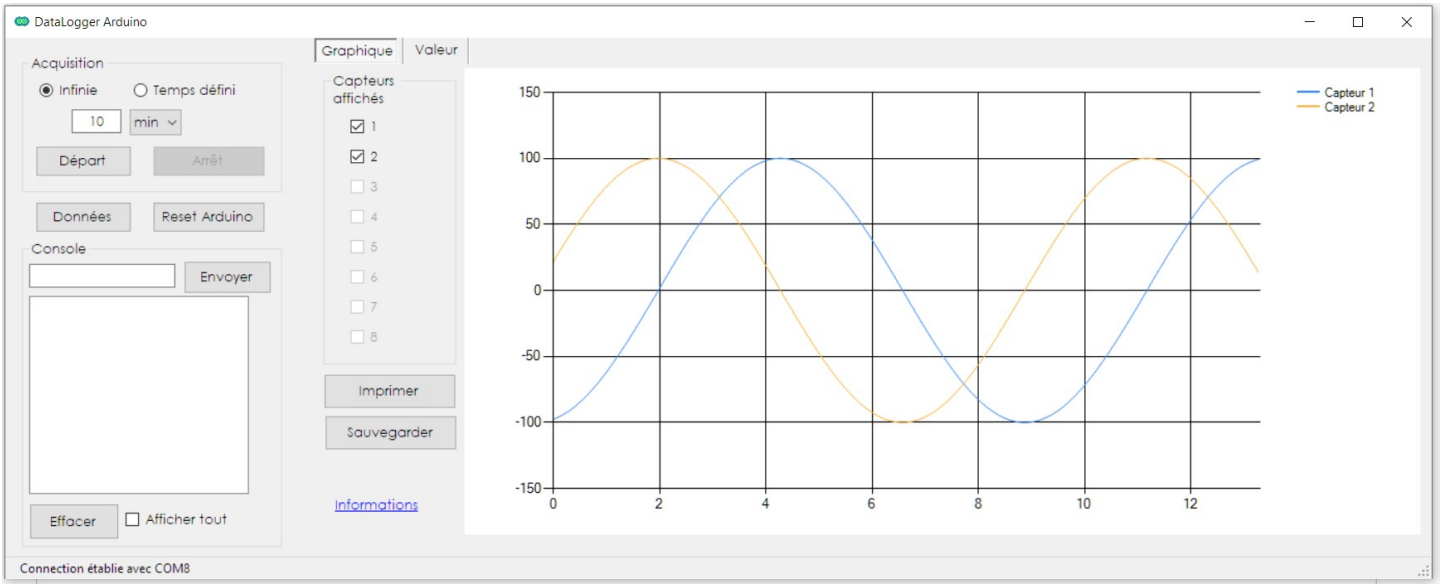
DataLogger pour Arduino/Micro:Bit

et toute autre carte programmable utilisant une communication série

Notice d'utilisation



Jean-Luc DUROU

jean-luc.durou@ac-bordeaux.fr

Sommaire

I – Introduction et crédits

DataLogger pour Arduino est un logiciel autonome pour Windows permettant de récupérer des données numériques (issues de capteurs par exemple) envoyées via le port série par une carte Arduino ou tout autre dispositif respectant le protocole (configurable).

Ce logiciel a été développé initialement par B@tto en Visual Basic, et je l'ai repris pour améliorer ses fonctionnalités, et particulièrement :

- langue en Français ;
- graphique imprimable ;
- graphique zoomable ;
- graphique redimensionné en plein écran ;
- récupération de données sur des réels ;
- données exportables en .csv...

La communication série entre la carte programmable et le PC doit être configurée avec :

- 1 bit start
- 8 bits de données,
- sans parité,
- avec un bit stop.

C'est la configuration par défaut sur les cartes Arduino et Micro:bit

II – Côté Arduino

Le programme Arduino doit suivre le même protocole que celui utilisé par le logiciel, à savoir un caractère spécifique (préfixe configurable) au début de chaque ligne de données, un caractère de séparation des données (séparateur configurable) sur une ligne. Ce protocole permet de dissocier les données des capteurs, qui seront intégrées au graphique, des données de dialogue, qui s'afficheront sur le logiciel.

