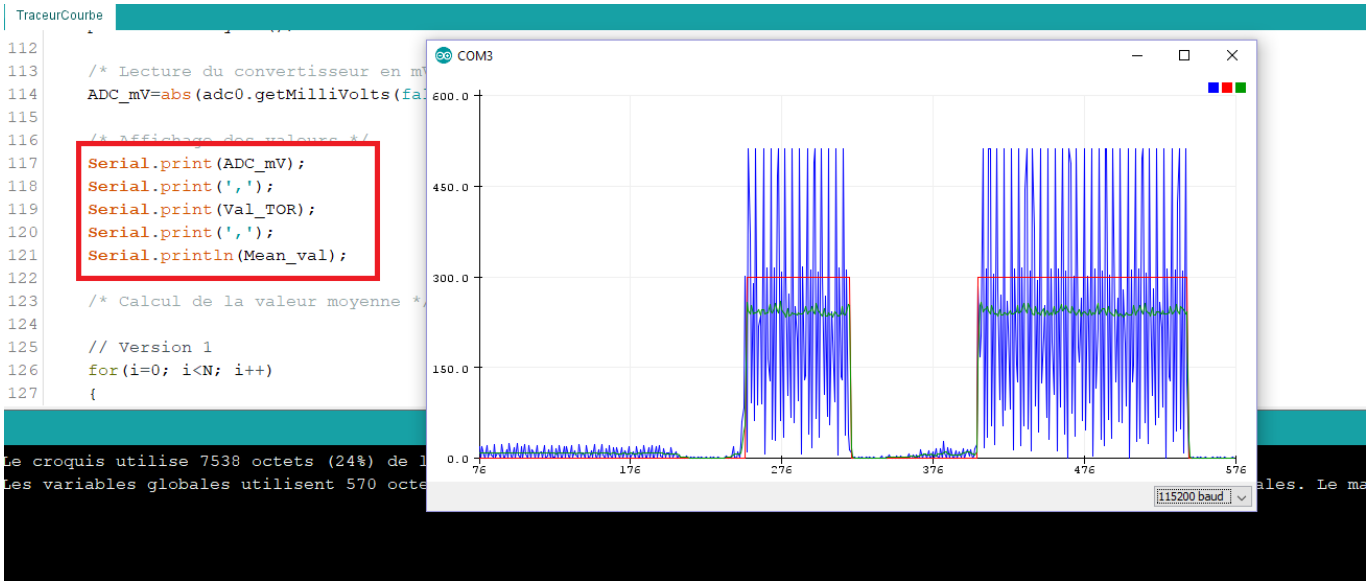


Objectifs

- Savoir comment utiliser le traceur des courbes avec Arduino (Oscilloscope)
- Savoir afficher une ou plusieurs courbes avec Arduino
- Savoir calculer la valeur moyenne d'un signal
- Savoir convertir un signal analogique en un signal TOR
- Savoir utiliser le convertisseur A/N ADS1115
- Etc.

