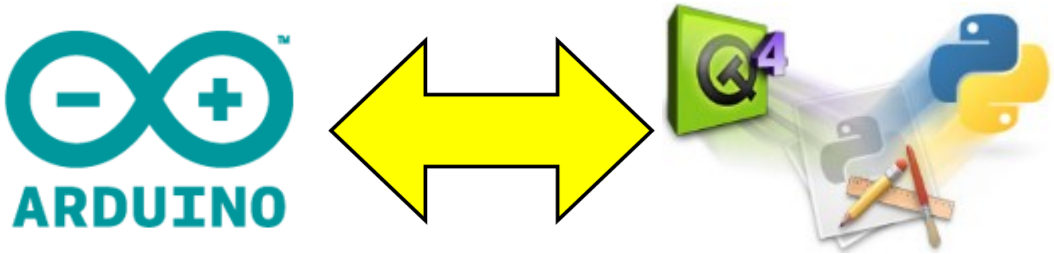


Ateliers Python+Qt et Arduino : Recevoir une chaîne avec paramètre numérique en provenance du PC.

par X. HINAULT
www.mon-club-elec.fr

Tous droits réservés – 2013.

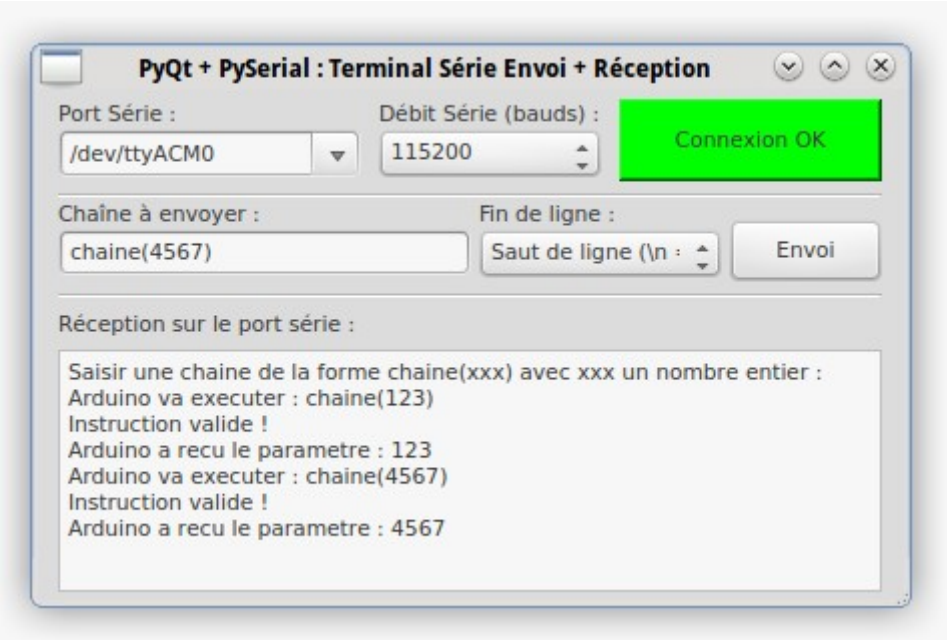
Document gratuit.
Ce support PDF d'atelier Python + Qt vous est offert.
Pour acheter d'autres supports d'ateliers Python + Qt rendez-vous ici :
http://www.mon-club-elec.fr/pmwiki_mon_club_elec/pmwiki.php?n=MAIN.PYQT

Vous avez constaté une erreur ? une coquille ? N'hésitez pas à nous le signaler à cette adresse : support@mon-club-elec.fr

Truc d'utilisation : visualiser ce document en mode diaporama dans le visionneur PDF. Navigation avec les flèches HAUT / BAS ou la souris.
En mode fenêtre, activer le panneau latéral vous facilitera la navigation dans le document. Bonne lecture !

