Classe de problèmes MECATRO-1

Piloter un système à l'aide d'un microcontrôleur.

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 de visualiser les **composants** des chaînes fonctionnelles en mettant en évidence la **chaîne d'énergie** et la **chaîne d'information** puis déterminer comment un moteur à courant continu peut être piloter par une carte de commande.

Le support proposé est une carte Arduino reliée à un moteur à courant continu via un hacheur. Pour piloter la carte, deux approches sont proposées :

- **programmation numérique** d'une part, avec l'implémentation de code via le logiciel Arduino (langage Arduino)
- **simulation numérique** d'autre part avec l'utilisation de schémas-blocs via le module Xcos de Scilab

1 Présentation

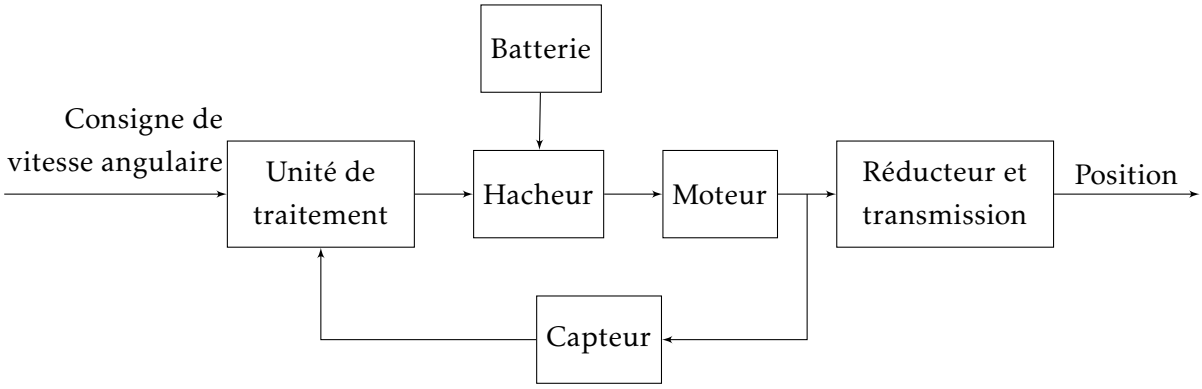
1.1 Contexte

De nombreux systèmes autonomes en énergie d'alimentation électrique (batteries) sont équipés de moteurs à courant continu. Ces moteurs permettent de convertir l'énergie électrique en énergie mécanique qui sera adaptée de manière à agir.

Par exemple, pour un robot tondeuse à gazon électrique, l'énergie mécanique de rotation fournie par un moteur est adaptée par un réducteur et une transmission de manière à entraîner en rotation une des roues motrices qui va agir sur le sol et provoquer le déplacement du chariot.

La variation de vitesse d'une roue motrice est obtenue en modulant l'énergie électrique d'alimentation du moteur à l'aide d'un hacheur. Le hacheur est piloté par l'unité de traitement. Cette dernière envoie un signal au hacheur de telle manière que la vitesse de rotation du moteur sera proportionnelle à ce signal.

Les chaînes d'énergie peuvent être équipées d'un ou plusieurs capteurs qui permettent d'évaluer des grandeurs physiques comme par exemple la vitesse de rotation du moteur. Dans certains cas, les grandeurs physiques évaluées pourront être utilisée pour optimiser la commande du moteur. On parle de système asservi.



1.2 Présentation de la carte *Arduino-Uno*

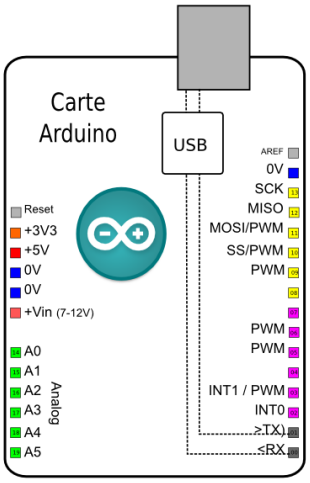
La carte **Arduino-Uno** est un exemple d'unité de traitement. L'élément central de la carte est le microcontrôleur. Son rôle est de traiter les données et d'assigner les sorties.
Les autres éléments principaux sont le connecteur USB, les 14 broches **numériques**, les 6 broches **entrées analogiques**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

