)
    compt++;
  else
    compt--;
}

// === Commandes moteur ===
void moteurAvant(int alpha_percent) {
  int alpha_pwm = map(alpha_percent, 0, 100, 0, 255);
  digitalWrite(sortie_commande_AR_pont_H, LOW);
  analogWrite(sortie_commande_AV_pont_H, alpha_pwm);
}

void moteurArriere(int alpha_percent) {
  int alpha_pwm = map(alpha_percent, 0, 100, 0, 255);
  digitalWrite(sortie_commande_AV_pont_H, LOW);
  analogWrite(sortie_commande_AR_pont_H, alpha_pwm);
}

// === Fonctions d'interaction ===
void choix_sens() {
  sens = lireTexteSerie("Quel fonctionnement souhaitez-vous ? (F = Forward,
R = Reverse)");
}

void choix_consigne() {
  consigne = lireTexteSerie("Quel rapport cyclique souhaitez-vous (20 à 100
%) ?");
  alpha = map(consigne.toInt(), 0, 100, 0, 255);
  Serial.print("→ PWM = ");
  Serial.println(alpha);
}

void choix_position() {
  position = lireTexteSerie("Quelle position souhaitez-vous atteindre (20 à
2000) ?");
  positionfinale = position.toInt();
  Serial.print("→ Position finale = ");
  Serial.println(positionfinale);
}
```