1. PyQt + Arduino : Recevoir une chaîne avec paramètre numérique en provenance du PC.

Par X. HINAULT – Janvier 2013 – www.mon-club-elec.fr – Tous droits réservés



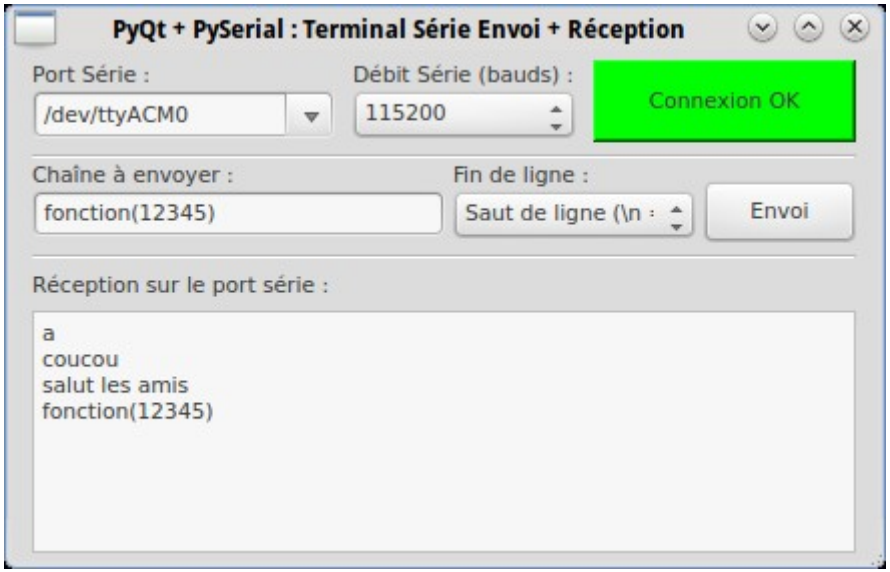
Ce que nous allons faire

- Dans ce tuto, je vous montre comment envoyer une chaîne de caractères avec paramètre numérique au format **chaine(xxx)** à partir d'une interface PyQt et à décoder la chaîne reçue pour en extraire le paramètre numérique du côté Arduino, à l'aide de ma librairie Arduino Utils.

Ressources utilisées

Côté PC : l'interface Python + Qt utilisée :

- On utilisera ici simplement le terminal série « Arduino-like » écrit en PyQt :



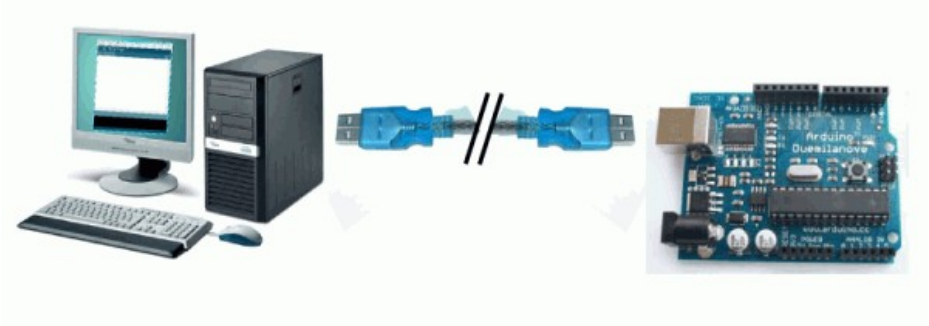
Côté Arduino : le programme utilisé :

- On utilisera un programme Arduino basé sur librairie Utils que j'ai écrite et qui permet d'extraire facilement les paramètres numériques d'une chaîne de la forme chaine (val1, val2, val3, ..., valn)
- Cette librairie dispose de plusieurs fonctions utiles facilitant grandement la réception de chaînes passant des valeurs numériques ou non en paramètre sous la forme chaine(xxx). Noter que chaîne est totalement paramétrable, ce qui offre une souplesse maximale et permet surtout de définir de véritables « fonctions » qui seront reconnues en réception sur le port série.

2. Matériel nécessaire pour cet atelier Python+Qt et Arduino

Pour cet atelier, vous aurez besoin de tout ou partie des éléments suivants pour pouvoir réaliser les exemples proposés :

De l'espace de développement Arduino



L'espace de développement Arduino associe :

- un ordinateur sous Windows, Mac Os X ou Gnu/Linux (Ubuntu)
- avec le logiciel Arduino installé (voir : http://www.arduino.cc/)
- un câble USB
- une carte Arduino UNO ou équivalente.

disponible chez : http://shop.snootlab.com/ ou http://www.gotronic.fr/

3. Pré-requis : Installation et présentation de la librairie Utils (une librairie www.mon-club-elec.fr)

La librairie utilisée

- Pour la suite, nous allons utiliser une librairie que j'ai écrite et qui permet de facilement recevoir une fonction avec un ou plusieurs paramètres sur le port série. Cette librairie s'appelle Utils
- Cette librairie va permettre de recevoir des chaînes de la forme chaine(val1, val2, val3, .. , valn) et d'extraire facilement les paramètres numériques reçus.

