Développement d’IHM sur PC en programmation scientifique

(Python, MATLAB, GNU Octave, Scilab)

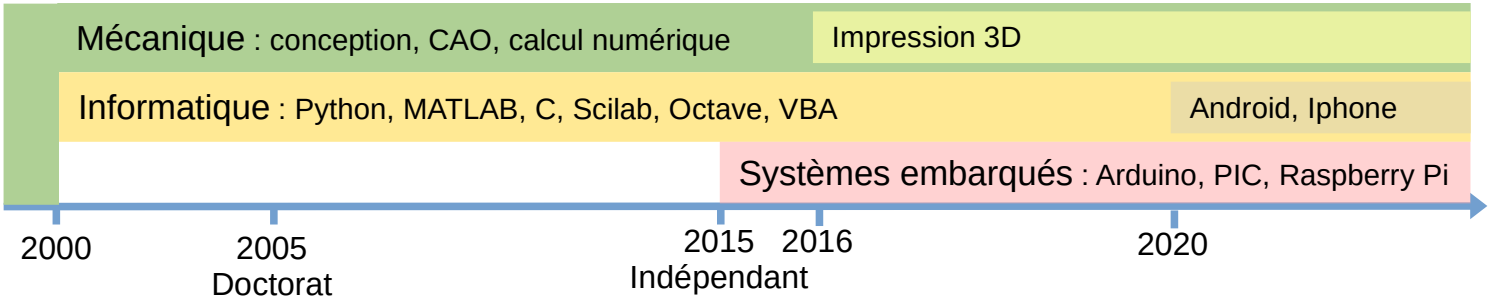
Jérôme Briot

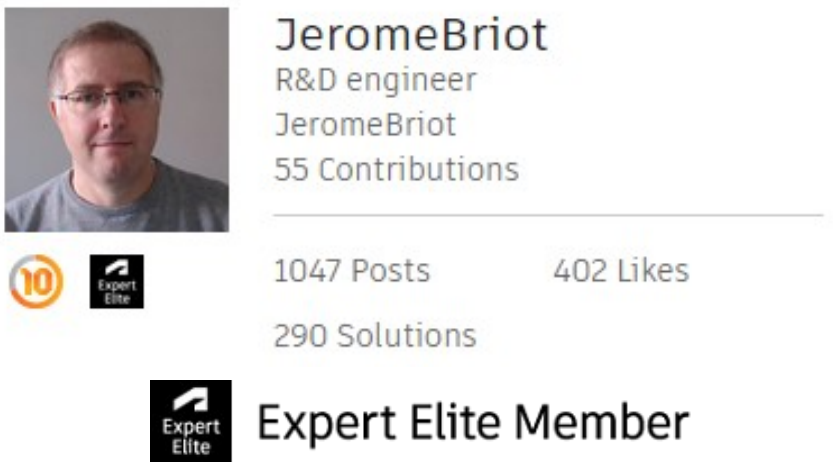
Ingénieur indépendant
jbtechlab@gmail.com
Référencé CNRS et IRD

Exemples disponibles sur :