L'exemple ci-dessous nous montre comment envoyer deux données réelles, la première avec une décimale et la seconde avec trois décimales via le port série, avec les caractères « X » comme préfixe et « , » comme séparateur de données :

```
//Préfixe protocolaire pour reconnaître une ligne de données
const char Prefixe = 'X' ;

//Séparateur des données dans une ligne
const char Separateur = ',' ;

int x = 0;

void setup()
{
  //Ouverture du port de série à la vitesse maximale
  Serial.begin(115200);
}

void loop()
{
  //Dans l'exmple on envoie les valeurs de 100.sin x et 100.cos x
  float s = sin(x*PI/180)*100;  //Calcul de 100.sin x
  float c = cos(x*PI/180)*100;  //Calcul de 100.cos x
  //Début de la transmission
  Serial.print(Prefixe);        //Préfixe d'une ligne
  Serial.print(Separateur);     //Séparateur entre chaque donnée
  Serial.print(s,1);            //100.sin x avec une décimale
  Serial.print(Separateur);     //Séparateur entre chaque donnée
  Serial.println(c,3);          //100.cos x avec trois décimales
  //println car c'est la dernière donnée de la ligne
  delay(50);
  //Ce délai (en ms) permet de définir la fréquence d'envoi des données
    (échantillonnage)
  x+=2;
}
```

Remarque : il est interdit de prendre le point comme séparateur des données, ou comme préfixe, car c'est le séparateur décimal par défaut d'Arduino ! Il ne faut bien entendu pas choisir de caractère numérique (0-9), ni le signe « - ».

III – Côté Micro:bit

Le programme Micro:bit doit suivre le même protocole que celui utilisé par le logiciel, à savoir un caractère spécifique (préfixe configurable) au début de chaque ligne de données, un caractère de séparation des données (séparateur configurable) sur une ligne. Ce protocole permet de dissocier les données des capteurs, qui seront intégrées au graphique, des données de dialogue, qui s'afficheront sur le logiciel.

L'exemple ci-dessous nous montre comment envoyer les données de l'accéléromètre, la première avec une décimale et la seconde avec trois décimales via le port série, avec les caractères « X » comme préfixe et « , » comme séparateur de données :

```
# Import des fonctions Micro:bit
from microbit import *
# Préfixe protocolaire pour reconnaître une ligne de données
prefixe = "X"
# Séparateur des données dans une ligne
separateur = ";"
# Initialisation de la variable "trame"
# Ce sera la ligne envoyée sur le port série
trame = ""

while True:
    # Lecture des accélérations et conversion en "g"
    AX = accelerometer.get_x()/1000
    AY = accelerometer.get_y()/1000
    AZ = accelerometer.get_z()/1000
    # Constitution de la trame
    trame = (
        prefixe             # Préfixe d'une ligne
        + separateur        # Séparateur entre chaque donnée
        + str(round(AX, 1))  # Accélération X avec une décimale
        + separateur        # Séparateur entre chaque donnée
        + str(round(AY, 2))  # Accélération Y avec deux décimales
        + separateur        # Séparateur entre chaque donnée
        + str(round(AZ, 3))  # Accélération Z avec trois décimales
    )
    # Envoi de la trame sur le port série
    print(trame)
    # Ce délai (en ms) permet de définir la fréquence
    # d'envoi des données (échantillonnage)
    sleep(200)
```

Remarque : il est interdit de prendre le point comme séparateur des données, ou comme préfixe, car c'est le séparateur décimal par défaut des cartes Micro:bit ! Il ne faut bien entendu pas choisir de caractère numérique (0-9), ni le signe « - ».

IV – Au lancement du logiciel

La première fenêtre qui s'ouvre permet de configurer la communication série entre le logiciel et la carte programmable, et de définir le protocole.

① Sélection du port (la carte programmable doit être raccordée, le port série correspondant apparaît dans la liste)

② Sélection de la vitesse de transmission (sélectionner la même que celle configurée dans le programme)

③ Reset sur connection : cocher cette case permet de redémarrer la carte programmable (Reset) lors du clic sur le bouton « Connection ». Cela fonctionne sur une carte Aduino par exemple.

④ Rafraîchir permet de relancer un scan des ports série disponibles si parfois vous aviez oublié de raccorder votre carte programmable au PC avant de lancer le logiciel