Télécharger la librairie

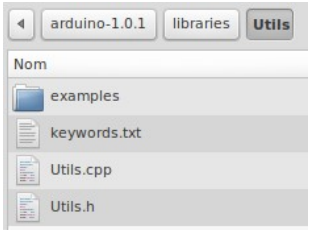
- L'archive est disponible ici : librairie Utils

Documentation de la librairie

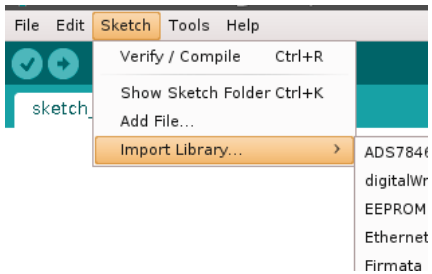
- La documentation de la librairie est ici : http://www.mon-club-elec.fr/pmwiki_reference_lib_arduino_perso/pmwiki.php?n=Main.Utils

Installation

- Télécharger l'archive. au format zip ou autre. L'extraire
- Vérifier que le nom du répertoire de la librairie est strictement le même que le nom du fichier *.h ou *.cpp principal. Corriger au besoin. Ici le nom est **Utils**
- Copier/coller le répertoire de la librairie dans le répertoire libraries de votre répertoire Arduino



- Relancer Arduino et vérifier que la librairie est présente dans le menu **Sketch > ImportLibrary.**



Le constructeur principal

- Le constructeur principal se nomme Utils et est de la forme :

```
Utils utils ;
```

Fonctions de la librairie

Fonctions de réception de chaîne de caractères sur le port Série :

String waitingString (boolean debugIn) : réception d'une chaine sur le port Série

String waitingString () : réception d'une chaine sur le port Série

void waitForString(String stringForWaitIn) : attente de la réception d'une chaine précise sur le port Série

Fonctions d'analyse de chaîne de caractères :

String testInstructionString (String chaineTest, String chaineRefIn): extraction d'un paramètre texte

boolean testInstructionLong (String chaineReception,String chaineTest, int nbParam, long paramsIn[]) : extraction d'un ou plusieurs paramètres entiers

Fonctions de conversion de chaînes :

long stringToLong (String chaineLong)

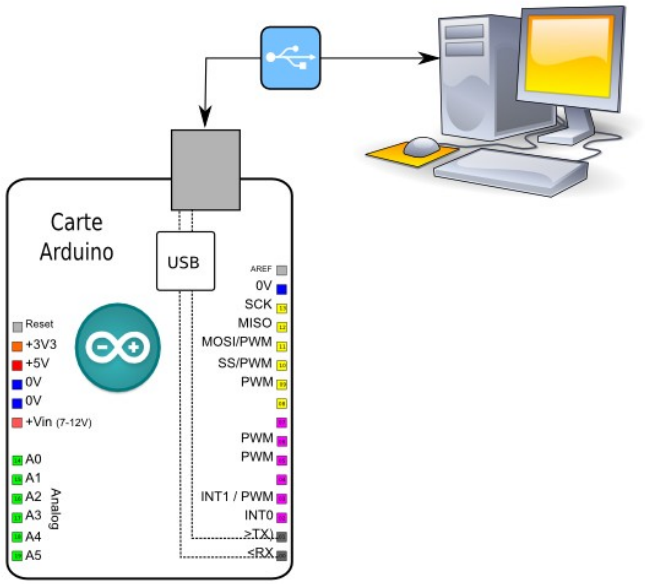
float stringToFloat (String chaineFloat)

Code d'exemple

```
#include <Utils.h>

Utils utils; // déclare objet racine d'accès aux fonctions de la librairie Utils

String chaineIn; // déclare un String de réception

void setup() {

  Serial.begin(115200); // Initialisation vitesse port Série
  // Utiliser vitesse idem cté Terminal série

}

void loop() {

    //chaineIn=utils.waitingString(false);
    chaineIn=utils.waitingString();

    Serial.println("Arduino a recu : " + chaineIn);

}
```

4. Le montage à utiliser

- Dans cet atelier, nous allons faire quelque chose de très simple : la carte Arduino devra simplement être connectée au PC utilisé (poste fixe, netbook, RaspberryPi, ...)