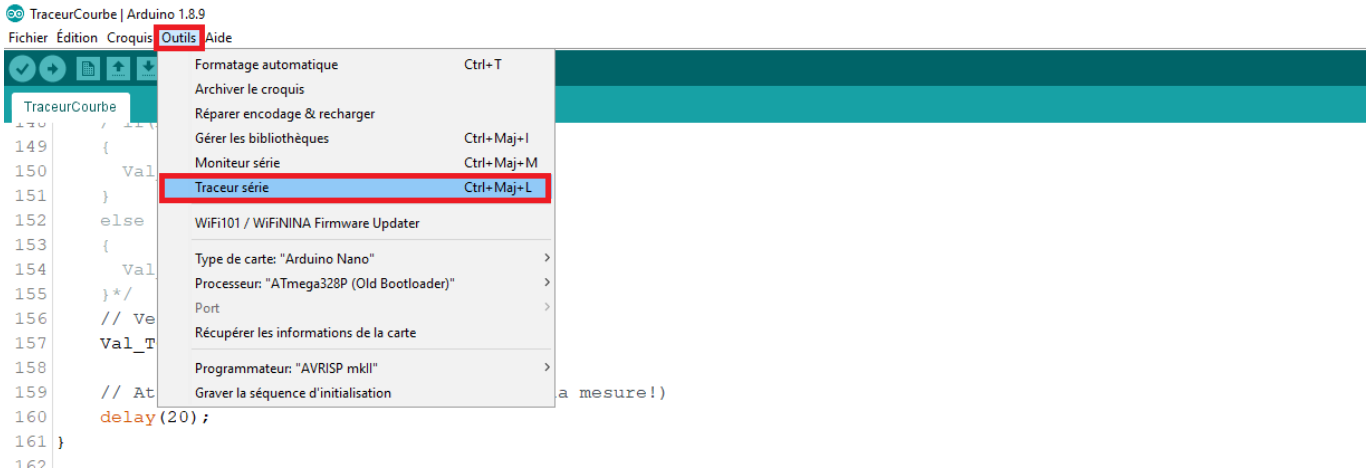
Vidéo démonstration

Fonctionnement

Nous avons utilisé le même schéma du projet capteur de toucher avec Arduino. Je vous recommande de référer à ce dernier pour plus d'informations. Dans le présent article on abordera comment utiliser l'outil « traceur série » et les techniques d'implémentation de la valeur moyenne, ainsi la conversion d'un signal analogique en un signal TOR.

À quoi sert le traceur série ?

Le traceur série est un nouvel outil intégrer dans l'interface de l'IDE Arduino, à partir de la version 1.8.9. Il est semblable au « moniteur série » dont l'objectif est d'afficher les données envoyées. Contrairement au « traceur série » son objectif est de « visualiser » les données séries, comme un oscilloscope. Ci-dessous la fenêtre d'accès au traceur :



Comment tracer les courbes avec Arduino ?

Le traceur série permet d'afficher une ou plusieurs courbes au même temps, l'éditeur affecte une couleur différente pour chaque courbe. On peut afficher une valeur entière ou flottante. Le format d'affichage est normalisé en fonction du nombre de courbes à visualiser.

