Créez des interfaces graphiques dans vos applications programmées en Python

Cédric Perrot

▶ **To cite this version:**

Cédric Perrot. Créez des interfaces graphiques dans vos applications programmées en Python. 14. Journées de la mesure et de la métrologie, INRAE, Oct 2016, Blois, France. hal-03889125

HAL Id: hal-03889125
https://hal.inrae.fr/hal-03889125v1

Submitted on 7 Dec 2022

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Créez des interfaces graphiques dans vos applications programmées en Python

PERROT Cédric

La Chaussée-Saint-Victor, 12 Octobre 2016

SOMMAIRE

- Origine du besoin
- Python
- Lier une interface graphique à Python
- Distribuer son application Python sous Windows
- Applications réalisées

01

Origine du besoin

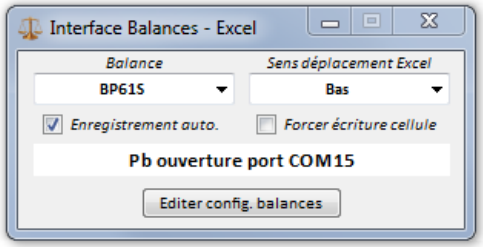
Pourquoi développer une interface graphique ?

Distribuer une application informatique auprès d'utilisateurs pas forcément expérimentés

Comment rendre conviviale une application informatique ?

Création de **fenêtres graphiques** constituées d'éléments visuels (**widgets***) pour que l'opérateur :

- renseigne des données
- génère des actions (ex : clic sur un bouton, sélection d'un item d'une liste déroulante)
- visualise des résultats (texte, tableaux, courbes)
- etc.



NATIONAL INSTRUMENTS LabVIEW™ très répandu dans la communauté des programmeurs

Comment créer des interfaces de dialogue avec Python?

Widget : élément visuel d'une interface graphique (bouton, barre de défilement, liste déroulante, etc.)

02
Python

Pourquoi choisir Python?

- Gratuit, libre et performant
- Pour automatiser des tâches quotidiennes
- Tutoriels et communauté de programmeur importants
- Nombreux modules disponibles y compris par les fabricants de matériels
- Très répandu dans le milieu scientifique

↳ *Exemples d'utilisation dans mon unité* :

➢ OpenAlea (Plateforme d'analyse, de visualisation et de modélisation du fonctionnement et de la croissance de l'architecture des plantes)

➢ LAMi (Logiciel d'Analyse du MIcroclimat d'un stade basé sur un modèle qui calcule la zone du terrain à éclairer et le temps d'application des rampes lumineuses)

➢ **Traitement d'images** (NumPy , OpenCV Python)

✓Simple à installer sur

✓Python par défaut sur *(ex : Raspberry Pi)*

Quelle version de Python?

