



MISE EN ŒUVRE

→ **TRAITER** : ARDUINO UNO (EDI Arduino)

→ **AQUERIR** : Potentiomètre

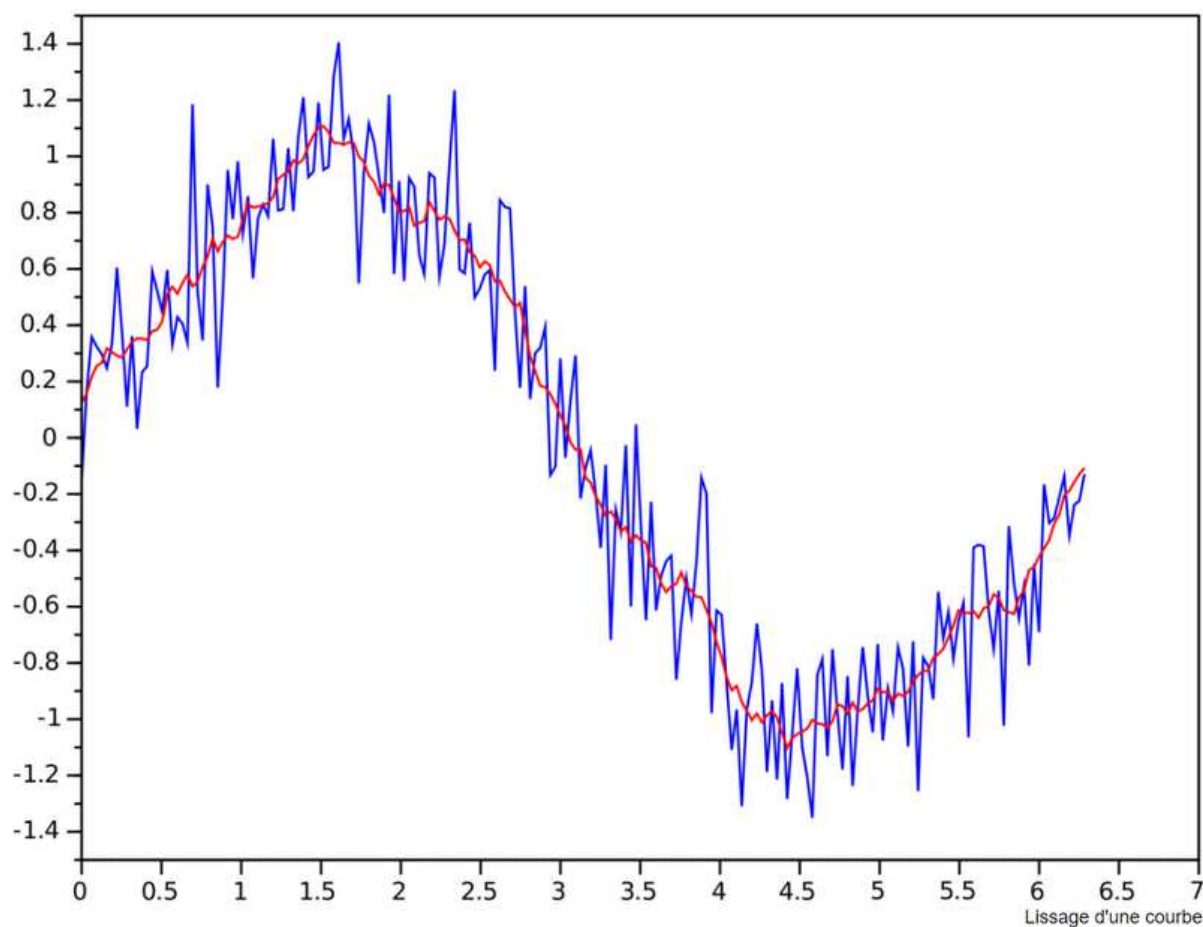
→ **COMMUNIQUER** : Moniteur série

1 – Mise en situation

Les courbes obtenues après importation des valeurs brutes issues d'un capteur peuvent être parfois difficiles à exploiter en raison des **bruits de mesures**.