https://github.com/JeromeBriot/ihm-pc-cnrs-06-2022

Mon parcours

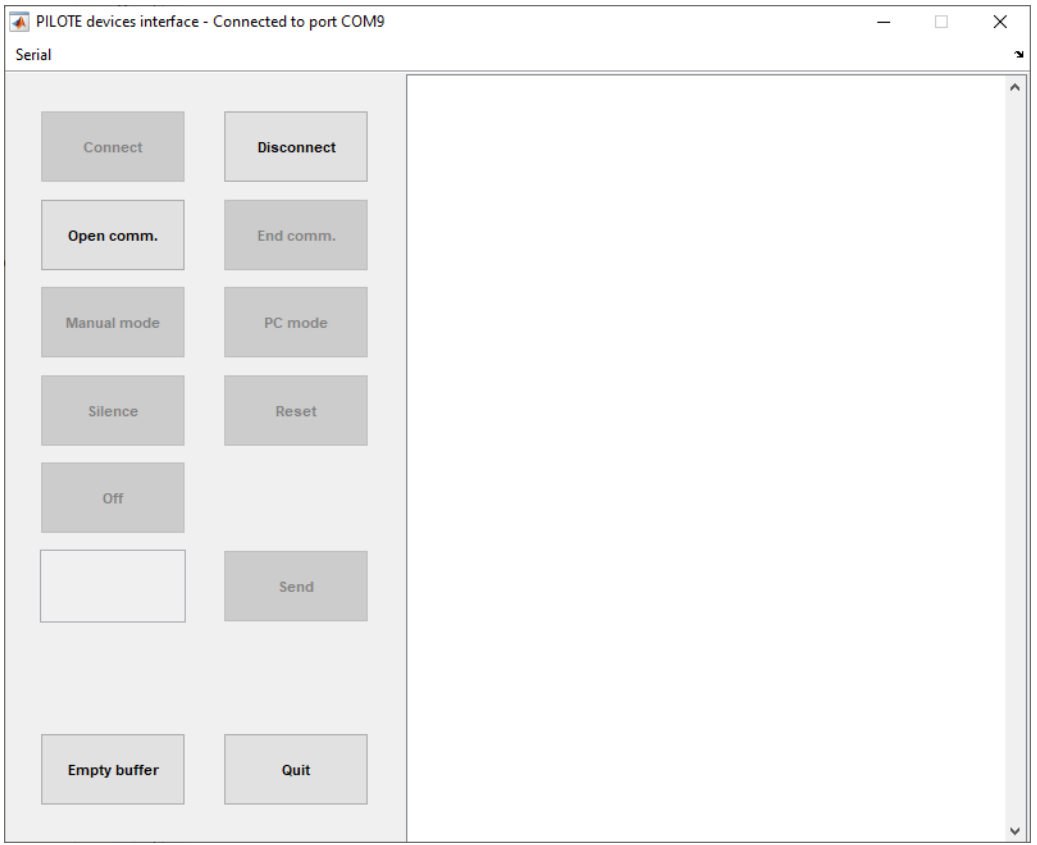




Quelques réalisations

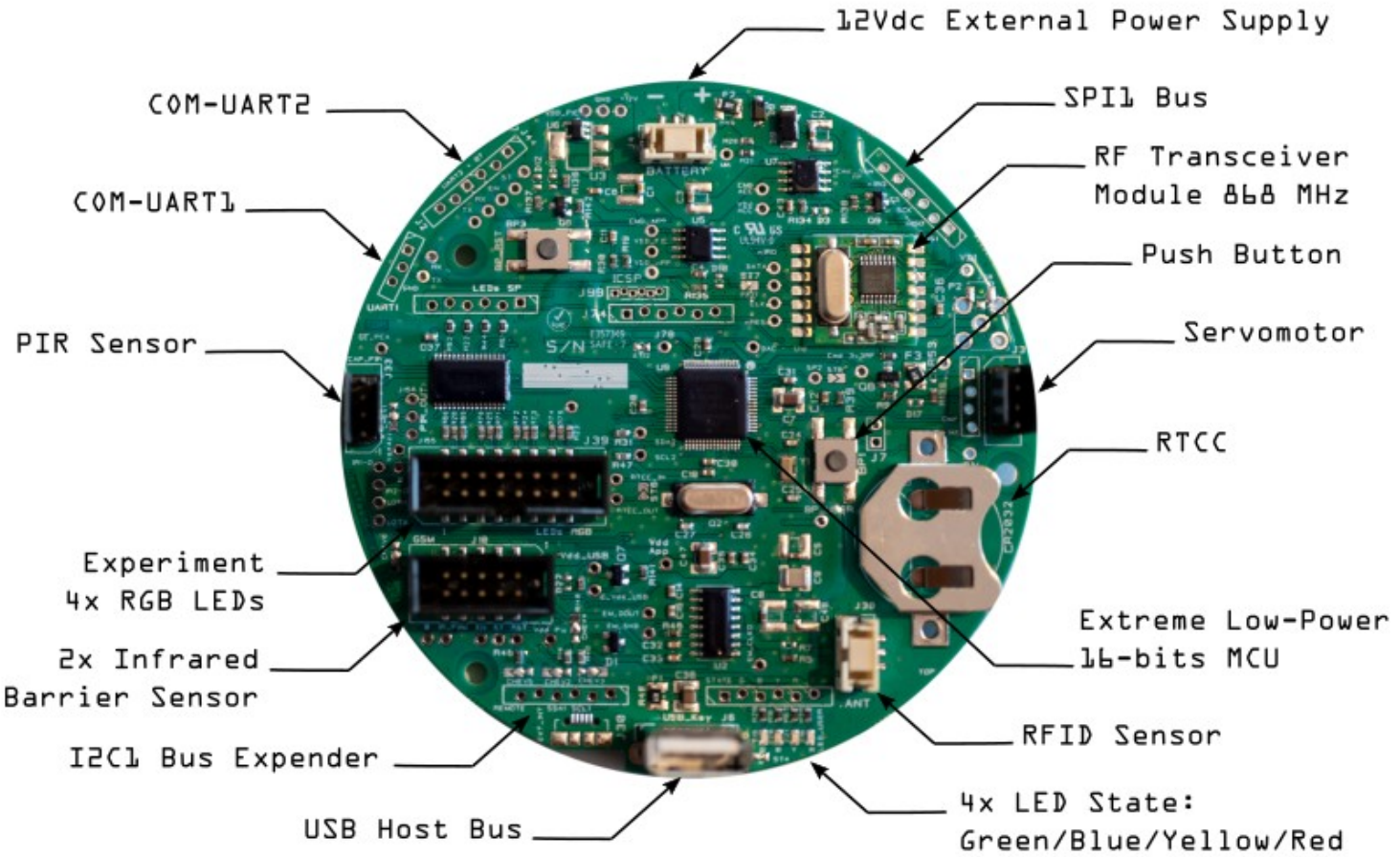
Pousse seringue

Laboratoire de Biomécanique, Toulouse



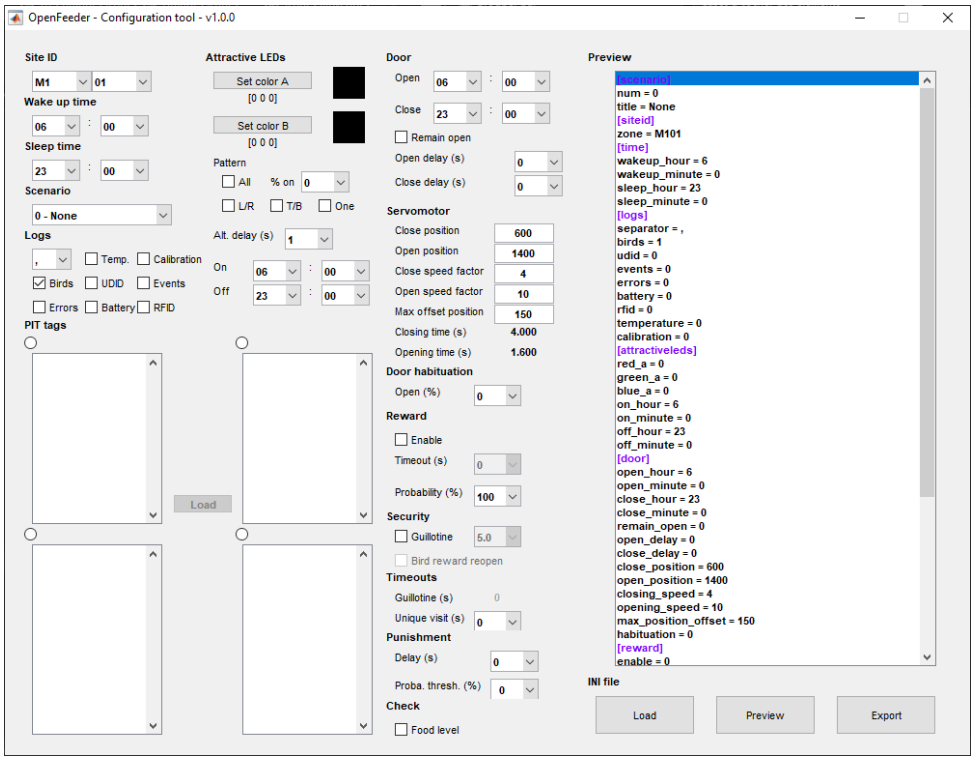
Openfeeder

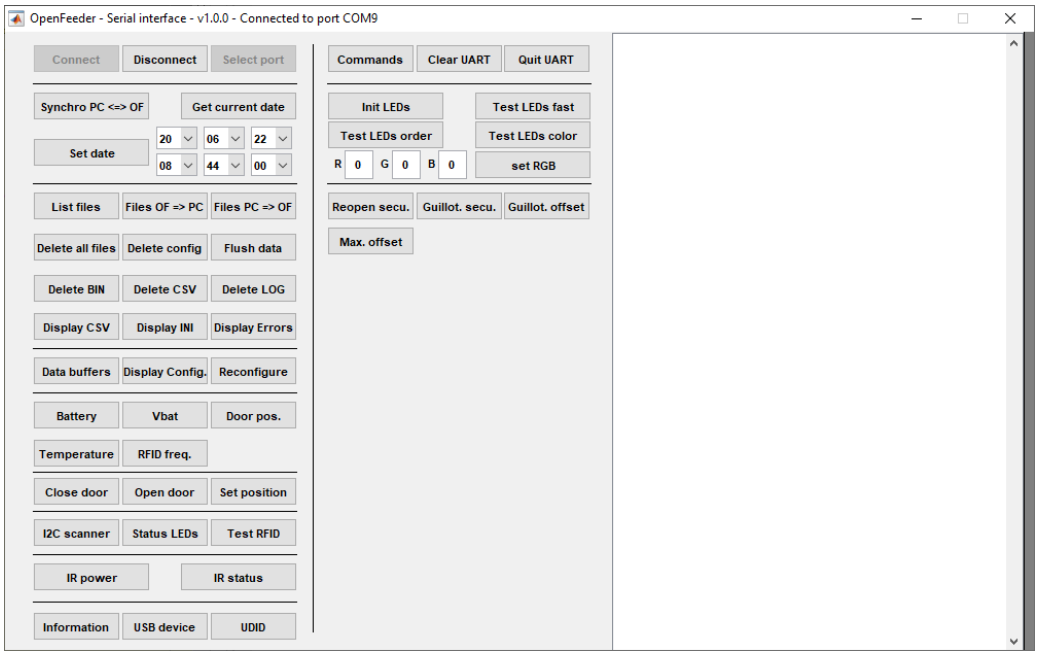
SETE, CNRS, Moulis



Openfeeder

SETE, CNRS, Moulis



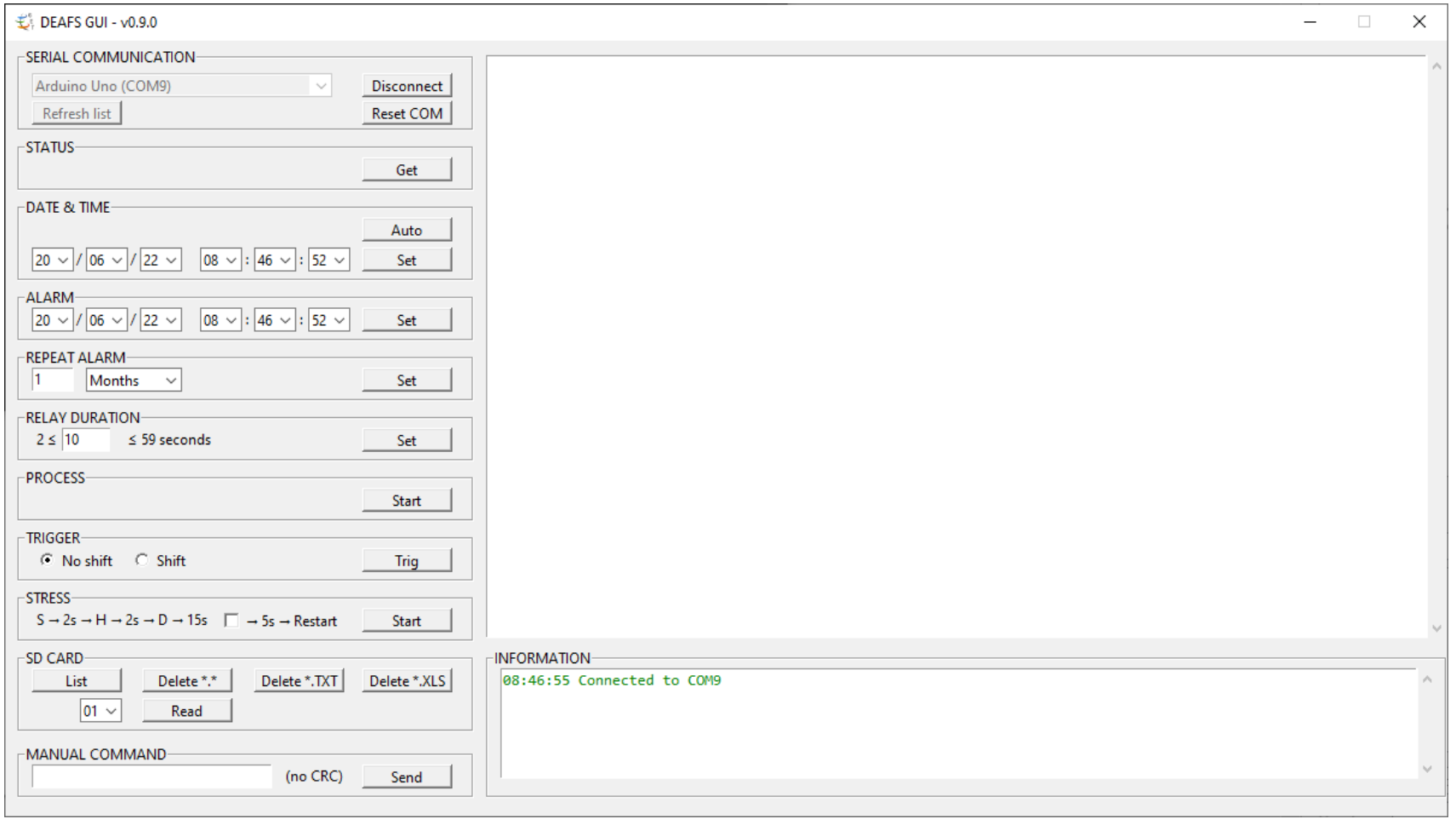


Openfeeder