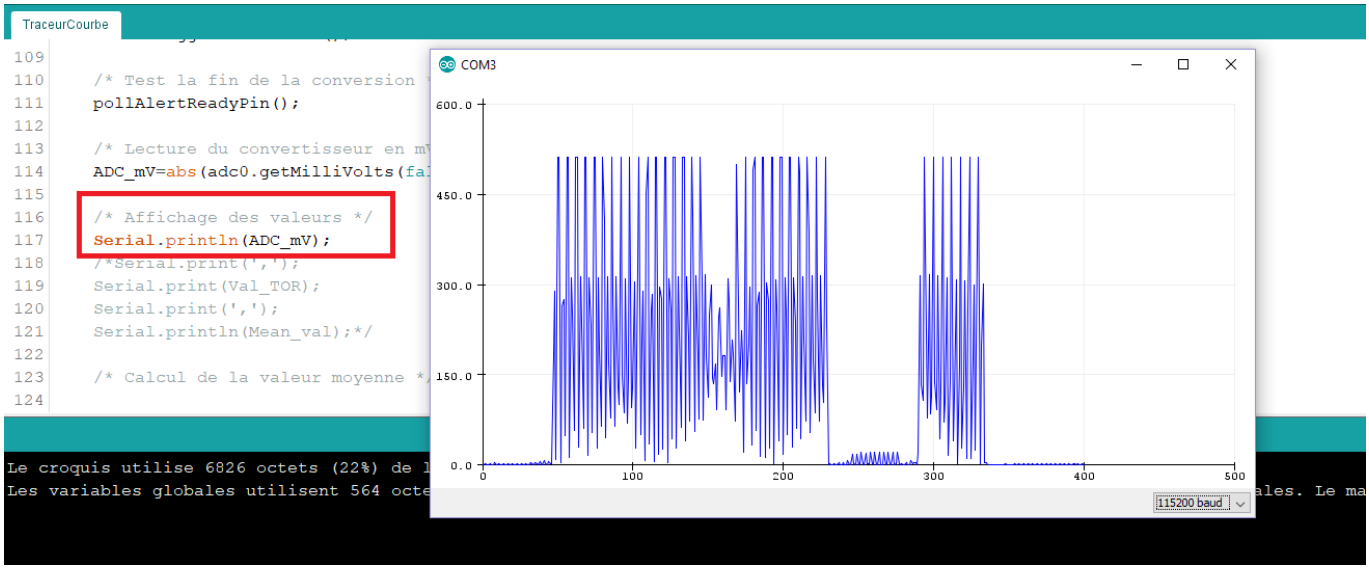
Format une courbe

On peut afficher une courbe par l'envoi des valeurs séparées par un retour à la ligne « \n » (V1 \n V2 \n,..., Vi). La fonction Serial.println(Vi) intègre le retour automatique à ligne ou bien Serial.print(Vi) suivie par Serial.print(\n).

> La couleur de la première courbe : Bleu

```
...
Serial.println(Val);
// Ou bien
Serial.println(Val);
Serial.println("\n");
```

...



Format deux courbes

Les valeurs de la première courbe (Vi) doivent être séparées par une virgule par rapport à la deuxième courbe (Vj). On peut afficher les deux courbes comme suit :

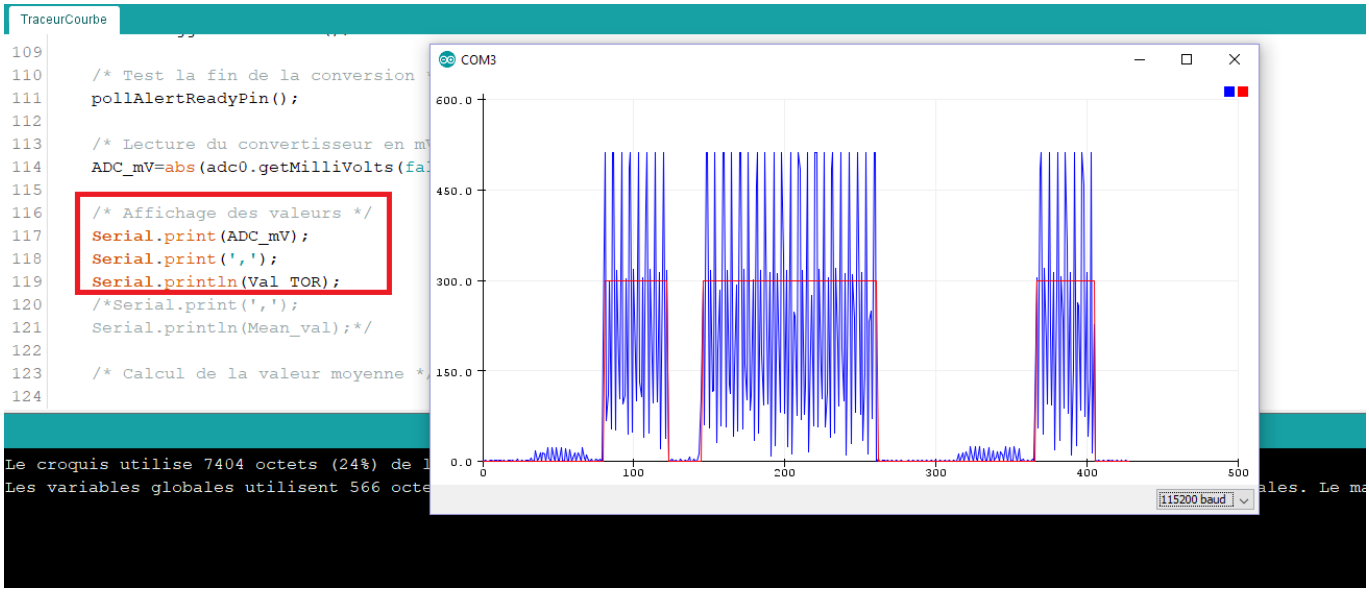
1. Envoi de la valeur Vi
2. Envoi de la virgule « , »
3. Envoi de la valeur Vj
4. Envoi du retour à la ligne « \n »