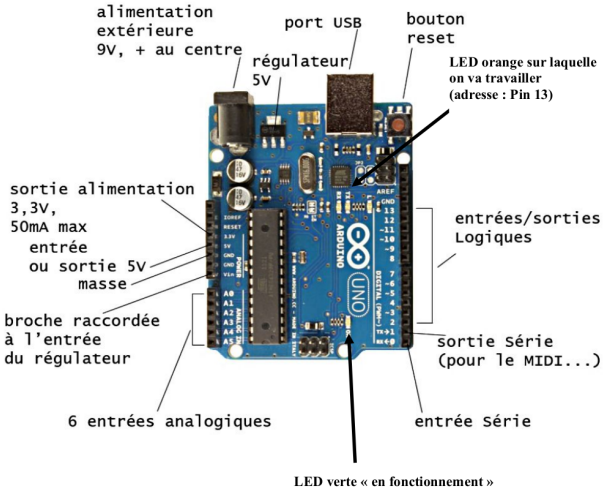
Le connecteur USB assure la communication de la carte avec le micro-ordinateur, et permet son alimentation en l'absence d'une autre source d'alimentation.

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 Exemples de câblages

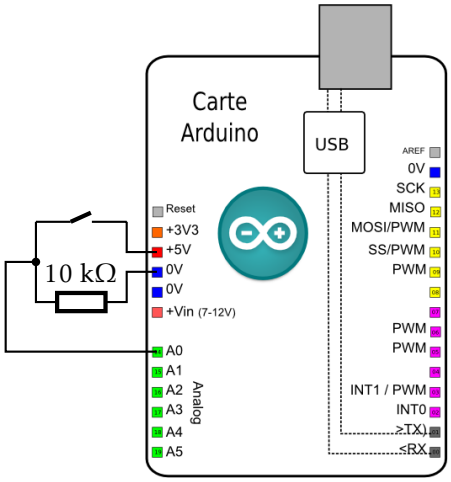


FIGURE 1 – Câblage d'une entrée TOR avec bouton poussoir

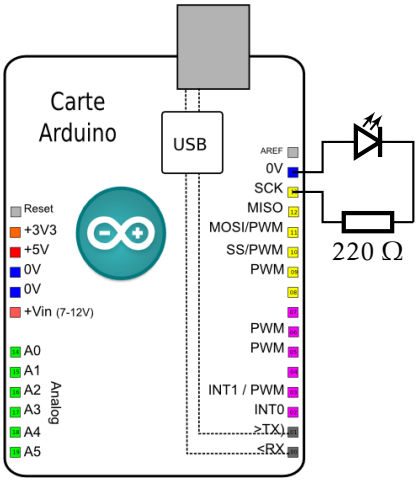


FIGURE 2 – Câblage d'une LED

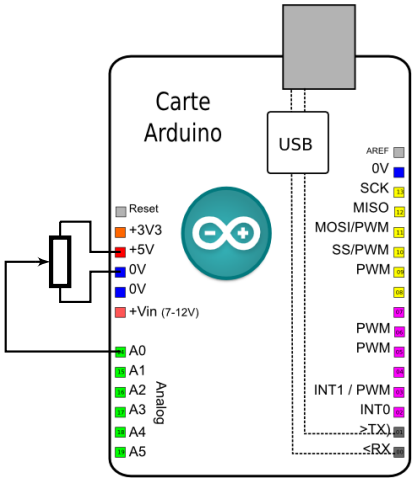


FIGURE 3 – Câblage d'une entrée analogique

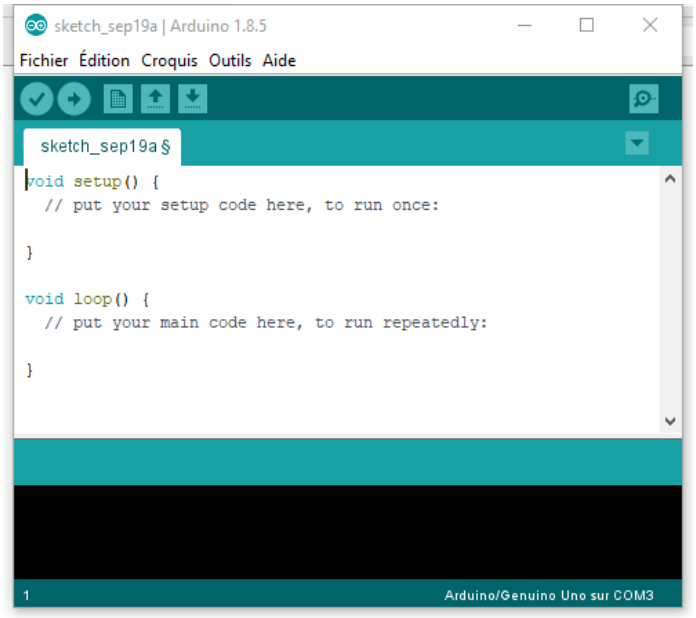
3 Vérification du fonctionnement de la carte

La première étape dans le pilotage de microcontrôleurs et automates est de vérifier la communication entre le PC sur lequel on programme et l'unité de traitement de l'automate. Le plus facile est souvent de faire clignoter un diode (ou LED) . La carte Arduino est constituée d'une LED interne reliée à la broche numérique 13. On se propose donc de la faire clignoter avec deux approches différentes.

3.1 Approche code Arduino

- Lancer le logiciel Arduino en cliquant sur 