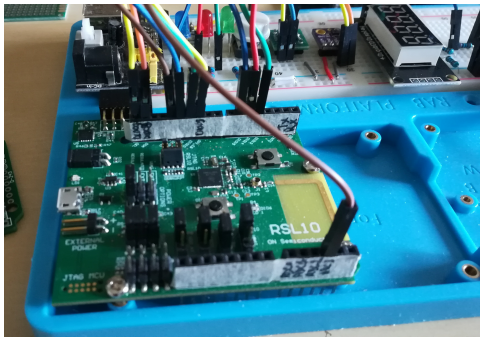
SETE, CNRS, Moulis

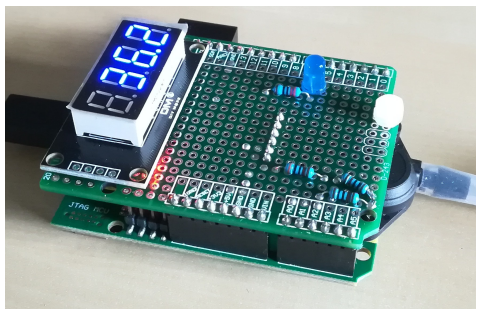
DEAFS

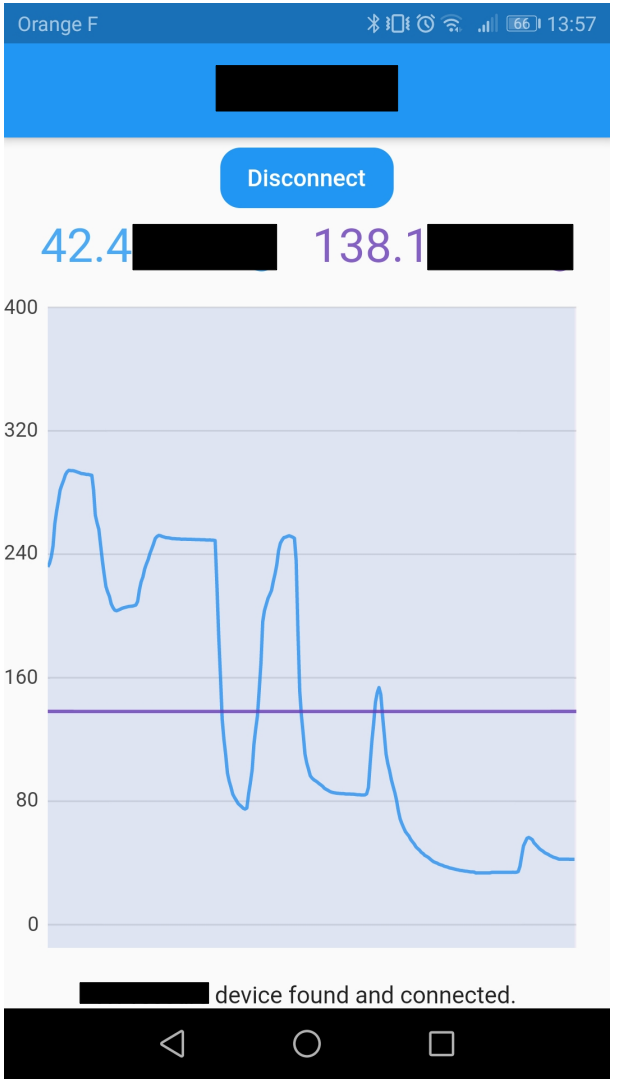
GET, CNRS, Toulouse



Mesure de pression et transmission BT





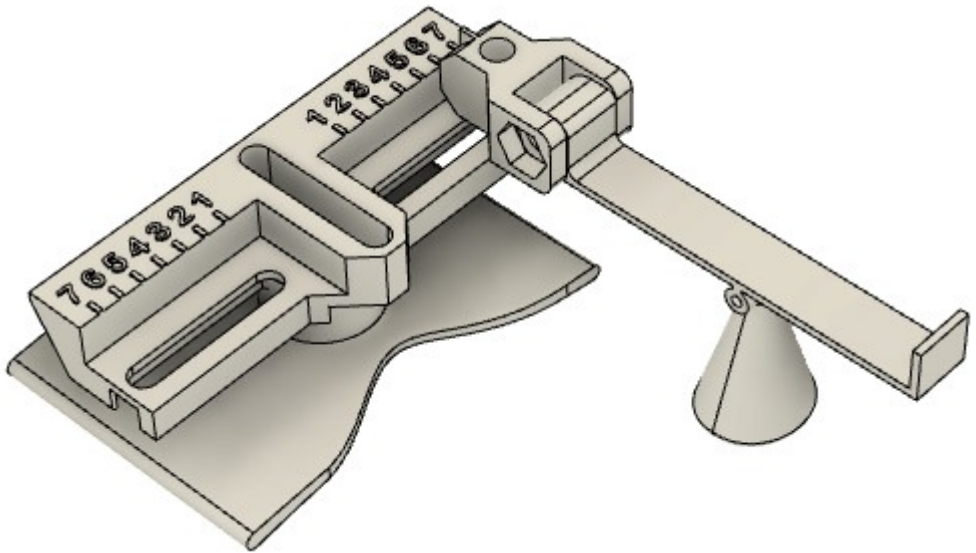


Boîtier moteur

Laboratoire de Biomécanique & ENVT, Toulouse

Outil de mesure

Laboratoire de Biomécanique & CHU, Toulouse



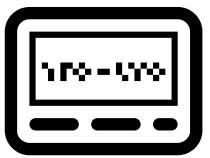