> La couleur de la deuxième courbe : Rouge

```
...
Serial.print(Vi);
Serial.print(",");
Serial.print(Vj);
Serial.print("\n");

// Ou bien
Serial.print(Vi);
Serial.print(",");
Serial.println(Vj);
```

...



Format trois courbes (généralisation)

Les valeurs de la première courbe (Vi) doivent être séparées par des virgules. La dernière valeur est suivie par le retour à la ligne. On peut afficher les trois (ou plusieurs) courbes comme suit :

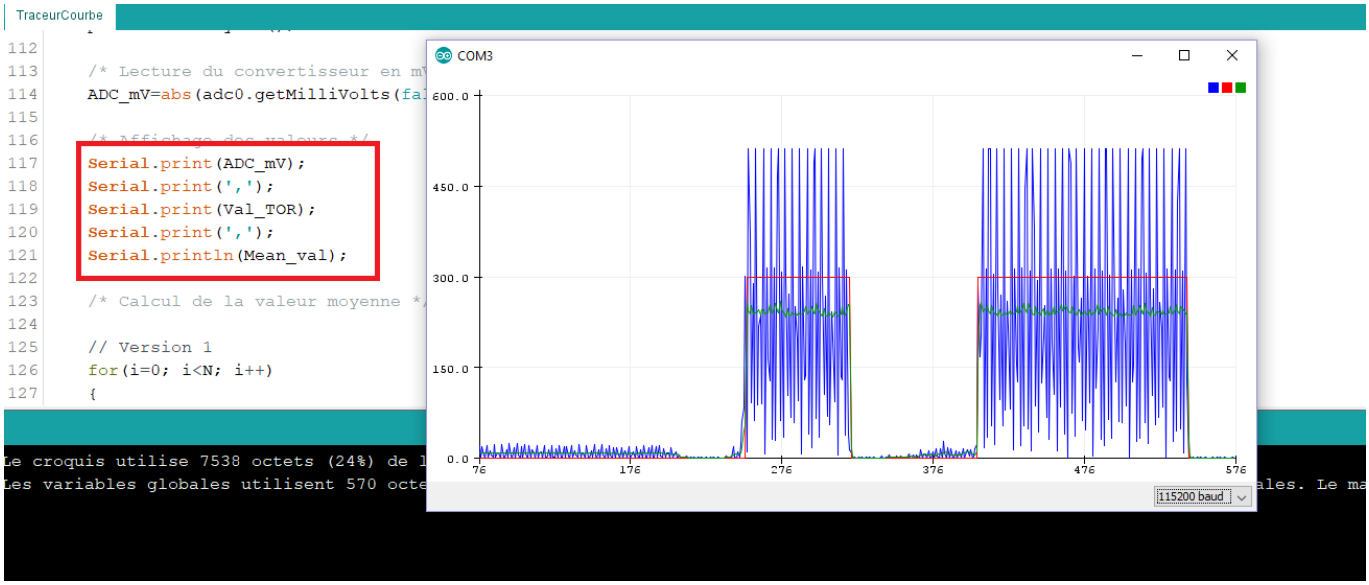
1. Envoi de la valeur Vi
2. Envoi de la virgule « , »
3. Envoi de la valeur Vj
4. Envoi de la virgule « , »
5. Envoi de la valeur Vk
6. ..
7. Envoi du retour à la ligne « \n »

La couleur de la troisième courbe : Vert

```
...
Serial.print(Vi);
Serial.print(",");
Serial.print(Vj);
Serial.print(",");
Serial.print(Vk);
Serial.print("\n");

// Ou bien
Serial.print(Vi);
Serial.print(",");
Serial.print(Vj);
Serial.print(",");
Serial.println(Vk);
```

...