python™

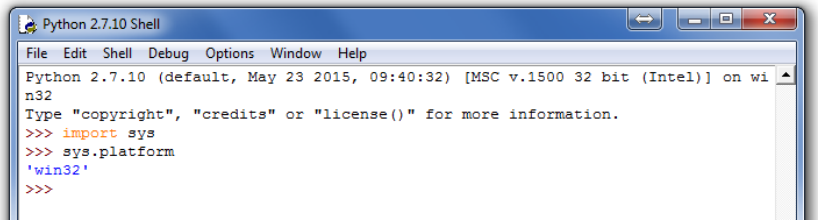
Version 2.7

Téléchargement de Python **:** https://www.python.org/download/releases/2.7

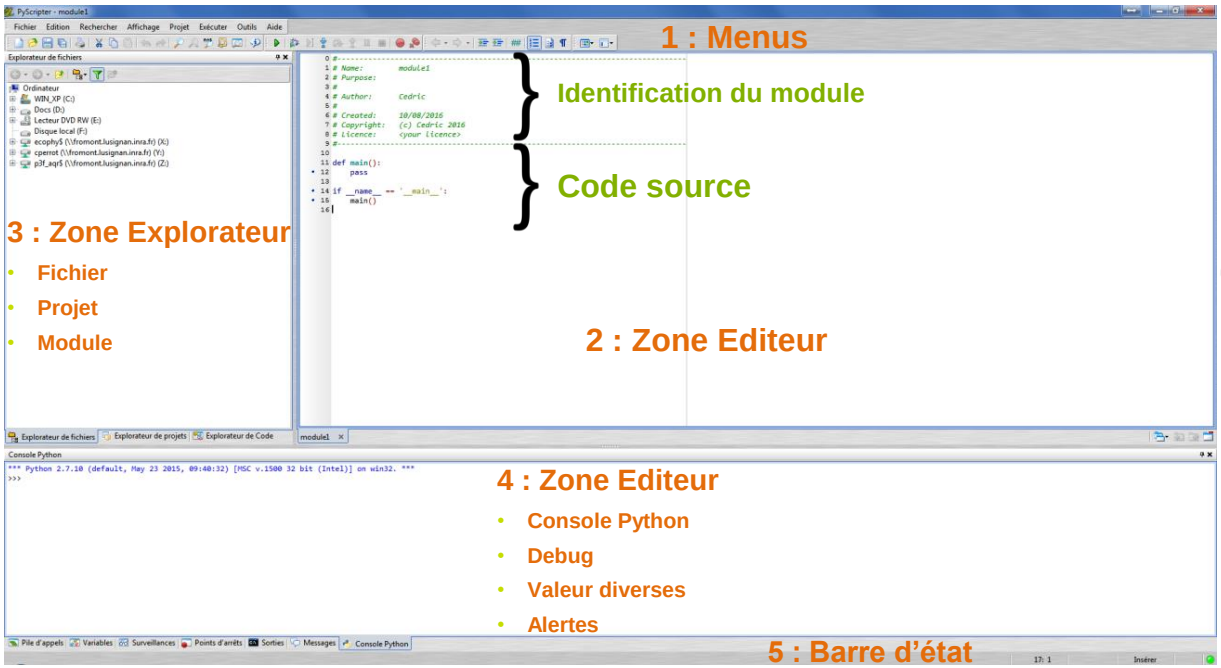
Environnement de développement - IDE Python

IDE pour Integrated Development Environment

- IDE Python installé par défaut (**IDLE**)



- Pour débuter :



PyScripter

Equivalent à :

spyder

ubuntu

- Niveau avancé :

eclipse + PyDev

Development Environment for Python
eric

Autres IDEs **:** https://wiki.python.org/moin/IntegratedDevelopmentEnvironments

Définitions

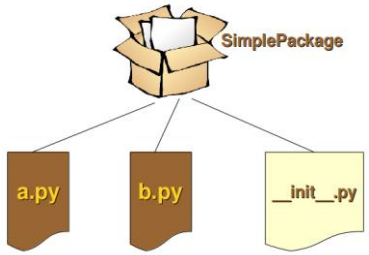
- **Fonction** : ensemble d'instructions avec un nom et des arguments optionnels, qui peut renvoyer des valeurs

- **Module** : ensemble de fonctions et/ou classes regroupées dans un fichier

- **Script** : module qui contient du code source qui lui permet d'être directement exécuté comme un programme

- **.py** : modifiable
- **.pyc** : compilé (généré automatiquement quand le module est importé)
- **.pyw** : exécutable sans apparition du terminal (sous Windows)

- **Package** : dossier qui regroupes des modules (y compris __init__.py)



- **Import** : importe des fonctions d'un autre module (depuis PYTHON PATH interne à python OU depuis un module quelconque)

Structure d'un module

Module.py

Import

```
import module                          # toutes les fonctions d'un module
from module import fonction            # une fonction d'un module
from package.module import fonction    # une fonction d'un module contenu dans un package
```

Fonction

```
def fonction(a,b,c) :
    valeur = a+b+c
    return valeur
```

Corps principal du programme

```
val = fonction(1,2,3)
print val
if __name__ == '__main__' :
    # Instruction exécutée uniquement à partir de ce script
    print 'Execution du script'
```

Ex : Calcul_volume_sphere.py

```
import math

def cube(n) :
    return n**3

def volSphere(r) :
    vs =  4 * math.pi * cube(r) / 3
    return vs

r = input( ' Entrez la valeur du rayon : ')
if __name__ == '__main__' :
    print  'Volume sphère = ', volSphere(r)
```