Développement d’IHM sur PC en programmation scientifique

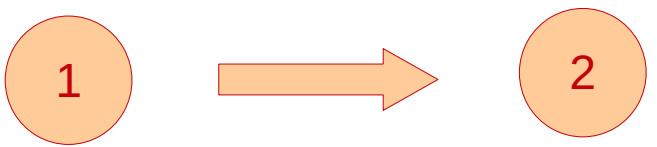
(Python, MATLAB, GNU Octave, Scilab)

Réflexions autour des IHM

Utilisateurs

UX Expérience utilisateur	UI Interface utilisateur
Résultat de l’interaction	Moyens de l’interaction





Environnement

Public cible ?

Utilisateurs adultes, scientifiques, experts

Matériel cible ?

PC

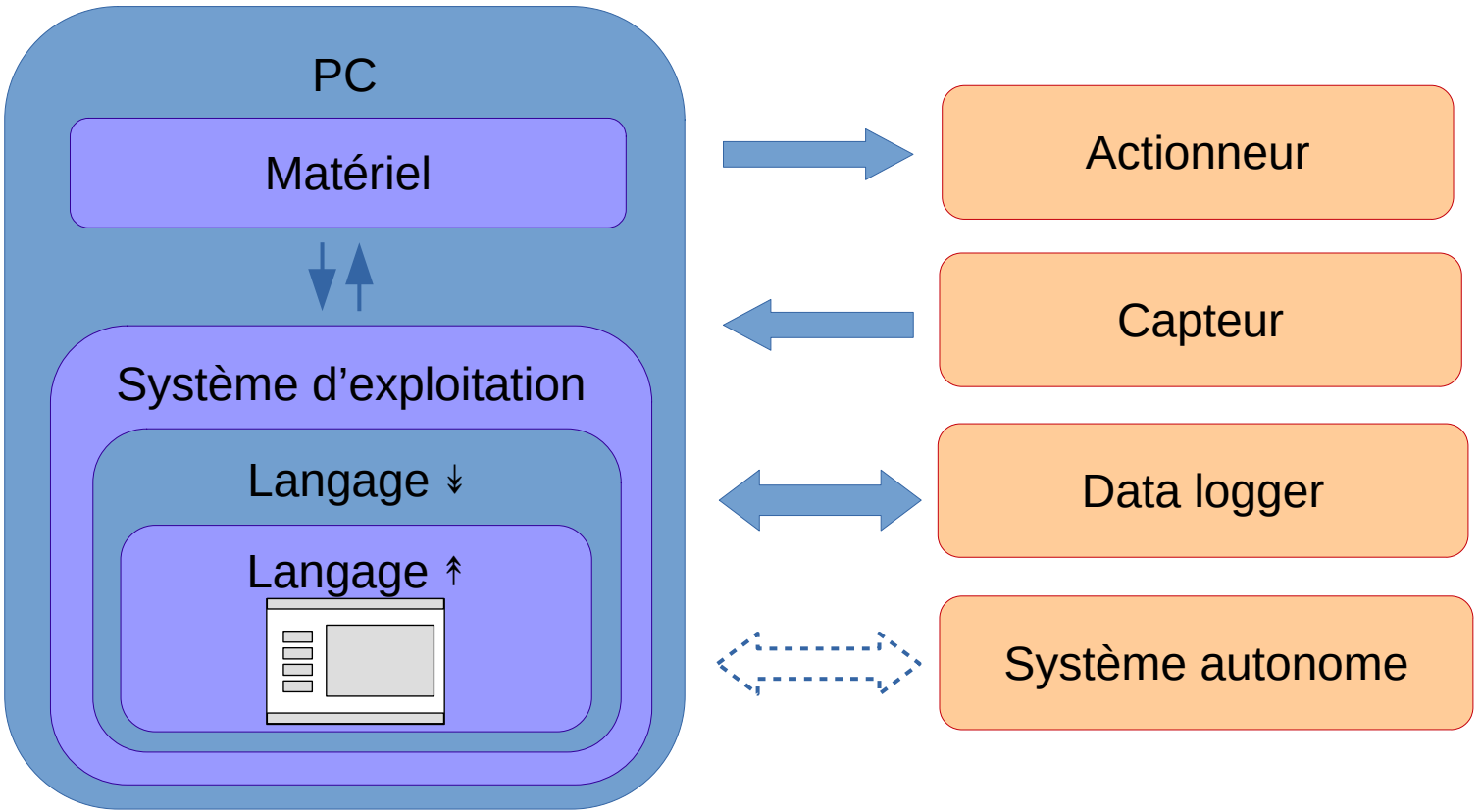
Usage ?