Comment calculer la valeur moyenne d'un signal ?

La valeur moyenne d'un signal est égale à son intégration dans un intervalle du temps donné. Lorsqu'un signal est strictement positif sa valeur moyenne est nulle. La valeur moyenne d'un signal symétrique à zéro (alternance positive =-alternance négative) est égale à zero. La valeur moyenne est peu sensible aux transitions dans un signal contrairement à la valeur instantanée. Par conséquent, elle peut être un bon indicateur en instrumentation. On peut donc utiliser la valeur moyenne à la place de la valeur instantanée. En électronique analogique, la valeur moyenne est égale à la composante continue du signal (DC)(Voir la vidéo).

Pour calculer la valeur moyenne il suffit de sommer N échantillons d'un signal y(i) puis diviser la somme par N. Ici, on va faire l'acquisition de N valeurs du convertisseur A/N puis on divise la somme par N=16. Ci-dessous deux implémentations d'une valeur moyenne avec Arduino:

Implémentation 1

L'implémentation est basée sur une boucle de N itérations. Pour chaque itération, on fait

l’acquisition d’un échantillon, une accumulation. Puis on calcule la valeur moyenne en divisant par N la somme à la fin de la boucle. La méthode est précise car on recalcule la valeur moyenne pour chaque itération dans la boucle principale (loop()), mais nécessite N acquisitions. La méthode est peu efficace lorsque N’est important (N=512, 1024,64000, etc.) en termes du temps d’intégration (Voir la vidéo)..

```
...
Somme=0.0
for(i=0; i<N; i++)
{
  // Lecture ADC
  adc0.triggerConversion();
  pollAlertReadyPin();
  ADC_mV=abs(adc0.getMilliVolts(false));
  // Accumulation des échantillons
  Somme=Somme+ADC_mV;
}

// La valeur moyenne
Mean_val=Somme/N;

// Initialisation
```

```
Somme=0.0;
```

Implémentation 2

La méthode ne nécessite pas une boucle for. On fait une seule acquisition par itération dans la boucle principale. La mise à jour ou le calcul de la valeur moyenne est effectuée lorsqu'on atteint N itérations. Autrement dit, la valeur moyenne reste constante pendant N échantillons (ancienne valeur reste en mémoire), elle sera mis à jour chaque N itérations. La méthode est rapide simple à mettre en oeuvre. Le seul inconvénient est la durée de mise à jour (Chaque N itérations), contrairement à la première méthode dont la mise à jour est effectuée chaque itération (Voir la vidéo)..

```
...
// Accumulation
Somme=Somme+ADC_mV;

// Incrémentation compteur
Count+=1;

// La valeur moyenne
if (Count==N)
{
  Count=0;
  Mean_val=Somme/N;
  Somme=Mean_val;
}
```

...

```
...
// Version 1
if(Mean_val>150)
{
  Val_TOR=300;
}
else
{
  Val_TOR=0;
}
...
// Version 2
Val_TOR=floor(Mean_val/150)*300;
```

...

Comment convertir un signal analogique en un signal logique ?

Un signal numérique issu du convertisseur A/N est constitué de plusieurs niveaux ou valeurs. En revanche, un signal logique est constitué de deux états (niveaux haut/bas). On peut définir un seuil de transition entre les deux niveaux :

- Lorsque le signal est inférieur au seuil è Niveau bas
- Lorsque le signal est supérieur ou égal au seuil è Niveau bas

L'instruction IF peut faire la conversion vers un signal logique :

```
...
// Version 1
if(Mean_val>150)
{
  Val_TOR=300;
}
else
{
  Val_TOR=0;
}
...
// Version 2
Val_TOR=floor(Mean_val/150)*300;
```

...