La fenêtre ci-contre apparaît. Elle présente la structure du programme avec deux fonctions (`void`).



La fonction `setup()` est appelée une seule fois lorsque le programme commence. Dans cette fonction, on écrira donc le code qui n'a besoin d'être exécuté qu'une seule fois.

La fonction `loop()` est la fonction principale du programme, elle est appelée en permanence. Le code est exécuté dans une boucle infinie.

En début de programme avant la fonction `setup`, on peut déclarer des variables.

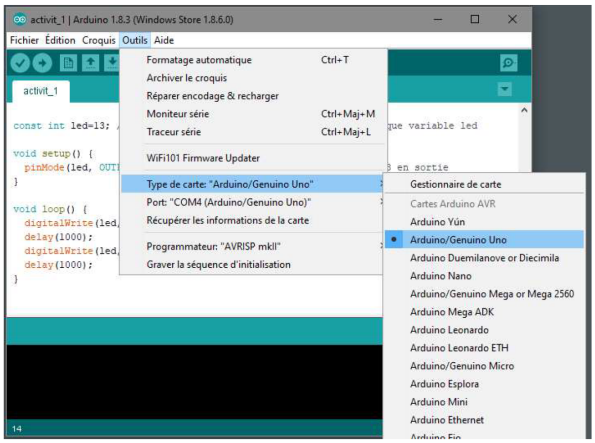
- Écrire le code ci-dessous (téléchargeable ici) :

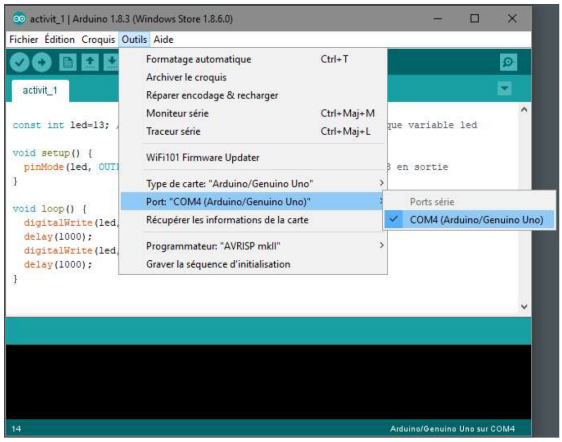
```
void setup() {
  pinMode(13, OUTPUT); // configuration de la broche numérique 13 en sortie
}

void loop() {
  digitalWrite(13, HIGH) ;  // Mise à 1 de la sortie 13
  delay(1000); // délai de  1000 ms soit 1s
  digitalWrite(13, LOW) ;  // Mise à 0 de la sortie 13
  delay(1000); // délai de  1000 ms soit 1s
}
```

- Brancher le câble USB entre la carte et le PC ; Avant de compiler et de téléverser ce code sur la carte Arduino, il faut s'assurer de la bonne configuration du logiciel par rapport à la carte, et du port USB.