03

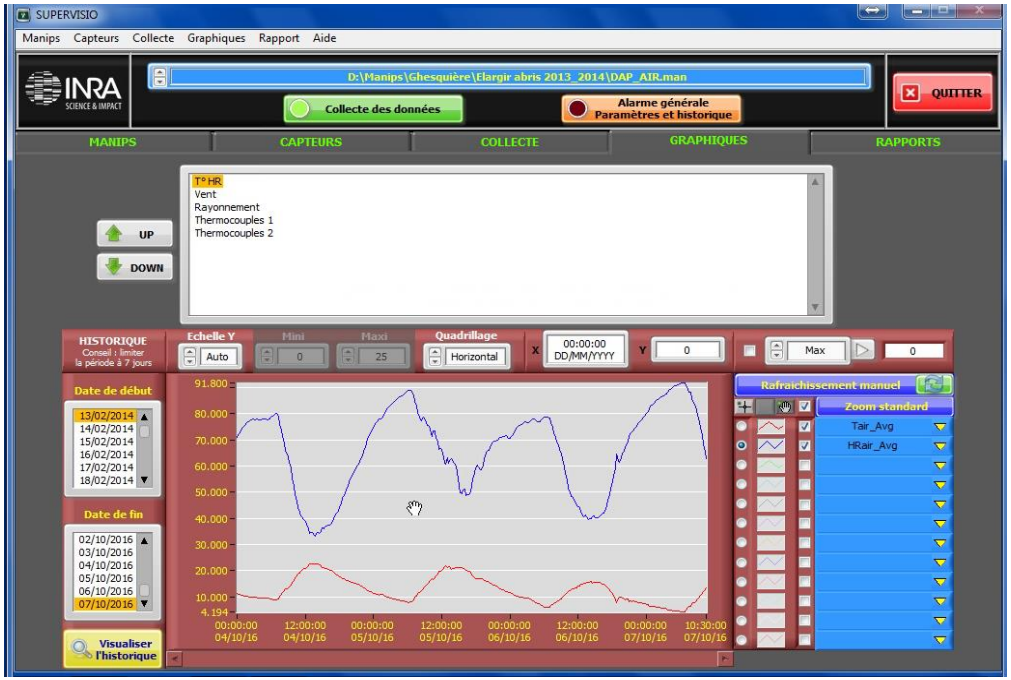
Lier une interface graphique à Python

Interface graphique - GUI

GUI pour Graphical User Interface

GUI composée de :

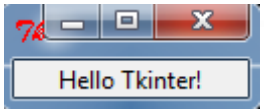
- **Fenêtres**
- **Widgets**
- **Menu**
- **Système de pointage**



Exemple d'interface graphique générée avec Labview

Tkinter

Module par défaut dans Python



- Eléments graphiques (widgets) peu esthétiques
- Programmation séquentielle imposée l'instruction mainloop()
- **Obligation d'écrire le code des fenêtres et des widgets**

Conseil : Utiliser Tkinter pour des scripts simples

PyQt : concevoir et lier des GUIs avec Python

Avantages d'utiliser PyQt

PyQt dispose du :

- Logiciel **Qt Designer** pour créer des fenêtres et positionner des widgets de manière visuelle
  - Gain de temps et d'esthétique
  - Evite les tâches répétitives d'écriture du code des éléments graphiques
- Module **pyuic.py** pour convertir les éléments graphiques en code Python

Installation de PyQt

Choisir **PyQt4** pour version de **Python 2.7**

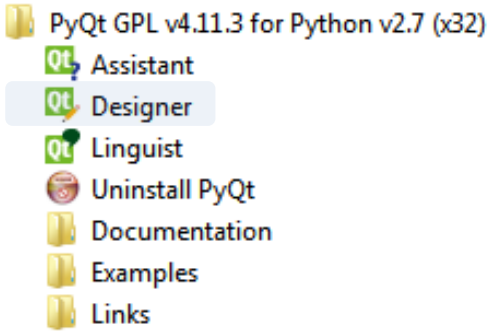
https://www.riverbankcomputing.com/software/pyqt/download

PyQt4-4.11.4-gpl-Py2.7-Qt4.8.7-x64.exe Windows 64-bit installer

PyQt4-4.11.4-gpl-Py2.7-Qt4.8.7-x32.exe Windows 32-bit installer

PyQt-x11-gpl-4.11.4.tar.gz Linux source

Programmes Windows



Designer

- Construire des GUIs à partir de composants http://doc.qt.io/qt-4.8/qtgui-module.html
- Composer et personnaliser vos fenêtres avec des widgets