Expérimentation scientifique

Finalité ?

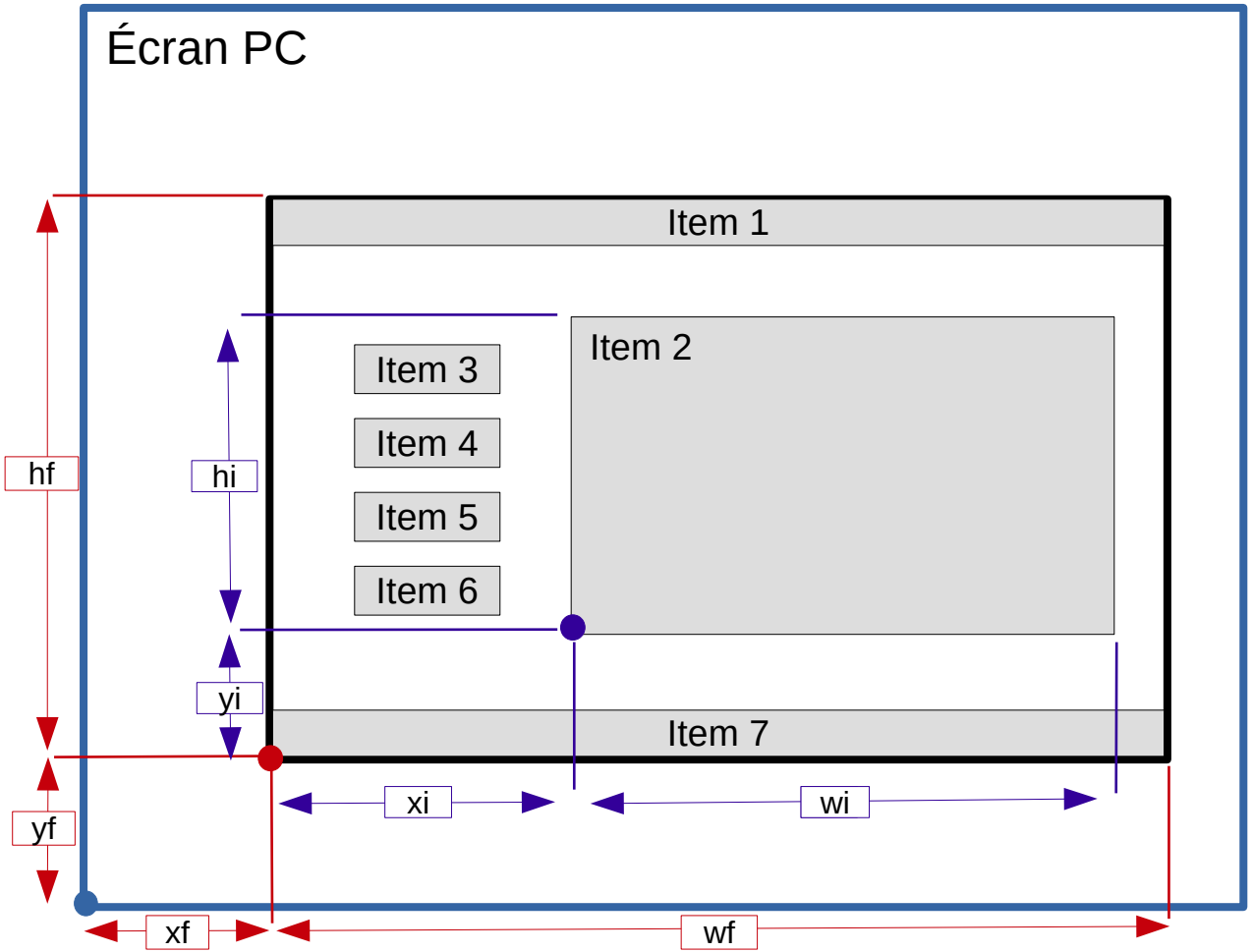
Publication scientifique et diffusion libre (open source)