  REMARQUE : pour trouver le port de connexion, aller dans le gestionnaire de périphériques qui se trouve dans le panneau de configuration.
- Compiler et téléverser le programme en cliquant sur la flèche dans la barre d'accès rapide ; Remarque : lors du téléversement les LEDs internes Tx et Rx doivent clignoter.

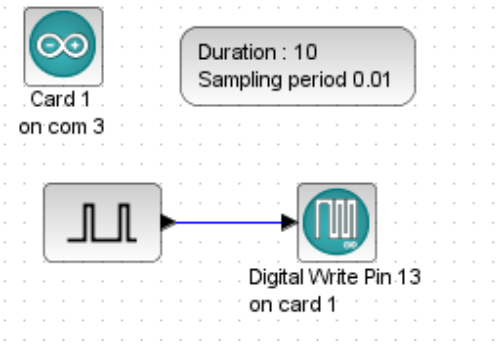




- Vérifier le bon fonctionnement de la carte.

3.2 Approche schéma-bloc Xcos de Scilab

- lancer **Scilab** puis taper xcos() dans l'interpréteur.
- A partir des éléments des bibliothèques, construire le schéma ci-contre :

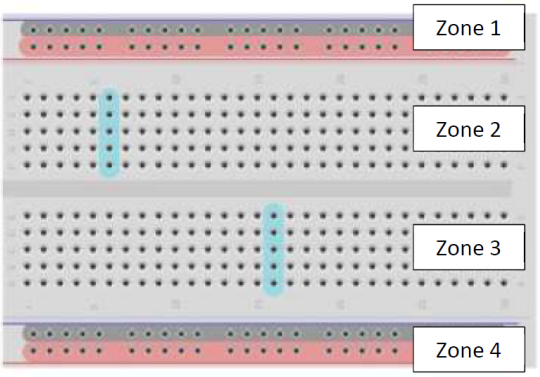


- Régler **Amplitude** à 5, **Period** à 1, **Pulse Width** à 35 et **Phase delay** à 0
- Bien **choisir le port** de communication (généralement COM 3).
- Cliquer sur le triangle **Démarrer**.

Normalement, la diode située au niveau de la patte 13, clignote.

3.3 LED sur une autre broche

On peut aussi faire clignoter une LED autre que celle de la carte. Pour cela on a besoin d'une plaque d'essai (**breadboard**), c'est une platine pleine de trous, dans lesquels on peut piquer des pattes de composants électroniques ou des fils.



Dans les zones 1 et 4, les trous situés sur une même ligne horizontale (plus grande longueur) sont reliés électriquement.

Dans les zones 2 et 3, les trous situés sur une même ligne verticale sont reliés électriquement.

Remarque : les zones 1 et 2 servent à l'alimentation des composants électroniques.