Pour en savoir plus sur Qt Designer : https://openclassrooms.com/courses/programmez-avec-le-langage-c/modeliser-ses-fenetres-avec-qt-designer

Convertir une fenêtre de Qt Designer en Python

Développeur (vous)

dessine →

.ui

Fichier **Qt Designer**
*(ex : **IHM.ui**)*

exécute →

programme ↓

.bat

.py → Import → .py *principal* ← Import ← _UI.py

Autres fichiers sources écrits par vous

(ex dans le module principal :
from** PyQt4 **import** QtGui, QtCore
***from** IHM_UI **import** Ui_MainWindow**)*

Code source de la fenêtre généré automatiquement
*(ex **: IHM_UI.py**)*

Script pour automatiser la conversion d'un fichier .ui en .py

⚠ Idéal de mettre tous les fichiers .ui, .bat et .py dans le même dossier

Script de conversion .bat

Convertisseur_UI en PY.bat - Bloc-notes

Fichier Edition Format Affichage ?

```
set folder=D:\Python\
set infile=IHM.ui

set outfile=%infile:.ui=_UI.py%
set sep=\
"C:\Python27\python.exe" "C:\Python27\Lib\site-packages\PyQt4\uic\pyuic.py" -x "%folder%%sep%%infile%" -o "%folder%%sep%%outfile%"
::explorer %folder%
::pause
```

Renseigner le chemin du dossier et le nom du fichier .ui

Conseil : Générer autant de scripts de conversion .bat que de fenêtres graphiques .ui

Gestion des événements

L'utilisateur effectue une **action** sur un **widget**
(ex : clic sur un bouton)

Signal — connect — Slot

A chaque événement, le **widget** émet un **signal**

Chaque signal émis peut être géré par une **fonction**, un **slot**

Code dans le script principal pour chaque widget : **object.connect (Widget, Signal, Slot)**

Exemple :

```
object.connect (bouton,  QtCore.SIGNAL("clicked()"), slot_bouton)
def slot_bouton() :
        Instructions
```

Tutoriel **:** http://tcuvelier.developpez.com/tutoriels/pyqt/signaux-slots-layouts

Construction du script principal

Import

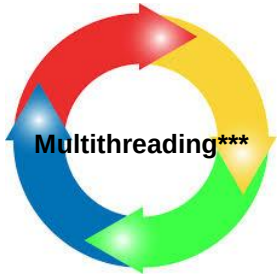
```
from IHM_UI import Ui_MainWindow
import threading
```

Classe dédiée à l'interface graphique

Class App_IHM

- Connexion des **signaux** à des **slots** des objets qui génèrent des événements
- Déclaration des threads
- Définitions des propriétés des objets
- **Slots** - Méthodes* liées aux événements des objets
- Méthode* Stop_threads

Classes Thread**



- Class Thread_main — **Tâche principale**
- Class Thread_port com — **Tâche dédiée à la gestion du port COM**
- Class Thread_log — **Tâche dédiée à l'affichage des messages utilisateur**
- etc.

Corps principal

- **QApplication** — **Exécution d'une boucle principale d'événement**
- **IHM.show()** — **Affichage de la fenêtre graphique principale**

* ***Méthode*** **:** fonction d'une classe
** ***Thread*** **:** tâche
*** ***Multithreading*** **:** exécution de plusieurs thread simultanément

04

Distribuer son application Python sous Windows

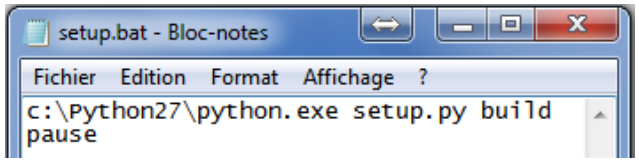
Transformer un script Python en exécutable