Interfaçage PC - Dispositif

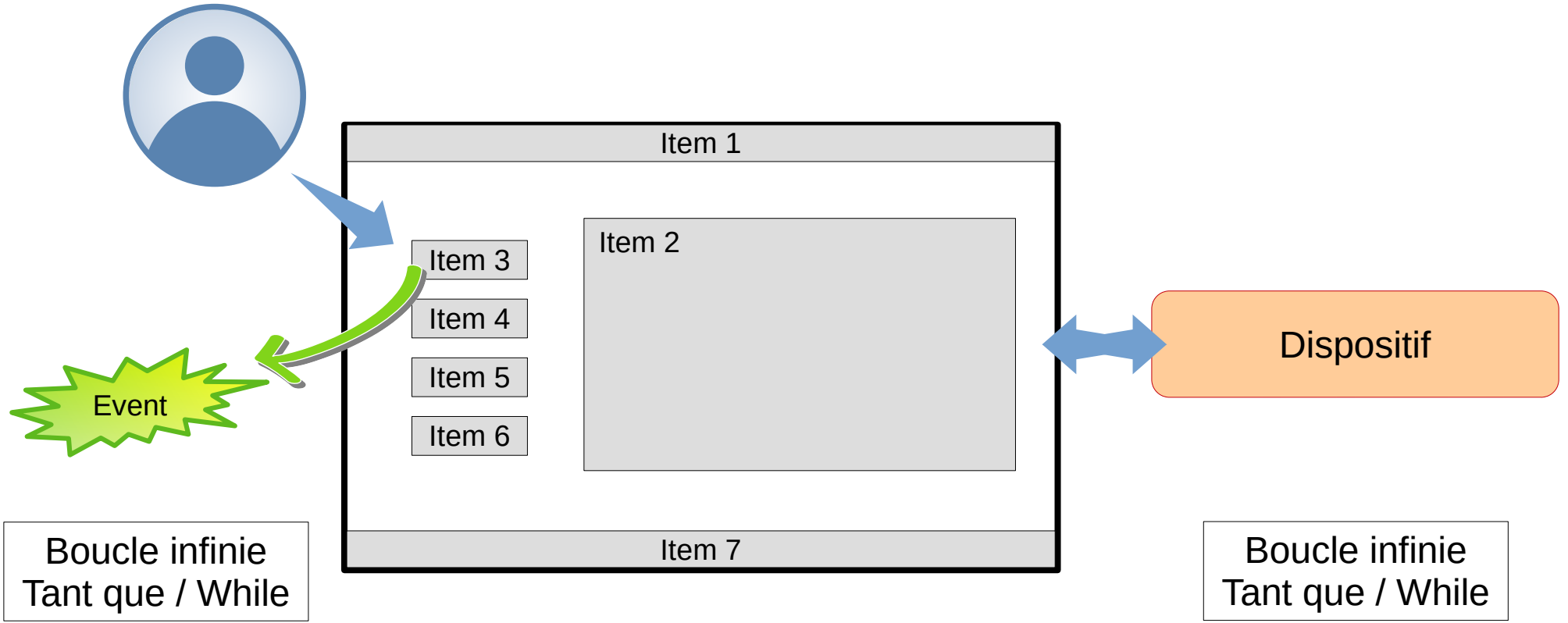


Étapes de la conception d’une IHM PC

Étape 1 : positionnement



Étape 2 : événements



Mise en application

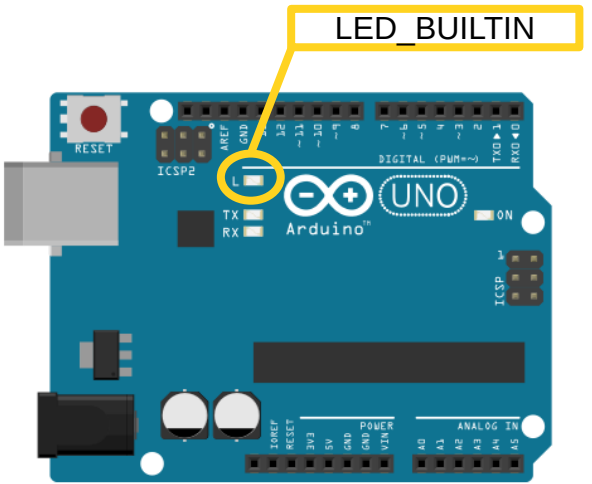
Matériel

Windows 10

macOS Monterey

Xubuntu 22.04

UART

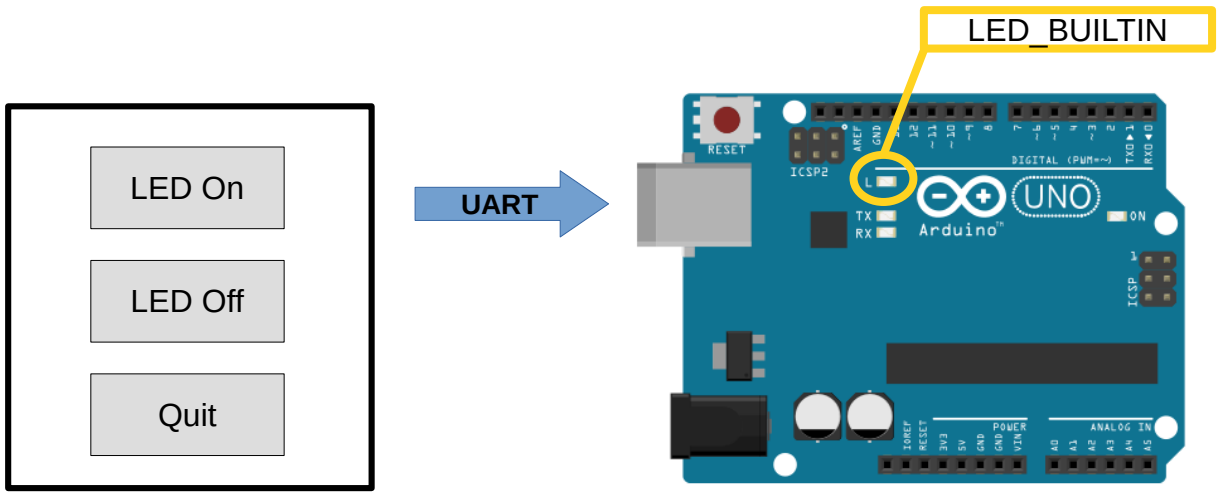


Langages de programmation

	Python 3.10	MATLAB R2022a	GNU Octave 7.1.0(1)	Scilab 6.1.1
Bibliothèque graphique	Tkinter	Intégrée (Java)	Intégrée (Qt)	Intégrée (Java)
Bibliothèque liaison série	pySerial 3.5	Intégrée(2)	Instrument Control Package 0.7.1	Serial Communication Toolbox 0.5(3)

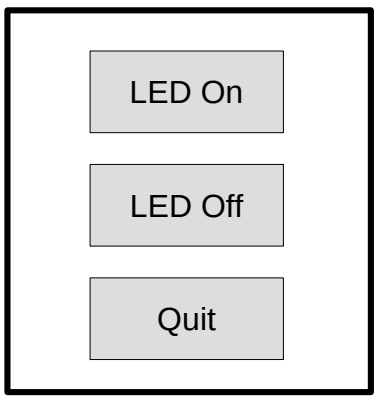
1 : GNU Octave version 6.4.0 utilisée sur Linux
2 : *serialport* depuis MATLAB R2019b ; *serial* pour les versions antérieures
3 : Serial Comunication Toolbox non disponible sur macOS

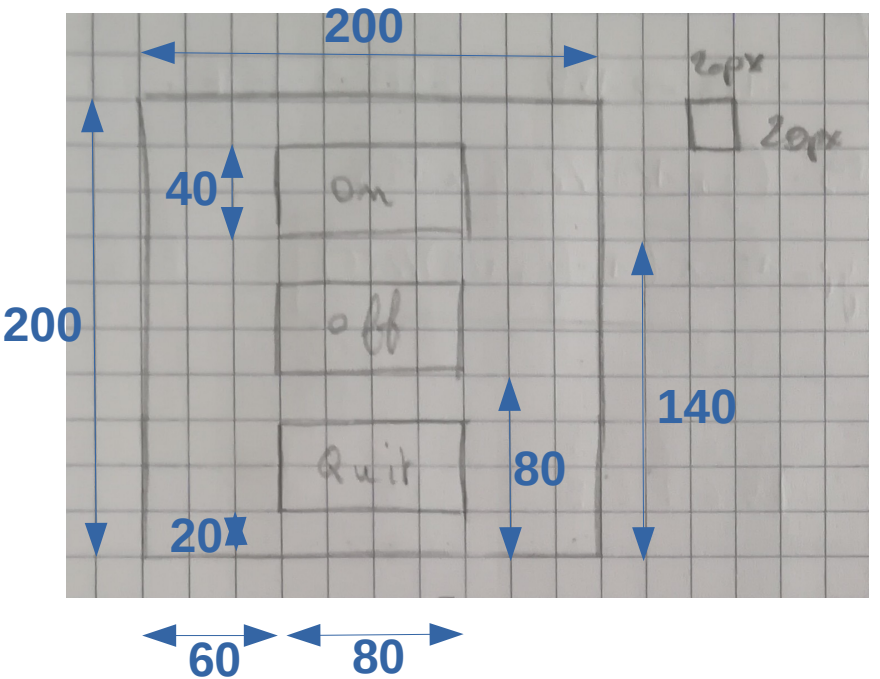
Commande LED Arduino via communication série