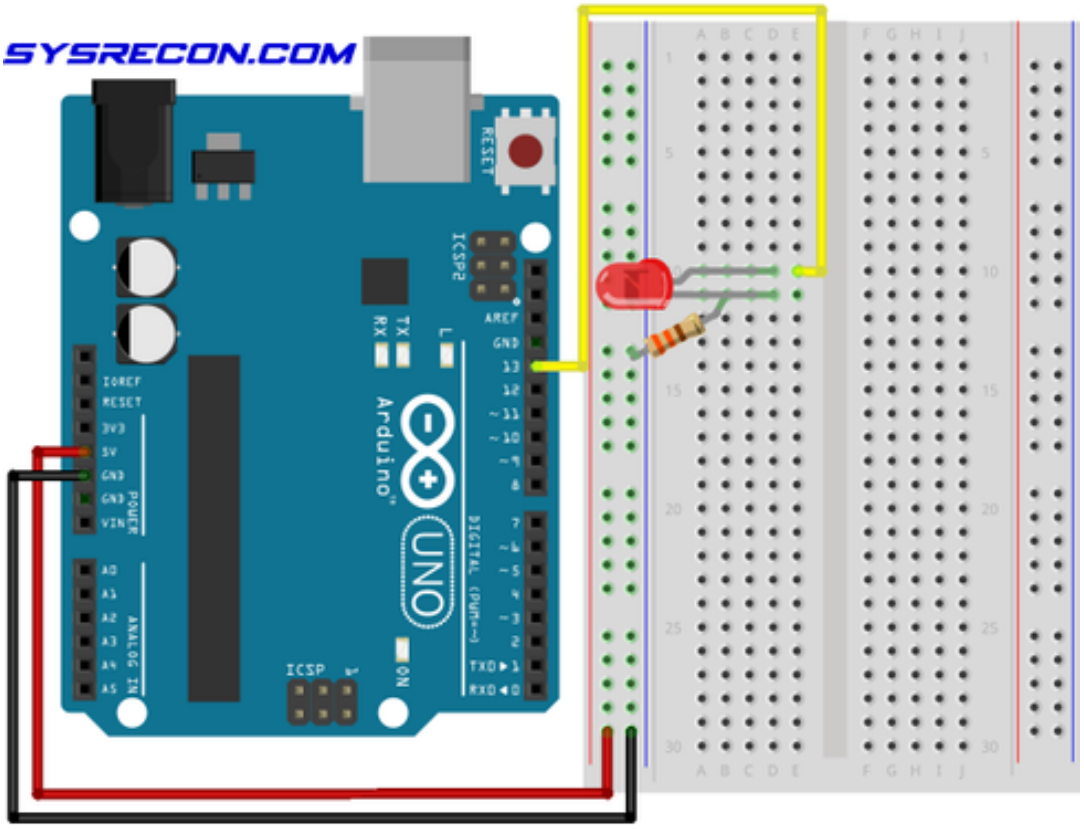
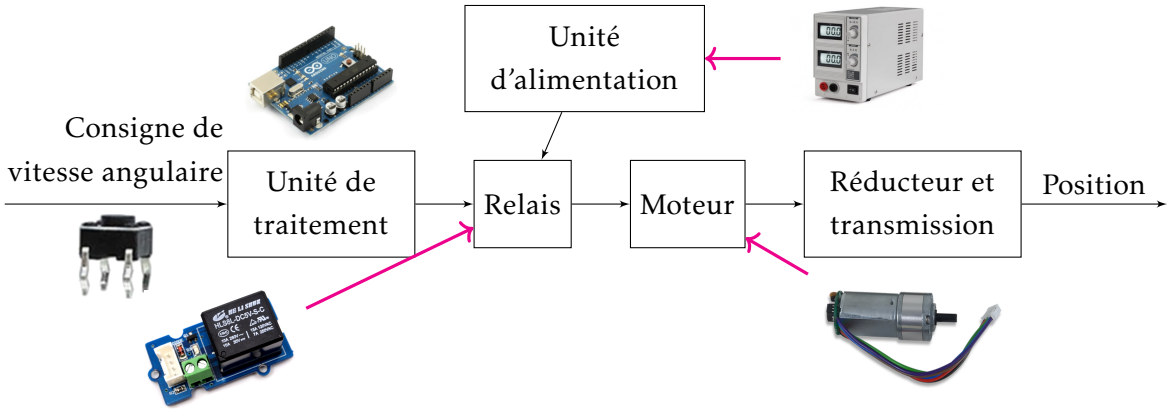