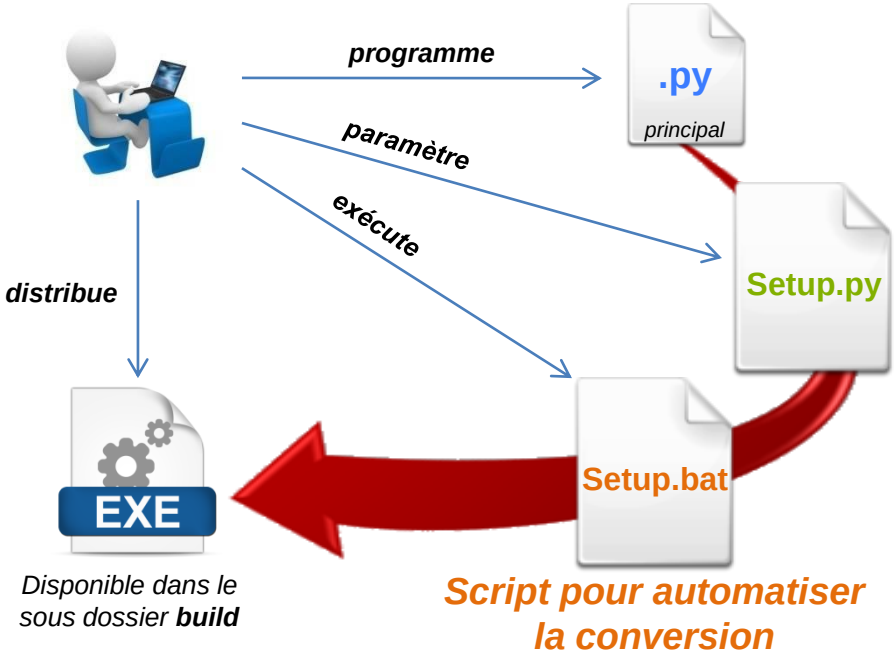
.exe pour distribuer le script sans installer Python

- **Cx_Freeze** https://pypi.python.org/pypi/cx_Freeze
- **Py2exe** https://sourceforge.net/projects/py2exe/files/py2exe/0.6.9/

Utilisation de Cx_Freeze

- ✓ Créez un fichier **Setup.py** (code disponible sur : http://python.jpvweb.com/mesrecettespython/doku.php?id=cx_freeze)
- ✓ Dans le même dossier, créez un fichier **Setup.bat**





*Disponible dans le sous dossier **build***

Script pour automatiser la conversion

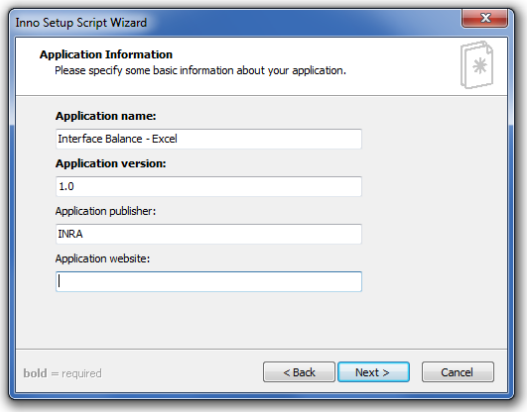
- ✓ **Nom du script python à transformer** (ex : Script_principal.py)
- ✓ **Personnalisez votre exécutable :**
  - **Icône (.ico)**
  - **Nom, version, description, etc.**

⚠ Idéal de mettre tous les fichiers .py et .bat dans le même dossier

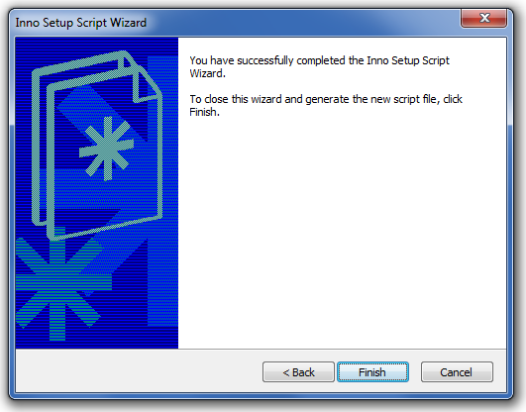
Distribuez votre logiciel sous Windows

http://www.jrsoftware.org/isdl.php

1. Configurez votre installeur en quelques fenêtres très intuitives



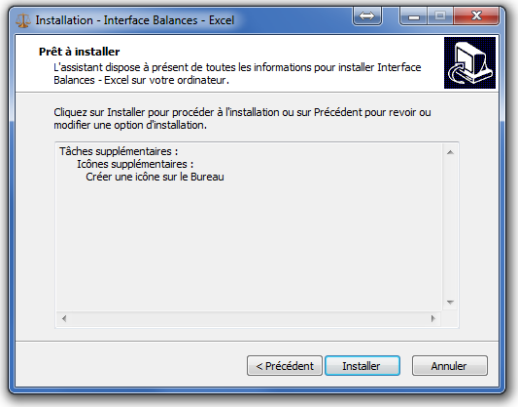
...