Note : la deuxième version est rapide. Dans le cas général, in faut éviter d'utiliser l'instruction IF ou FOR car elles alourdit l'exécution du programme.

Programme principal

```
#include <I2Cdev.h>
/*
-----------------------------------------------
Librairies Arduino
-----------------------------------------------
I2C: https://github.com/jrowberg/i2cdevlib
ADS1115: https://github.com/addicore/ADS1115
-----------------------------------------------
Branchement Arduino-ADS1115:
-----------------------------------------------
ADS1115 -->  UNO
  VDD        5V
  GND        GND
  SCL        A5 (or SCL)
  SDA        A4 (or SDA)
  ALRT       2
-----------------------------------------------
*/

#include "ADS1115.h"

ADS1115 adc0(ADS1115_DEFAULT_ADDRESS);

// Paramètres
const int alertReadyPin = 2;
const int AlarmePin=3;
double ADC_mV=0.0;
double Seuil_Alarme_mV=50.0;

// Moyenne
float Mean_val=0.0, Somme=0.0;
int N=16, Count=0, i;
```

```
int Val_TOR=0;

void setup() {
    //Connexion du bus I2C
    Wire.begin();

    // Init interface série
    Serial.begin(115200);

    // Test de la connectivité
    Serial.println("Testing device connections...");
    Serial.println(adc0.testConnection() ? "ADS1115 connexion réussie"
: "ADS1115 connexion non établie");

    // Initiation du convertisseur ADS1115
    adc0.initialize();

    // Mode: Lecture unique (Il existe le mode continu!)
    adc0.setMode(ADS1115_MODE_SINGLESHOT);
    // Affectation de la fréquence d'échantillonnage (16SPS)
    adc0.setRate(ADS1115_RATE_860);
    // Affectation du Gain
    adc0.setGain(ADS1115_PGA_0P512);

    // Connexion de l'indicateur de fin de conversion
    pinMode(alertReadyPin,INPUT);
    adc0.setConversionReadyPinMode();

    // Alarme
    pinMode(AlarmePin,OUTPUT);
}
```

```
void loop()
{
    /* Sélection de l'entrée (A0) */
    adc0.setMultiplexer(ADS1115_MUX_P0_NG);

    /* Lancement de la conversion */
    adc0.triggerConversion();

    /* Test la fin de la conversion */
    pollAlertReadyPin();

    /* Lecture du convertisseur en mV */
    ADC_mV=abs(adc0.getMilliVolts(false));

    /* Affichage des valeurs */
    Serial.print(ADC_mV);
    Serial.print(',');
    Serial.print(Val_TOR);
    Serial.print(',');
    Serial.println(Mean_val);

    /* Calcul de la valeur moyenne */
    // Version 1
    for(i=0; i<N; i++)
    {
      adc0.triggerConversion();
      pollAlertReadyPin();
      ADC_mV=abs(adc0.getMilliVolts(false));
      Somme=Somme+ADC_mV;
    }
    Mean_val=Somme/N;
    Somme=0.0;
```

```
    // Version 2
    /*Somme=Somme+ADC_mV;
    Count+=1;
    if (Count==N)
    {
      Count=0;
      Mean_val=Somme/N;
      Somme=Mean_val;
    }*/

    /* Conversion en valeur TOR */
    // Version 1
    /*if(Mean_val>150)
    {
      Val_TOR=300;
    }
    else
    {
      Val_TOR=0;
    }*/
    // Version 2
    Val_TOR=floor(Mean_val/150)*300;

    // Attente (Attention à la fréquence de la mesure!)
    delay(20);
}

// Fonction affichage de l'état de conversion après 100000 tentatives
void pollAlertReadyPin()
{
  for (uint32_t i = 0; i<100000; i++)
    if (!digitalRead(alertReadyPin)) return;
```

```
   Serial.println("Failed to wait for AlertReadyPin, it's stuck
high!");
```

```
}
```

Téléchargement

- Traceur série avec Arduino

Tout les projets microcontrôleur