Figure 4 – Montage avec le cablage d'une led

4 Commande simple d'un moteur avec bouton poussoir

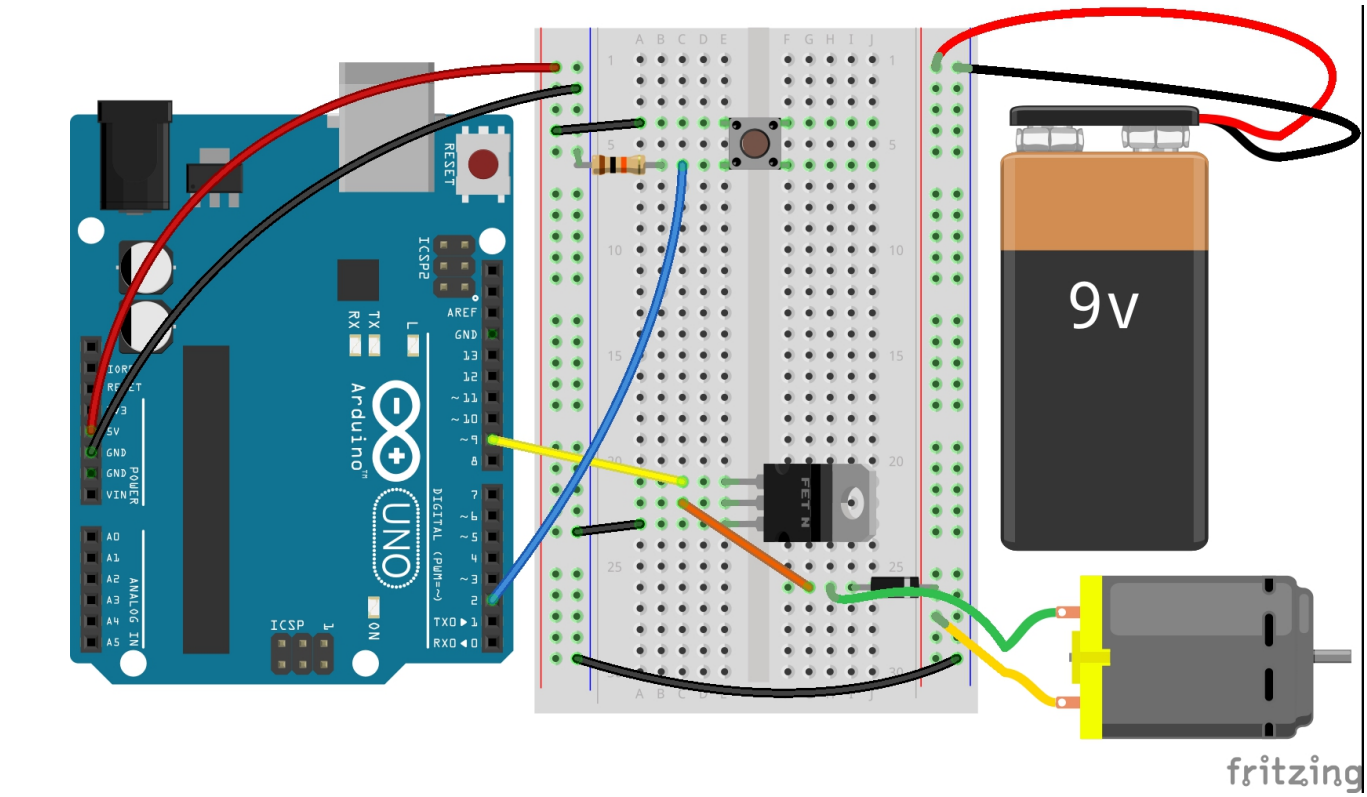
Objectif : réaliser la commande en rotation d'un moteur en appuyant sur un bouton poussoir avec une alimentation externe.



4.1 Détails des composants de base

Pour cela on utilise un moteur à courant continu en actionneur et un transistor MOSFET en préactionneur. Le transistor est un composant électronique qui fonctionne comme un contacteur.
Il a trois pattes : deux des pattes doivent être reliées au circuit d'alimentation du moteur et la troisième patte permet son pilotage. Lorsque cette troisième patte n'est pas alimentée, le courant ne passe pas, lorsqu'elle est alimentée le courant passe.

Pour éviter les retours de charge générés par le moteur en phase de décélération lorsque l'arrêt est demandé, il faut utiliser une diode en parallèle du moteur en faisant attention à sa polarisation. Pour que le transistor fonctionne correctement, il faut relier les GND de la carte Arduino et de l'alimentation externe.