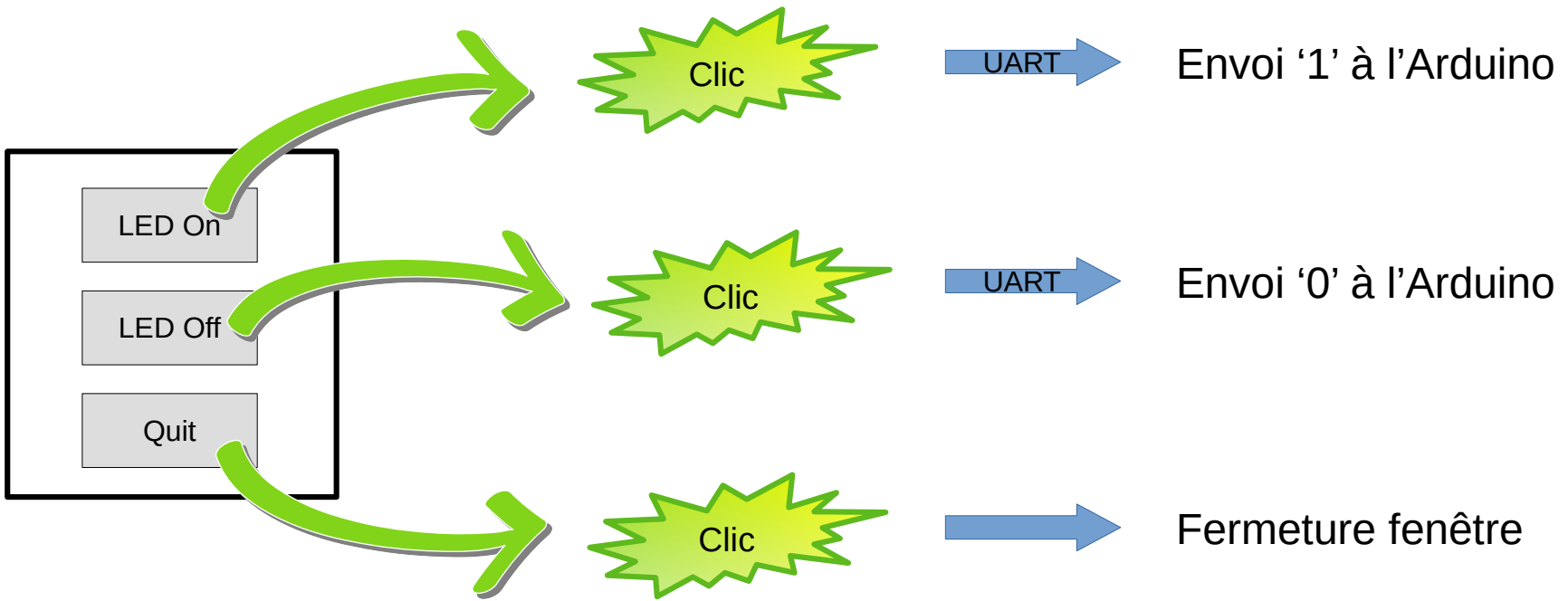

```
void setup() {

  Serial.begin(9600);

  pinMode(LED_BUILTIN, OUTPUT);

}

void loop() {

  int serialReceivedValue;

  if (Serial.available() > 0) {

    serialReceivedValue = Serial.read();

    if ('1' == serialReceivedValue) {
      digitalWrite(LED_BUILTIN, HIGH);
    }
    else if ('0' == serialReceivedValue) {
      digitalWrite(LED_BUILTIN, LOW);
    }

  }

}
```

Étape 1 : positionnement





Étape 2 : événements



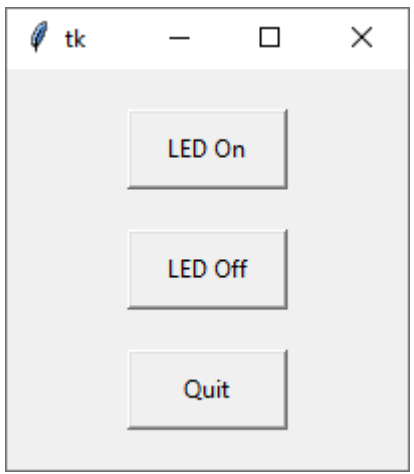
```python
import serial
import time

import tkinter as tk

ser = serial.Serial(port='COM9', baudrate=9600)

time.sleep(3) # Bootloader Arduino

def turnLedOn():

    ser.write(b'1')

def turnLedOff():

    ser.write(b'0')

def quitApp():

    ser.close()
    root.quit()

root = tk.Tk()

root.geometry('200x200')

buttonQuit = tk.Button(root, text='Quit',
                       command=quitApp)
buttonQuit.place(x=60, y=140,
                 width=80, height=40)

buttonOff = tk.Button(root, text='LED Off',
                      command=turnLedOff)
buttonOff.place(x=60, y=80,
                width=80, height=40)

buttonOn = tk.Button(root, text='LED On',
                     command=turnLedOn)
buttonOn.place(x=60, y=20,
               width=80, height=40)

root.mainloop()
```

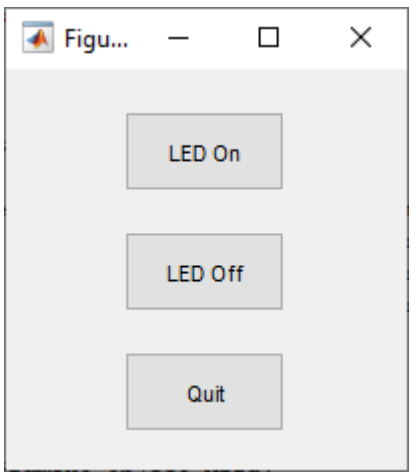
```
function led_toggle

ser = serialport('COM9', 9600);

pause(3) % Bootloader Arduino

figure('units', 'pixels', ...
    'position', [0 0 200 200])

uicontrol('style', 'pushbutton', ...
    'units', 'pixels', ...
    'position', [60 20 80 40], ...
    'string', 'Quit', ...
    'callback', @quitApp)

uicontrol('style', 'pushbutton', ...
    'units', 'pixels', ...
    'position', [60 80 80 40], ...
    'string', 'LED Off', ...
    'callback', @turnLedOff)

uicontrol('style', 'pushbutton', ...
    'units', 'pixels', ...
    'position', [60 140 80 40], ...
    'string', 'LED On', ...
    'callback', @turnLedOn)

    function turnLedOn(obj, event)
        write(ser, '1', 'char');
    end

    function turnLedOff(obj, event)
        write(ser, '0', 'char');
    end

    function quitApp(obj, event)
        delete(ser)
        close(gcbf)
    end

end
```