5. Le programme Arduino : Recevoir sur le port série une chaîne avec un paramètre numérique :

Ce qu'on va faire ici...

- La première chose nécessaire ici est d'être en mesure de recevoir sur le port série une chaîne au format chaine(val1, val2) et d'extraire automatiquement les valeurs ainsi reçues.
- Plusieurs possibilités :
  - soit tout coder de zéro,
  - soit utiliser la librairie RunF déjà présentée par ailleurs,
  - soit utiliser ma librairie Arduino Utils qui permet une gestion simplifiée de la réception de chaîne avec paramètres numériques
- Les librairies proposées ici ne sont pas des librairies standards, et elles devront donc être installées au préalable. Ma librairie Utils est disponible ici : http://www.mon-club-elec.fr/pmwiki_reference_lib_arduino_perso/pmwiki.php?n=Main.HomePage

Entête déclarative

Inclusion des librairies utiles

- On commence par inclure la librairie Utils

Déclaration broche utilisée

- On déclare une constante de broche pour la LED utilisée

Variables utiles

- On déclare :
  - un objet String pour la stocker la chaîne reçue en réception
  - un tableau de long destiné à stocké les paramètres reçus en réception : la taille du tableau fixe le nombre maximum de paramètres reçus.

```
#include <Utils.h> // inclusion de la librairie

Utils utils; // déclare objet racine d'accès aux fonctions de la librairie Utils

String chaineReception=""; // déclare un String
long params[6]; // déclare un tableau de long - taille en fonction nombre max paramètres attendus
```

Fonction setup()

Configuration broche utilisée

- On configure la broche de LED en sortie

Initialisation série

- On initialise la communication série à 115200 bauds

Code initial

- On affiche un simple message invitant à saisir une chaîne de la forme LED(pwm)

```
void setup() {

  Serial.begin(115200); // Initialisation vitesse port Série
  // Utiliser vitesse idem coté Terminal série

  Serial.println("Saisir une chaine de la forme chaine(xxx) avec xxx un nombre entier :"); // message initial

} // fin setup
```

Fonction loop()

Réception de la chaîne sur le port série

- La librairie Utils intègre la fonction waitingString() qui attend sur le port série une chaîne jusqu'à l'arrivée d'un saut de ligne (« \n ») : la chaîne reçue est stockée dans un objet String

Extraction des paramètres numériques

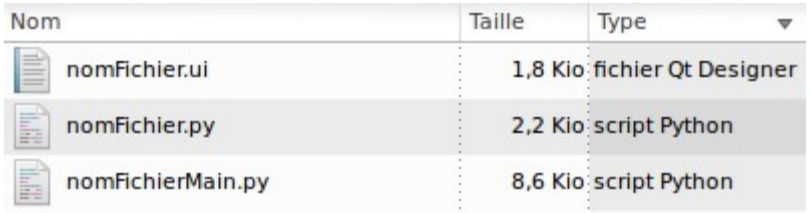
- Ensuite, à l'aide d'une fonction, on teste si la chaîne n'est pas vide :
  - on appelle alors la fonction testInstructionLong() en lui passant en paramètre :
    - la chaîne reçue,
    - la racine de la chaîne à reconnaître (la partie **chaine(** d'une chaine à reconnaître de la forme chaine(xxx))
    - le nombre de paramètre : 1 pour 1 paramètre, 2 pour 2, etc...
    - le tableau de stockage : celui que nous avons déclaré en début de programme
  - la fonction renvoie true si l'analyse est correcte, autrement dit qu'une chaîne **chaine(valeur)** a bien été reçue

```
void loop() {

    //chaineReception=utils.waitingString(true);// avec debug
    chaineReception=utils.waitingString();// sans debug

    if (chaineReception!="") { // si une chaine a été reçue

       if(utils.testInstructionLong(chaineReception,"chaine(",1,params)==true) { // si chaine chaine(valeur) bien reçue

              Serial.println("Arduino a recu le parametre : " + (String)params[0]);

       } // fin si testInstructionLong==true

    } // fin // si une chaine a été reçue

} // fin loop
```