- ✓ Nom de l'application
- ✓ Dossier d'installation
- ✓ Langue
- ✓ Nom de l'installeur
- ✓ Icône
- ✓ Etc.

2. Distribuez le fichier d'installation généré par INNO SETUP

Installeur_Interface Balances - Excel.exe



Pour en savoir plus sur Inno Setup : http://www.commentcamarche.net/faq/33511-creer-un-setup-d-installation-inno-setup-compiler

05

Applications réalisées

LAMI Studio

Mesures climatiques globales à l'échelle du stade

Serveur FTP

Thread envoi données

LAMI Studio

En acquisition | Bordeaux - 17 stations | MAJ Config FTP | Log | Relais reboot PC

	Adresse	Station	N°	Active	Cde Poll	Status OK % (-777,-888,-999/total)	Derniere acquisition	Tension batterie	Temp	HR	
1	0013a20040d91663	Periph_ADC_16	00	on	:0400000008..	100.0% (0,0,0/160)	2016-10-06 15:06:32	13.86	19.29	47.43	1920.33,2003.82,1346.74,1450.47,106
2	0013a20040c1606e	Periph_METPAK	01	on	?Q	100.0% (0,0,0/160)	2016-10-06 15:06:32	15.5	19.0	46.3	014,001.76,1012.5,046.3,+019.0,+007
3	0013a20040d91667	Periph_METPAK	02	on	?Q	95.63% (0,5,2/160)	2016-10-06 15:06:32	15.4	18.2	49.7	015,001.06,1011.9,049.7,+018.1,+007
4	0013a20040dbd229	Periph_METPAK	03	on	?Q	100.0% (0,0,0/160)	2016-10-06 15:06:32	15.5	17.7	47.1	028,000.43,1011.9,047.1,+017.7,+006
5	0013a20040d91672	Rampe_GPS	11	on	40d91672,GPS,RA,0,0,0,0	1.25% (0,158,0/160)					-888,-888,-888,-888,-888,-888,-888,-
6	0013a20040d9166c	Rampe	12	on	40d9166c,,RA,0,0,0,0	1.25% (0,158,0/160)					-888,-888,-888,-888,-888,-888,-888,-
7	0013a20040d9166a	Rampe_GPS	21	on	40d9166a,GPS,RA,0,0,0,0	1.25% (0,158,0/160)					-888,-888,-888,-888,-888,-888,-888,-
8	0013a20040d9166b	Rampe	22	on	40d9166b,,RA,0,0,0,0	1.25% (0,158,0/160)					-888,-888,-888,-888,-888,-888,-888,-
9	0013a20040e4f829	Rampe_GPS	31	on	40e4f829,GPS,RA,0,0,0,0	1.25% (0,158,0/160)					-888,-888,-888,-888,-888,-888,-888,-
10	0013a20040d91670	Rampe	32	on	40d91670,,RA,0,0,0,0	1.25% (0,158,0/160)					-888,-888,-888,-888,-888,-888,-888,-

Thread lecture données

2016-10-06 15:06:31 Aucune trame recue 0013a20040e4f829
2016-10-06 15:06:31 Aucune trame recue 0013a20040d91670
2016-10-06 15:06:31 Aucune trame recue 0013a20040d9166d
2016-10-06 15:06:31 Aucune trame recue 0013a20040d91674
2016-10-06 15:06:31 Aucune trame recue 0013a20040d91673
2016-10-06 15:06:31 Aucune trame recue 0013a20040d9166f
2016-10-06 15:06:32 Aucune trame recue 0013a20040d91669
2016-10-06 15:06:32 Aucune trame recue 0013a20040d91675
2016-10-06 15:06:33 2016-10-06-15.csv...2016-10-06.csv
2016-10-06 15:06:33 MAJ affichage IHM Ok