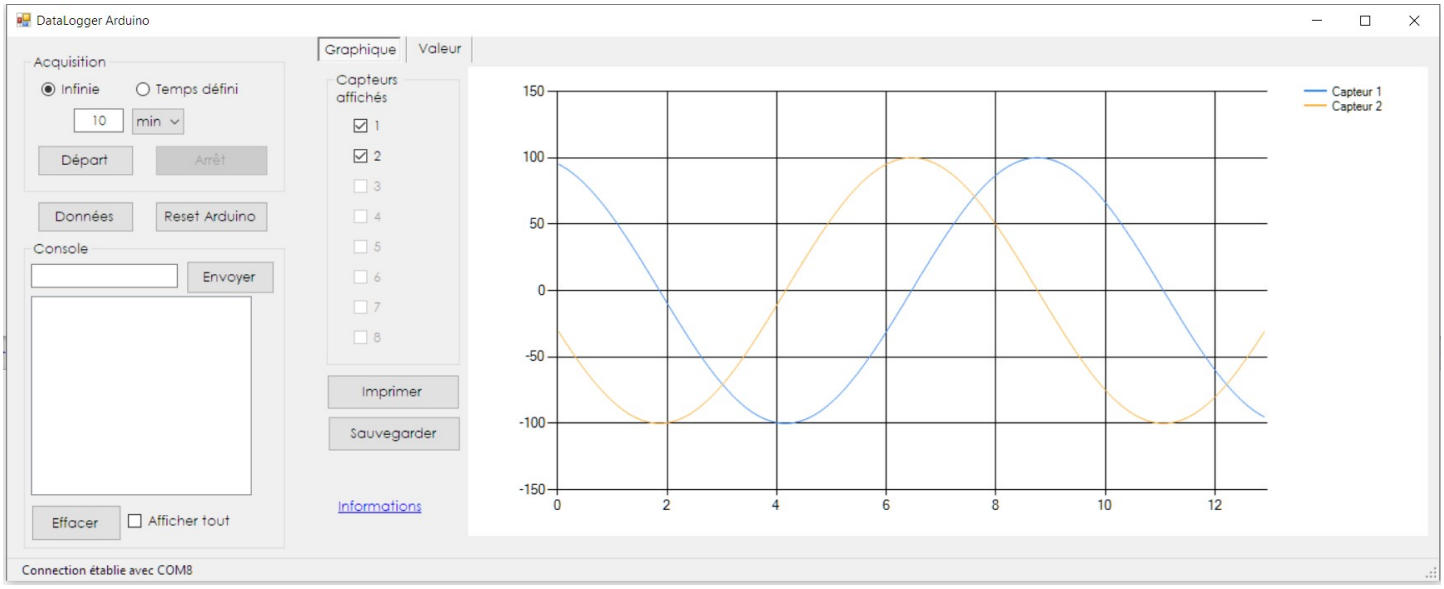
⑤ Séparateur et Préfixe : configure le séparateur de données sur une ligne de données et le préfixe de ligne de données. Ne choisir ni le point, ni un caractère numérique !

⑥ Nbre de capteurs : permet de définir combien au maximum de données vont transiter sur une ligne de données, correspondant au nombre de capteurs à enregistrer. De 1 à 8.

⑦ Connection : connecte le logiciel à la carte programmable, ferme cette fenêtre et ouvre la suivante.

V – Acquisition des données

La fenêtre suivante permet d'acquisition, le tracé, la sauvegarde et l'impression des données. Elle permet également d'afficher les messages de dialogue entre la carte programmable et le PC. Si tout s'est bien passé à l'étape précédente, le message `Connection établie avec COM8` apparaît en bas de la fenêtre.

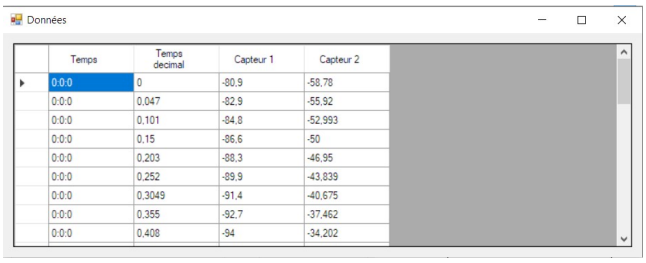


Acquisition :

L'acquisition peut se faire manuellement (durée non définie) ou être programmée sur une durée déterminée (en secondes, minutes, heures, jours). Le bouton « Départ » démarre l'acquisition, le bouton « Arrêt » la stoppe.

Un nouveau clic sur le bouton « Départ » après avoir fait une première acquisition relance une acquisition **en effaçant les données précédentes**.

Le bouton « Données » permet d'afficher la table des données de l'acquisition en cours.



Un copier-coller de cette table de données vers un tableur est possible.

Remarque : le temps est toujours affiché en secondes.

Le bouton « Reset Arduino » permet de redémarrer la carte Arduino pendant une acquisition.

Console :

Ce composant permet de dialoguer avec la carte programmable.

Toutes les données envoyées par la carte programmable et ne comportant pas de préfixe sont interprétées par le logiciel comme des éléments de communication et s'afficheront dans la console avec le préfixe [RX].

Il est également possible d'envoyer des caractères à la carte programmable en les saisissant dans la console puis en cliquant sur le bouton « Envoyer ». La carte programmable ne les interprétera que si vous l'avez programmée en conséquence.

Cocher la case « Afficher tout » permet de visualiser toutes les lignes reçues, y compris les lignes de données.

Le bouton « Effacer » n'efface que les communications affichées dans la console. Cela n'affecte pas les données.