Cependant pour simplifier le montage, le relais HLS8L sera utilisé à la place du MOSFET.

4.2 Acquisition de la consigne via un bouton poussoir

Avant de piloter le relais du moteur, on cherche à allumer ou éteindre la LED avec pour entrée Homme/Machine un bouton poussoir.

- Recopier le code ci-dessous (disponible ici) :

```
// 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() {
  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);
}
```

- Réaliser le câble FIG **??**.

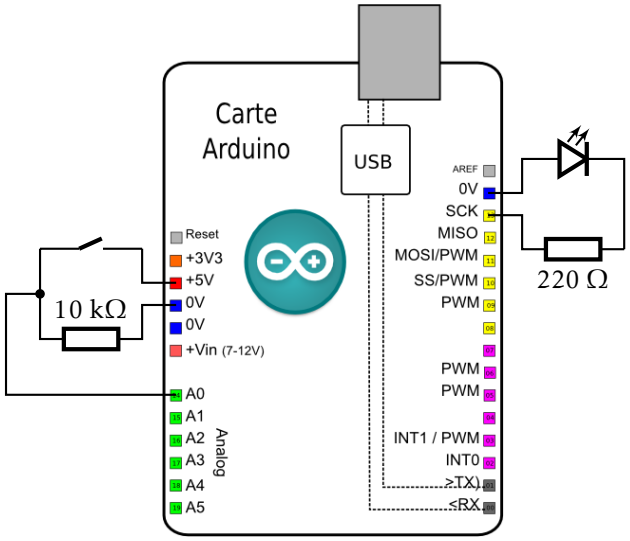


FIGURE 5 – Pilotage d'une LED avec un bouton poussoir pour IHM d'entrée

4.3 Pilotage Tout Ou Rien du moteur

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 **??** (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. 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) ;