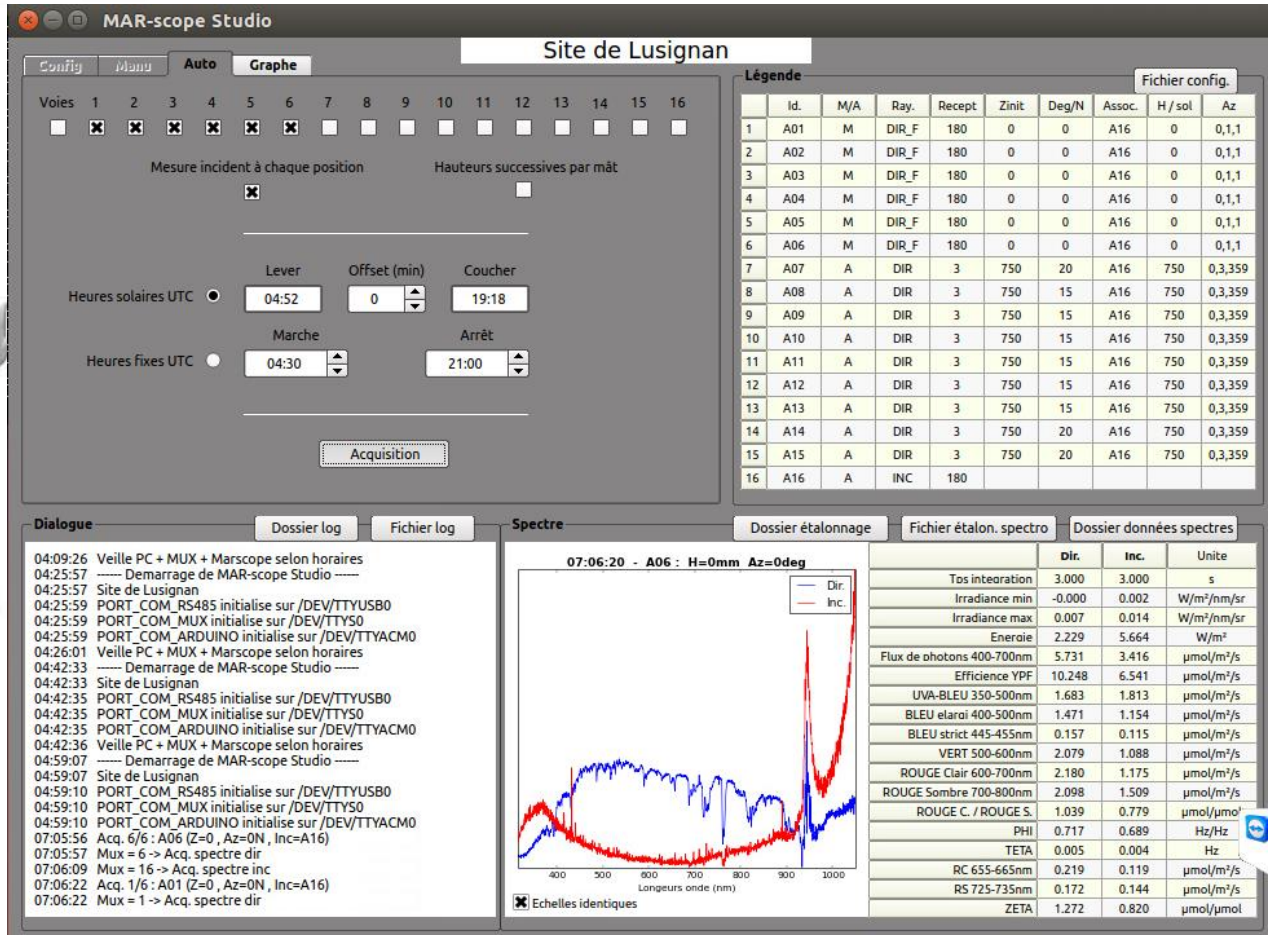
Thread log

Mesures climatiques locales au niveau de la pelouse

Etats et positions GPS des rampes lumineuses

MARSCOPE Studio

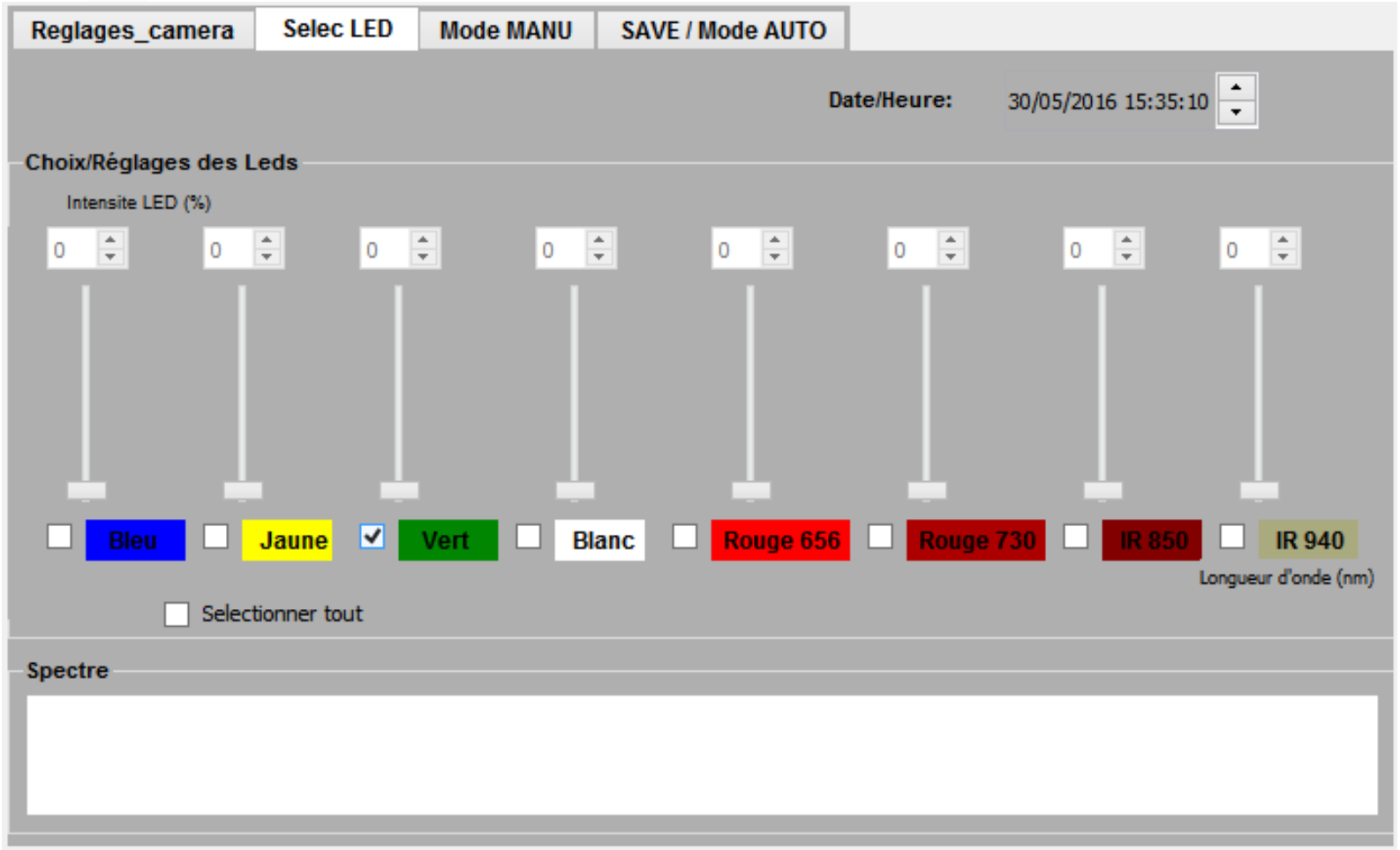
- **Contexte :** Projet Inter-unités ELPhe* département EA
- **But :** Piloter un dispositif d'acquisition automatisé des spectres de rayonnement au sein d'un couvert végétal
- **Particularité :** S'exécute sur ubuntu



*** ELPHe** : Environnement Lumineux et PHotomorphogenèse dans les Associations Végétales

Banc d'imagerie

- **But :** Piloter un banc d'imagerie multipspectral à Leds
- **Particularité :** S'exécute sur RaspberryPi




MERCI !!!