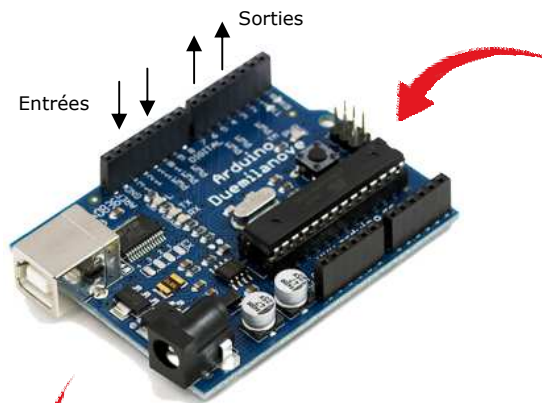
Il est donc intéressant de filtrer ces défauts dont la fréquence des variations est élevée (comparée aux variations du signal que l'on souhaite observer) pour ne conserver que l'évolution de la grandeur qui nous intéresse. Comme on peut le voir sur le graphe ci-après, **on supprime ainsi les fluctuations transitoires** de façon à souligner les tendances à plus long terme.

On parle de **l'opération de filtrage** qui est une étape indispensable du processus de conversion d'un signal analogique en signal numérique.



En **bleu** : le signal non traité

En **rouge** : le signal traité à l'aide d'une moyenne mobile



```

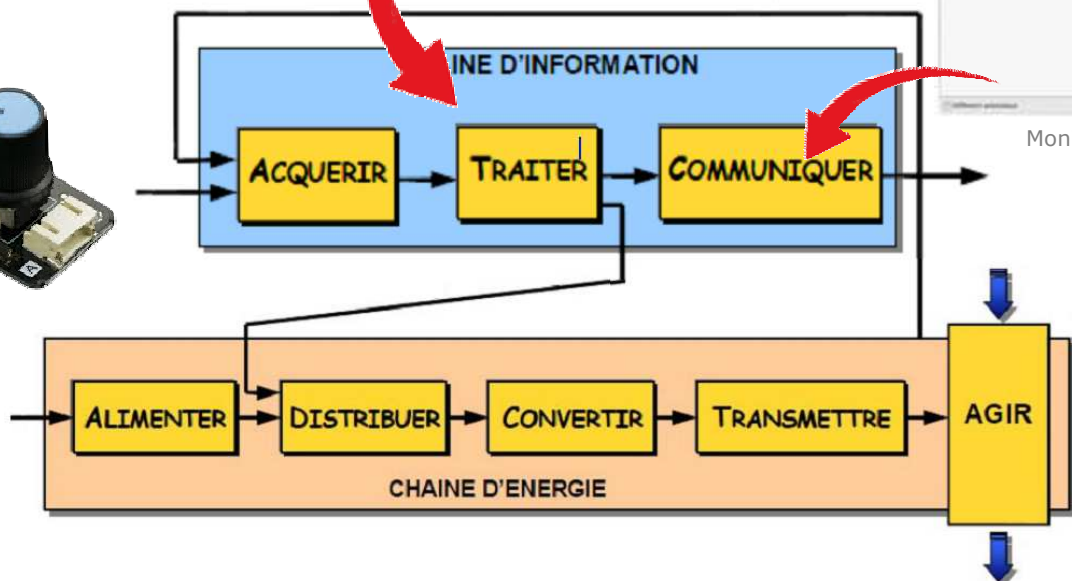
// sketch_007a
//
// Nomme des broches utilisées sur la carte Arduino
// Adresse PIN_vitesse 11 // PIN utilisée pour la variation de vitesse
// Adresse PIN_sens 3 // "broche 3" sur Arduino en "pin 1" sur L293D
// Adresse PIN_sens 4 // "broche 4" sur Arduino en "pin 2" sur L293D
//
// Le programme propose ici offre 3 fonctionnements possibles.
// choix = 1 -> Inversion du sens de rotation
// choix = 2 -> Variation de vitesse avec des échelons
// choix = 3 -> Variation de vitesse avec une rampe
//
// int choix = 2 // Régler ici celui qu'on veut utiliser...
//
void setup() {
  pinMode(PIN_vitesse, OUTPUT);
  pinMode(PIN_sens, OUTPUT);
  pinMode(PIN_sens, OUTPUT);
  Serial.begin(9600);
}

void loop() {
  if (choix == 1) { // Inversion du sens de rotation
    for (int i = 0; i < 5; i++) {
      digitalWrite(PIN_vitesse, HIGH);
    }
  }
}