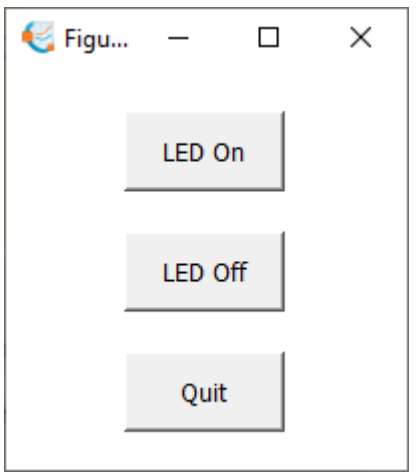
```
function led_toggle

pkg load instrument-control

ser = serialport('COM9', 9600);

pause(3); % Bootloader Arduino

figure('units', 'pixels',
'position', [0 0 200 200])

uicontrol('style', 'pushbutton', ...
'units', 'pixels', ...
'position', [60 20 80 40], ...
'string', 'Quit', ...
'callback', @quitApp)

uicontrol('style', 'pushbutton', ...
'units', 'pixels', ...
'position', [60 80 80 40], ...
'string', 'LED Off', ...
'callback', @turnLedOff)

uicontrol('style', 'pushbutton', ...
'units', 'pixels', ...
'position', [60 140 80 40], ...
'string', 'LED On', ...
'callback', @turnLedOn)

function turnLedOn(obj, event)
    write(ser, '1');
endfunction

function turnLedOff(obj, event)
    write(ser, '0');
endfunction

function quitApp(obj, event)
  clear ser
  close(gcbf)
endfunction

endfunction
```

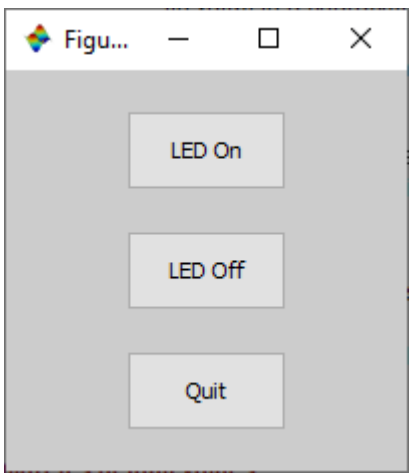
```
ser = openserial('COM9', '9600,n,8,1')

sleep(3, "s") // Bootloader Arduino

figure('position', [0, 0, 200, 200])

uicontrol('style', 'pushbutton', ...
'units', 'pixels', ...
'position', [60 20 80 40], ...
'string', 'Quit', ...
'callback', 'quitApp')

uicontrol('style', 'pushbutton', ...
'units', 'pixels', ...
'position', [60 80 80 40], ...
'string', 'LED Off', ...
'callback', 'turnLedOff')

uicontrol('style', 'pushbutton', ...
'units', 'pixels', ...
'position', [60 140 80 40], ...
'string', 'LED On', ...
'callback', 'turnLedOn')

function turnLedOn()
    writeserial(ser, '1')
endfunction

function turnLedOff()
    writeserial(ser, '0')
endfunction

function quitApp()
    closeserial(ser)
    close()
endfunction
```

Windows 10

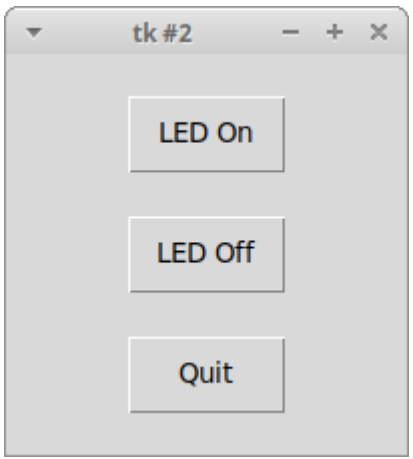

tk
LED On
LED Off
Quit

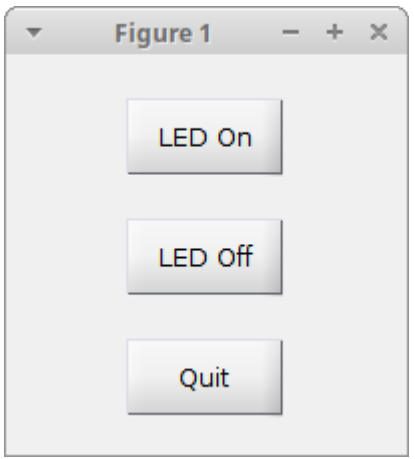

Figu...
LED On
LED Off
Quit

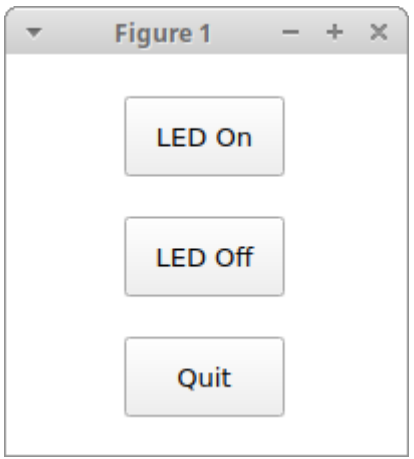

Figu...
LED On
LED Off
Quit

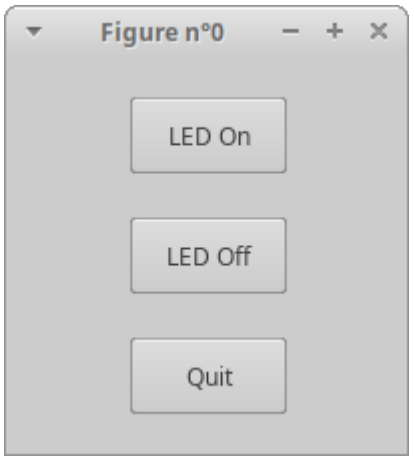

Figu...
LED On
LED Off
Quit

Xubuntu 22.04

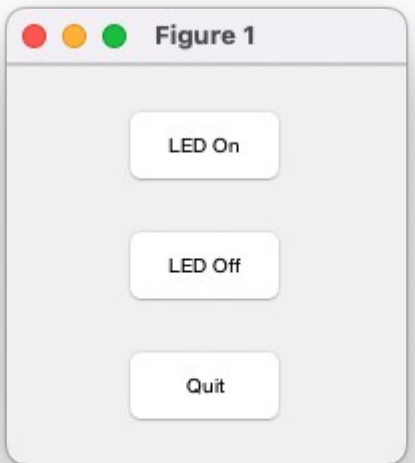


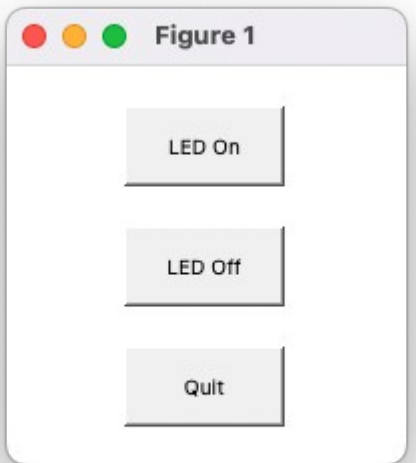






macOS Monterey





Merci