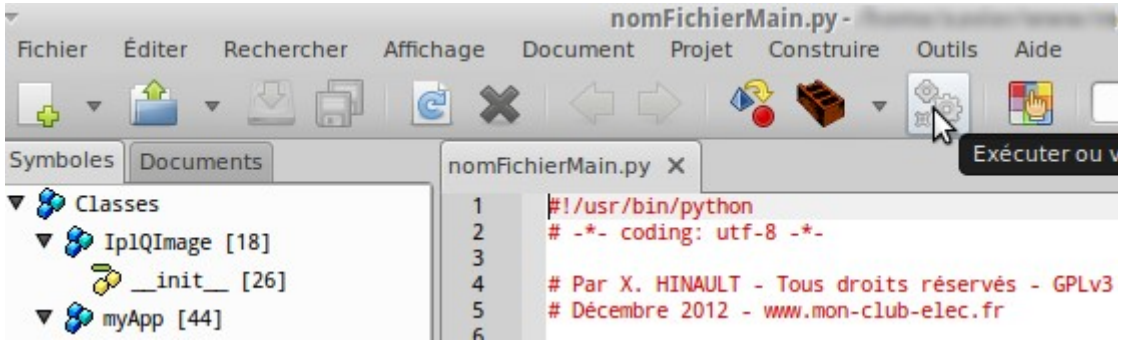
6. L'interface Python + Qt utilisée : Terminal série « Arduino like » écrit avec PyQt

Exécution

- Cette interface est disponible ici : terminal série « Arduino-like » écrit en PyQt
- vous devez disposer dans votre répertoire des 3 fichiers suivants (adapter le nom à votre situation) :
  - le fichier **nomFichier.ui** correspondant au fichier généré initialement par QtDesigner (ce fichier n'est pas obligatoire pour l'exécution)
  - le fichier python **nomFichier.py** correspondant au fichier généré à partir du fichier nomFichier.ui avec la commande pyuic4
  - et enfin le fichier python contenant le code actif à proprement parler, le fichier **nomFichierMain.py**

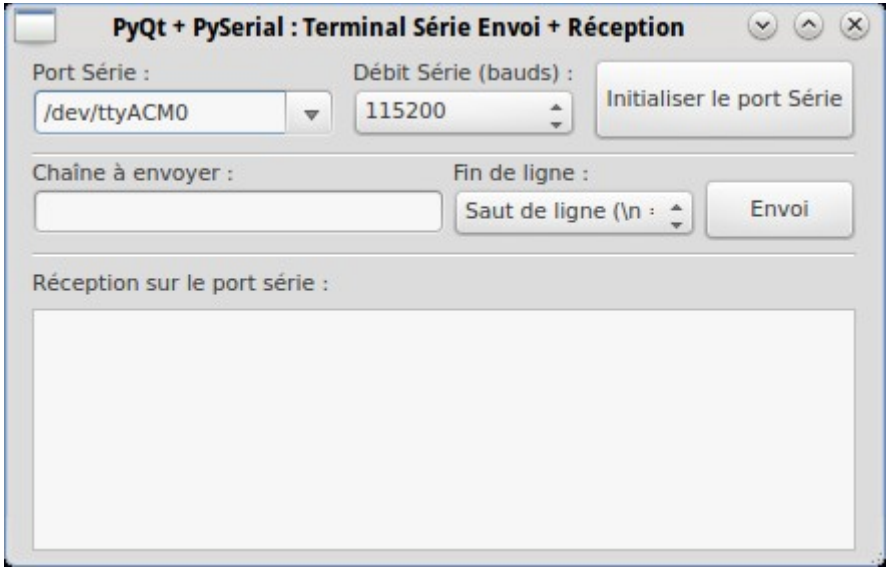


- Lancer l'exécution du code :
  - soit par simple double clic sur le fichier **nomFichierMain.py**
  - soit en ligne de commande avec **./nomfichierMain.py** (se placer au préalable dans le bon répertoire avec le commande **cd chemin** ...)
  - ou directement depuis l'éditeur Geany avec le bouton  , le fichier **nomFichierMain.py** étant ouvert.