```

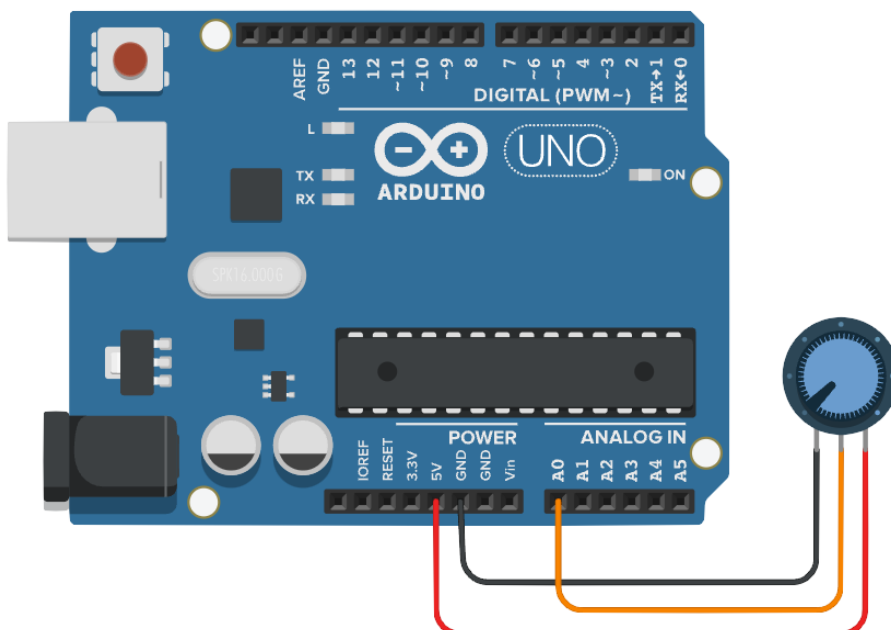
Programme Arduino

Carte programmable Arduino UNO



Moniteur série

2 – Plan de câblage / Montage



3 – Programme

Ce programme affiche dans le moniteur série les valeurs calculées de la moyenne mobile.

📖 Bibliothèques requises : RunningAverage.h

📖 Source : <https://github.com/RobTillaart/Arduino/tree/master/libraries/RunningAverage>

```
1 // =====
2 // Lissage : moyenne glissante
3 // =====
4 // source : https://github.com/RobTillaart/Arduino/tree/master/libraries/RunningAverage
5 /*
6 Ce programme ne fonctionne pas dans le simulateur TinkerCad.
7 c'est normal, ceci est du au fait que le bibliothèque "RunningAverage.h"
8 n'existe pas dedans.
9 */
10
11 // Appel des bibliothèques
12 #include "RunningAverage.h"
13
14 // Paramètres de la moyenne mobile (à régler comme on veut)
15 int inputPin = A0; // Broche d'un signal analogique ou mettre un nom de variable...
16 int taille = 10; // nombre de valeurs à prendre pour calculer la moyenne mobile
17 int pause = 100; // valeur en ms du delay() à la fin du calcul de la moyenne mobile
18
19
20 // Création d'un objet "Moyenne glissante"
21 RunningAverage MoyGliss(taille); // objet "Moyenne Glissante" avec nombre d'échantillons pour le calcul
22
23 // Paramètres nécessaire au fonctionnement
24 int MoGl_cpt = 0; // Compteur d'échantillons pris
25 float moyenneG = 0; // Moyenne glissante calculée sur les échantillons
26
27 void setup(void) {
28   Serial.begin(9600);
29   MoyGliss.clear(); // Initialisation de l'objet "Moyenne Glissante"
30 }
31
32 void loop(void) {
33   int echantillon = analogRead(inputPin); // Lecture du capteur
34   MoyGliss.addValue(echantillon); // Ajout de l'échantillon à l'objet "Moyenne Glissante"
35   moyenneG = MoyGliss.getAverage(); // Récupération de la moyenne glissante
36
37   // Affichage des résultats
38   Serial.print(echantillon);
39   Serial.print(" - ");
40   Serial.println(moyenneG);
41
42   MoGl_cpt++; // Incrémentation du compteur
43   if (MoGl_cpt == 300) { // Effacement de l'historique pour éviter les débordements
44     MoGl_cpt = 0;
45     MoyGliss.clear();
46   }
47   delay(pause);
48 }
```