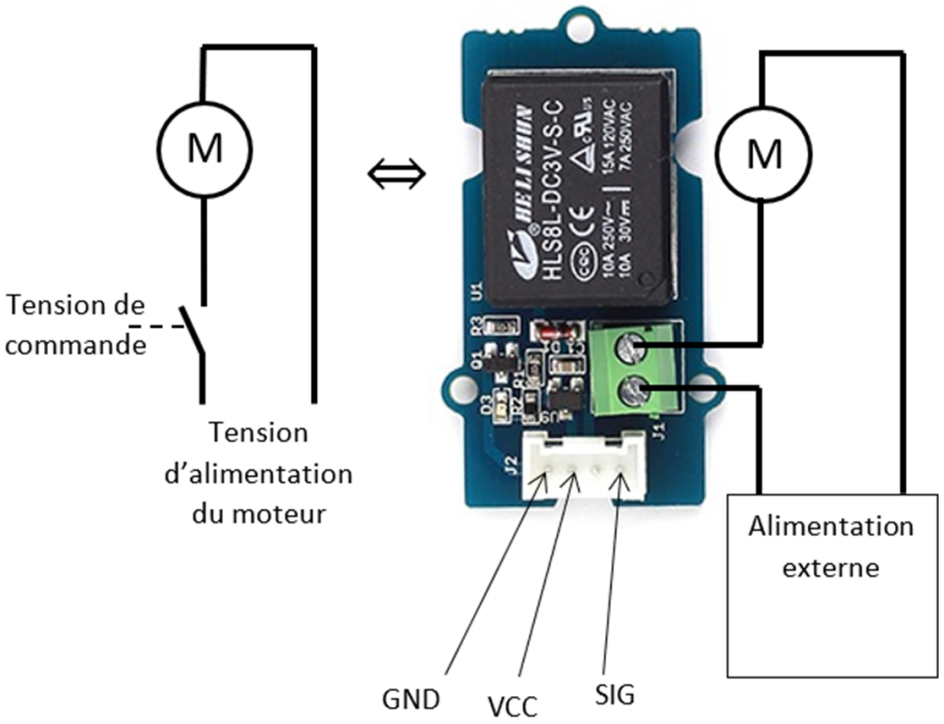


FIGURE 6 – Câblage du relais

[
Code des couleurs]Code des couleurs

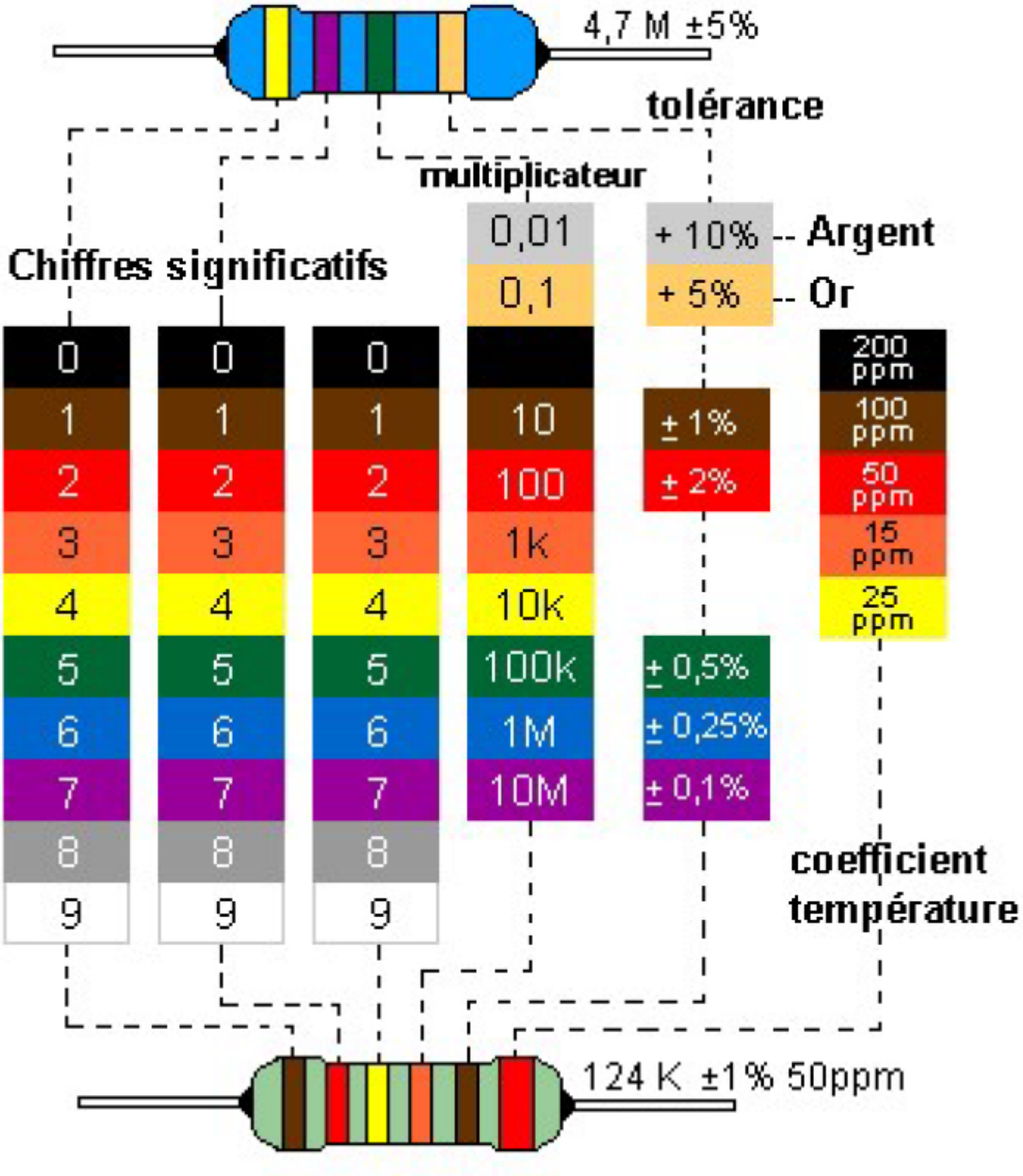


Figure 7 – Code couleurs des résistances