Le graphique :

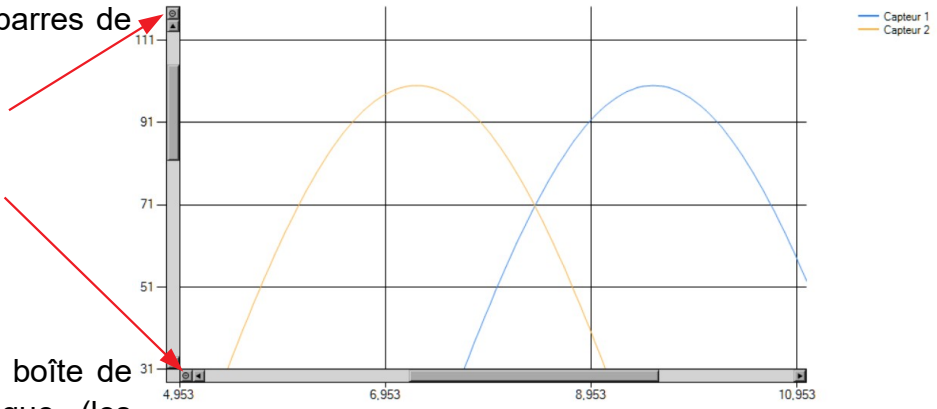
Le graphique se met à jour en temps réel lors de l'acquisition des données. Il faut pour cela cocher au moins un des capteurs à afficher.

A l'aide de la souris, il est possible de faire un zoom sur la zone graphique, aussi bien pendant l'acquisition qu'après.

Attention : si un zoom est réalisé pendant l'acquisition, alors la mise à jour dynamique du graphique est stoppée.

Après avoir réalisé un zoom, des barres de défilement x et y apparaissent.

Les boutons placés à l'extrémité des barres de défilement permettent de faire un zoom arrière sur l'axe concerné.

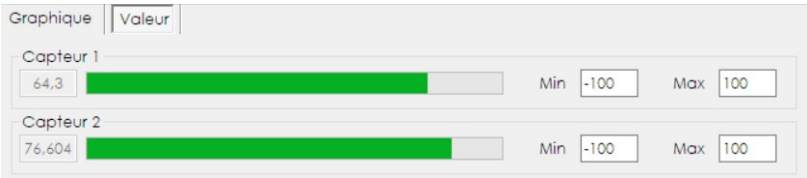


Le bouton « Imprimer » ouvre une boîte de dialogue d'impression du graphique (les barres de défilement ne seront pas imprimées en cas de zoom).

Le bouton « Sauvegarder » permet d'exporter TOUTES les données en .csv, format récupérable par un tableur. Le bouton « Données » vu précédemment permet un copier-coller direct des données dans un tableur, mais pas obligatoirement toutes.

Les valeurs :

L'onglet « Valeurs » permet d'afficher les données reçues sous forme de « BarGraphe » afin d'observer dans certains cas l'évolution des mesures.



Les cases « Min » et « Max » sont à saisir manuellement avec les valeurs minimale et maximale que sont sensés renvoyer les capteurs.

Pendant l'acquisition, le bargraphe évolue en fonction de la valeur reçue pour chaque capteur, la valeur instantanée est indiquée à gauche du bargraphe.

VI – Remarques techniques :

Sur la communication série

La communication série entre Arduino/Micro:bit et un PC est configurée par défaut avec :

- 1 bit start
- 8 bits de données,
- sans parité,
- avec un bit stop.

Un caractère envoyé via la port série nécessite l'envoi de 10 bits.

Une fin de ligne est caractérisée par les caractères

- Carriage Return (code ASCII 13) +
- Line Feed (code ASCII 10)

Dans l'exemple qui a été retenu pour cette notice, une ligne de données pouvait contenir au maximum 19 caractères : « X,-100.0,-100.000 » + CR + LF (16 en vérité car les fonctions sin et cos utilisées ici ne peuvent pas être simultanément égales à -1).

Une ligne de données nécessite l'envoi de 190 bits via le port série.

La fréquence d'envoi des données maximale qui pourra être obtenue dépendra de la vitesse de communication choisie :

- à 115 200 bits/s (bauds) on peut envoyer jusqu'à 606 lignes par seconde ;
- à 9 600 bauds, ce nombre d'échantillons descend à 50.

Sur la conversion analogique numérique

Une conversion Analogique – Numérique sur un Arduino n'est pas instantanée, elle prend environ 110 µs. Ce temps de latence est à prendre en compte pour évaluer l'échantillonnage du dispositif d'acquisition.

La conversion s'effectue sur une résolution de 10 bits (1024 points), donc renvoie un entier compris entre 0 et 1023. La précision de la conversion est de 1/1024 de la pleine échelle des tensions : 4,88 mV pour 5 V, 3,22 mV pour 3,3 V.