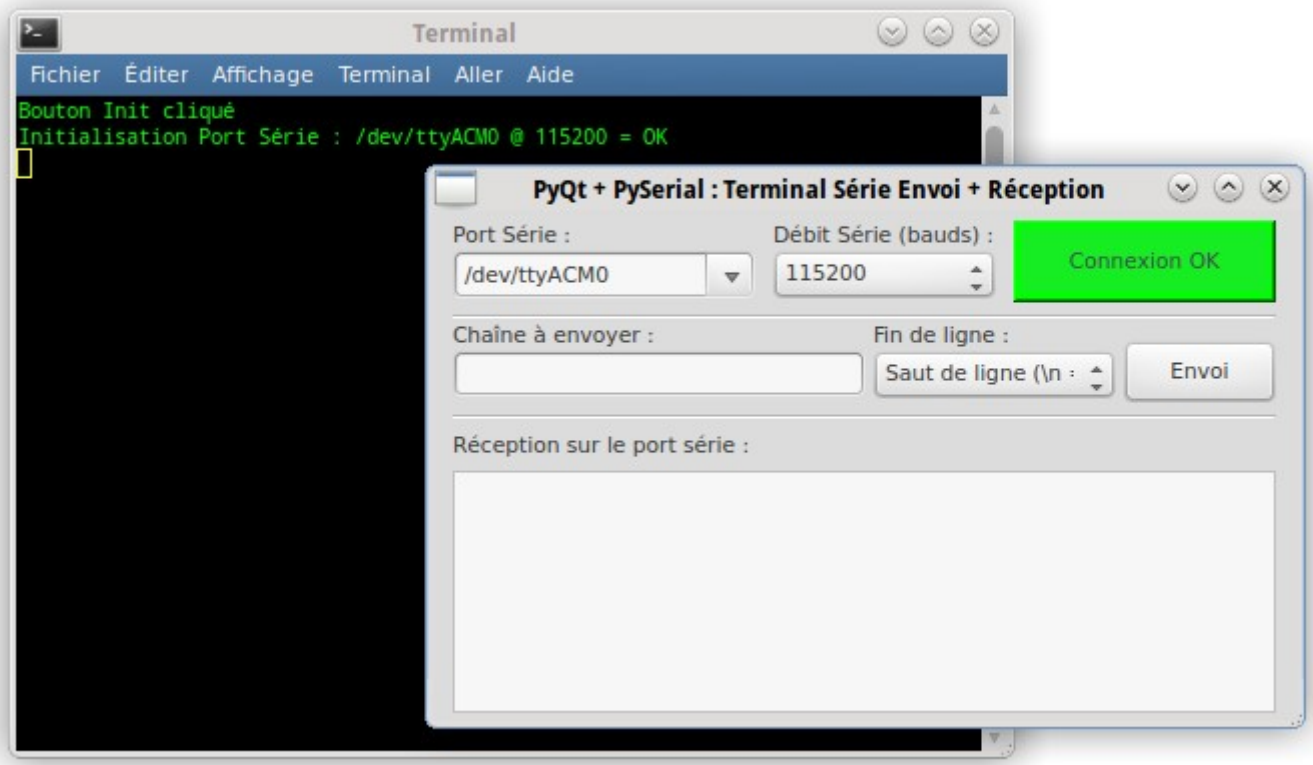


Résultat

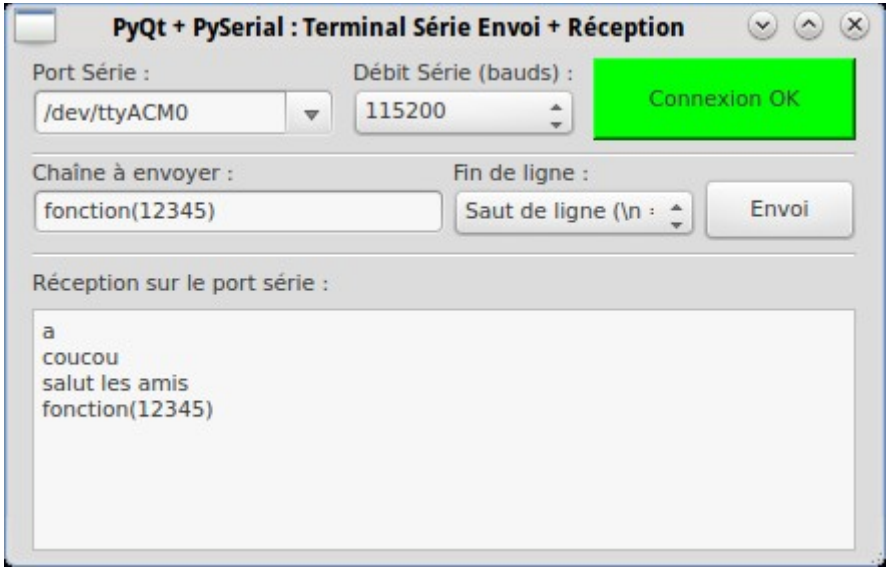
- On obtient au lancement (remarquer que notre terminal Série se lance même si la carte Arduino n'est pas connectée, ce qui n'est pas le cas avec le Terminal Série de l'IDE Arduino...) :



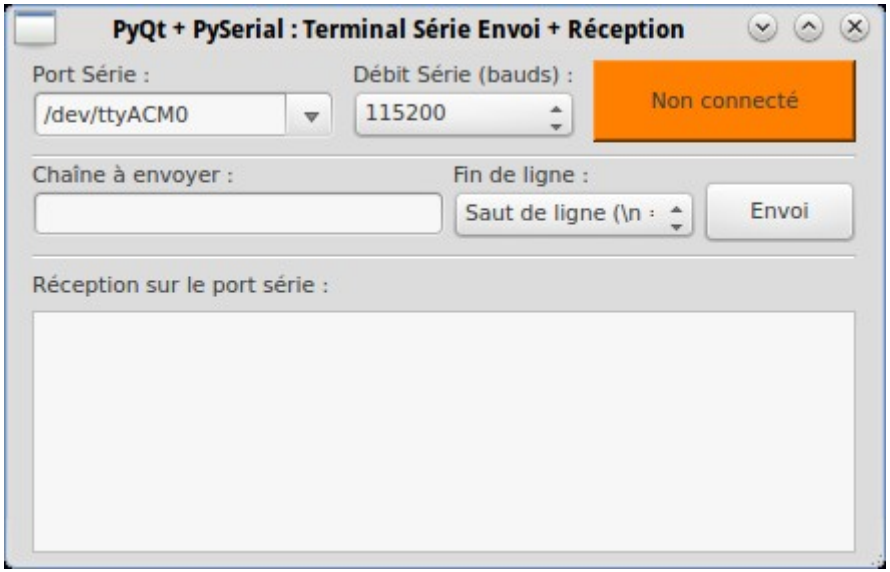
- Connecter une carte Arduino sur le port USB, la programmer avec le code ci-dessous ou équivalent... Régler le port, le débit voulu. Lorsque l'on appuie sur le bouton Initialiser, cela donne si tout se passe bien :



- Ensuite, il suffit de saisir des chaînes dans le champ de saisie et de cliquer sur « envoi ». Si vous utilisez le programme « miroir série » que je vous propose ci-dessous, vous voyez qu'Arduino renvoie la chaîne qui s'affiche dans le champ de réception :



- Si un problème survient, on obtient (remarquer que le programme ne s'interrompt pas... et qu'un nouvel appui est possible...) :



Au final, un code léger qui offre une fonctionnalité assez puissante en pratique.

7. Fonctionnement

- Programmer la carte Arduino avec le code ci-dessus
- Lancer l'interface Python + Qt et se connecter à la carte Arduino en réglant le débit sur 115200 et caractère de fin de ligne d'envoi sur «saut de ligne \n »
- A présent, saisissez une chaîne de la forme **chaine(123)** dans le champ d'envoi puis clic sur envoi : vous devez voir alors voir le message Arduino indiquant la valeur numérique décodée