Programmer en Python pour la carte Arduino

Remarque : Pour les différentes installations, vous aurez besoin d'une connexion internet et des fichiers zip **Nanpy_Arduino.zip** et **Nanpy_Python.zip**

1. Installation d'ARDUINO

a. Installer Arduino (Windows)

Installer Arduino en téléchargeant la version adaptée (Windows, Mac, Linux,..) sur ce lien : https://www.arduino.cc/en/Main/Software
Cliquer sur « I Agree » Cliquer sur « Next » Cliquer sur « Install »

b. Installer la librairie d'exemple Nanpy pour permettre d'utiliser la carte Arduino avec le langage Python

Ouvrir le logiciel **Arduino**

Dans l'application Arduino, à partir du menu [Croquis][Inclure une librairie][Ajouter la librairie .ZIP], installer la librairie « Nanpy_Arduino.zip » (fichier fourni dans le dossier « Codes Python pour second cycle)

Pour utiliser Nanpy :

Depuis le menu [Fichier][Exemples], ouvrir l'exemple " Nanpy"

Vérifier que le type de carte choisi est bien "Arduino/Genuino Uno" depuis le menu [Outils][Type de carte]

Sélectionner le port de communication de la carte Arduino, depuis le menu [Outils][Port] (La carte doit être connectée). **Bien noter ce numéro de port qui sera utile pour programmer en Python**

Téléverser la librairie " Nanpy" dans la carte Arduino, depuis le menu [Croquis][Téléverser]

Programmer en Python pour la carte Arduino

2. Installation de Python et Pyzo

L'installation de Python se fait en plusieurs étapes :

- Installation de l'environnement Python

- Installation du logiciel Pyzo (adapté à la programmation scientifique) et configuration du shell (interpréteur)

- Installation des bibliothèques dans Pyzo (pour calculs, tracés de courbes, communication avec la carte Arduino,…)

a. Installer l'environnement Python

Installer Python en téléchargeant la bonne version (Windows 32 bits, Windows 64 bits, Mac, Linux,..) sur ce lien : https://www.python.org/downloads/
Choisir "Customize installation » puis cocher la case « All Users » dans « Advanced Options ».

b. Installer Pyzo

Installer Pyzo en téléchargeant la bonne version (Windows 32 bits, Windows 64 bits, Mac, Linux,..) sur ce lien : https://pyzo.org/start.html

c. Configuration du shell (ou interpréteur)

Ouvrir **Pyzo**
Dans la partie droite, l'éditeur signale si un environnement python a été détecté (dans ce cas cliquer sur « detect » pour valider le choix).
Sinon il faut aller dans le menu Shell > Edit shell configurations, une fenêtre s'ouvre et il faut renseigner la ligne exe avec le bon chemin parmi les choix proposés (Python 3.7).

Pour choisir la langue française : menu Settings > Select language
Redémarrer Pyzo pour que les changements prennent effet.

d. Installation des bibliothèques

Ouvrir **Pyzo**
Dans le shell (qu'on peut faire glisser en haut par commodité), écrire **une à une** les commandes suivantes et valider à chaque fois (**Attention** : il faut être connecté à Internet)

pip install numpy
pip install matplotlib
pip install scipy

!!! Pour nanpy !!! **pip install D:\Nanpy_Python.zip**
(Indiquer chemin du fichier, ici sur la racine d'une clé USB sur le D :)

numpy et **scipy** sont des bibliothèques pour des calculs scientifiques
matplotlib contient les fonctions pour tracer des graphiques (avec le module pyplot)
nanpy permet la communication avec la carte Arduino

Programmer en Python pour la carte Arduino

3. Comment programmer en Python avec la carte Arduino

Rappel : La librairie Nanpy.zip doit être téléversée au préalable dans **Arduino** (voir ci-dessus)

Ouvrir **Pyzo**
Dans l'éditeur de script pour l'écriture du code, il faudra d'abord entrer les lignes suivantes pour que les applications utilisant Python communiquent avec la carte Arduino:

from Nanpy import ArduinoApi

from Nanpy import SerialManager

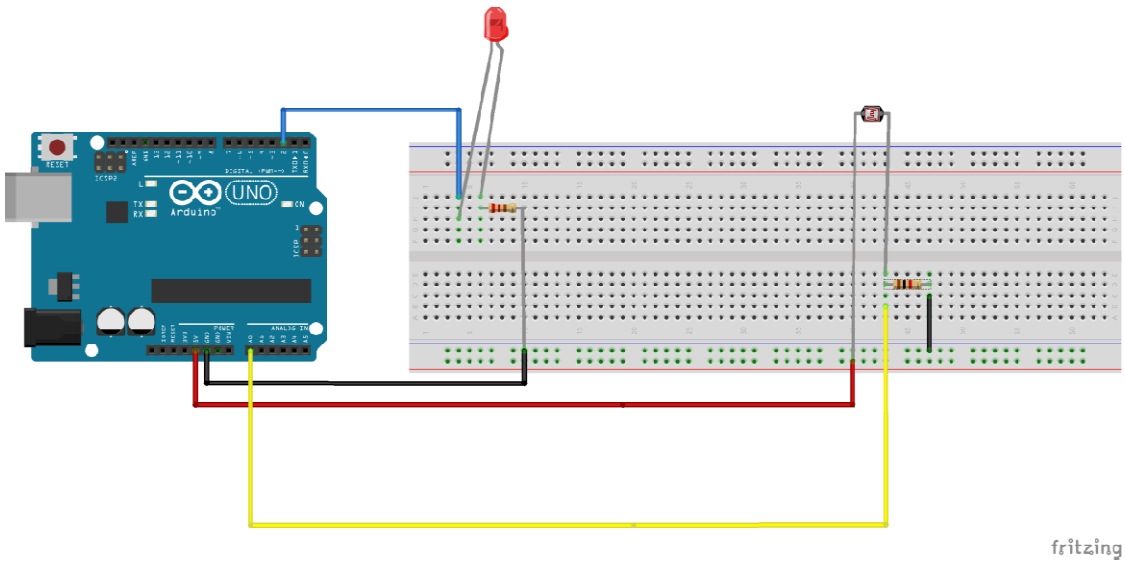
connection = SerialManager(device='com3 ')

a= ArduinoApi(connection=connection)

Remarque : le numéro de port, indiqué en rouge sur la 3ème ligne ci-dessus, est celui utilisé par la carte. Il est identifiable rapidement avec l'application « Arduino » par le menu [Outils][Port]

Il est alors possible d'utiliser toutes les commandes usuelles d'Arduino : il faut pour cela écrire **a.** devant ces instructions. Il peut aussi être utile d'importer la fonction **sleep** du module **time** pour faire des « pauses » (en secondes) dans l'exécution du programme.

Ci dessous un exemple simple de code qui permet de programmer un dispositif d'allumeur de réverbère en langage Arduino puis en langage Python.
Les codes Python spécifiques à Arduino sont écrits en gras.



Programmer en Python pour la carte Arduino

Avec Arduino	Avec Python

/*
Allumage d'une LED pour un faible éclaire-ment d'une photorésistance (LDR)
*/
	from nanpy import ArduinoApi
	from nanpy import SerialManager
	from time import sleep
	
	#lignes de commande pour connection à la carte Arduino (en indiquant le bon nu-méro de port)
	connection = SerialManager (de-vice='COM7')

int Valeur_A0; float Tension_A0; // Déclaration des va-riables : Valeur mesurée en AO (montage avec LDR), et la tension qui sera calculée à partir de cette valeur	a = ArduinoApi(connection=connection) # Remarque : il n'est pas nécessaire de déclarer les variables
void setup(){	

pinMode(2,OUTPUT);
Serial.begin(9600); // Moniteur série
}
	a.pinMode(2,**a.OUTPUT**)

void loop(){
Valeur_A0=analogRead(A0);
// lecture de la tension aux bornes de la LDR
	while True:
	Valeur_A0=**a.analogRead(0)**

Tension_A0=(float)Valeur_A0*5/1023;	Tension_A0=Valeur_A0*5.0/1023
// Si la luminosité est suffisante, la LED reste éteinte. En dessous d'un certain seuil, elle s'allume	

if (Tension_A0<4.0){
digitalWrite(2,HIGH);
}
	if (Tension_A0<4.0):
	a.digitalWrite(2,**a.HIGH**)

else{ digitalWrite(2,LOW) ; }	else: **a.digitalWrite**(2,**a.LOW**)

// Affichage sur Moniteur Série
Serial.print(« Tension LDR : ");
Serial.println(Tension_A0);
	print ('Tension LDR : ' ,Tension_A0)

// on attend 250 ms avant la prochaine boucle
delay(250);
}
	sleep(0.25)

Il est aussi possible d'utiliser le langage Python pour aller plus loin : mettre les valeurs mesurées dans des listes, tracer des graphiques et exploiter des données…

Programmer en Python pour la carte Arduino

Vous trouverez dans ce lien Youtube des explications détaillées ainsi que des fichiers à télécharger (bibliothèques, tutoriels, exemples d'activités Arduino avec les codes Python et codes Arduino équivalents,…) :

https://www.youtube.com/watch?v=oOXCgJFlUIw