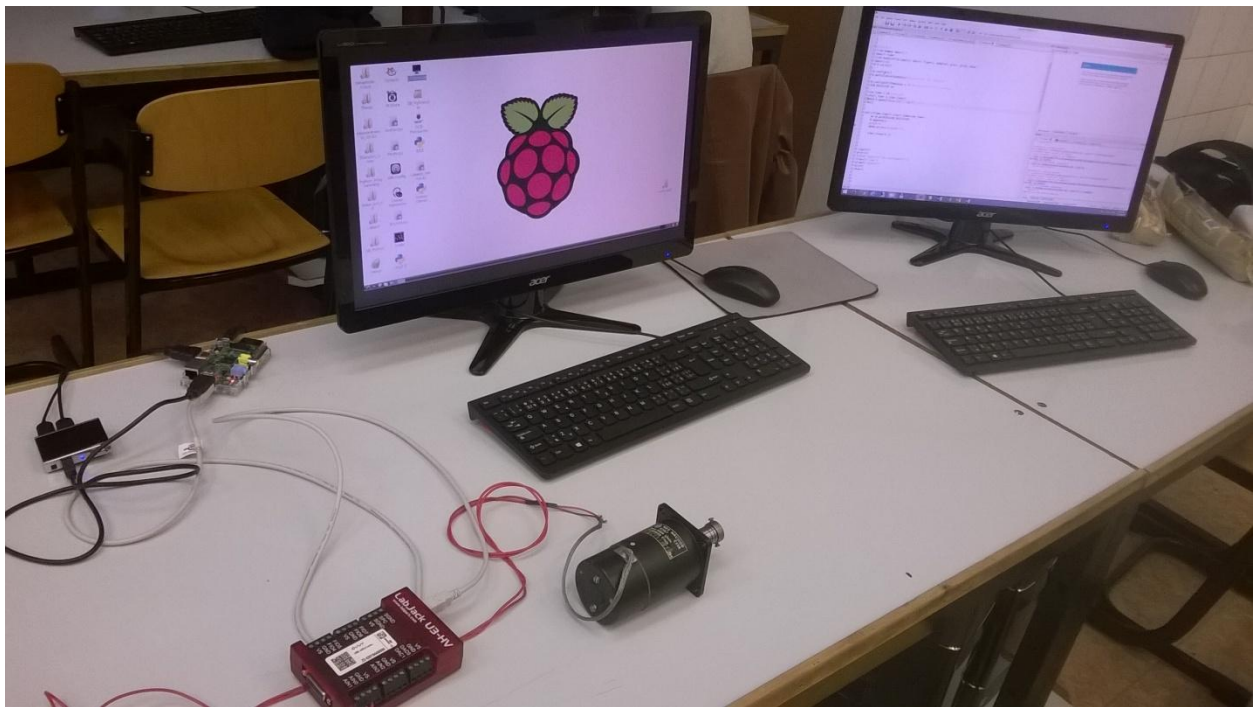


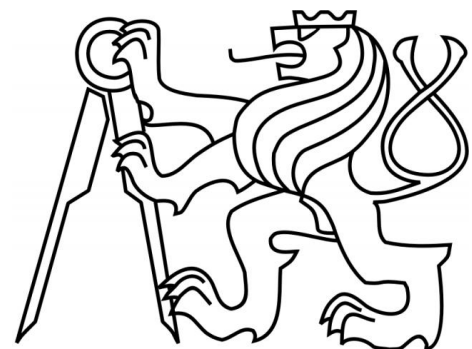
Python project

Python for scientific computation and control

How to manage a speed sensor with a Labjack U3 HV



Pierre COCAGNE
Maël Guillerm



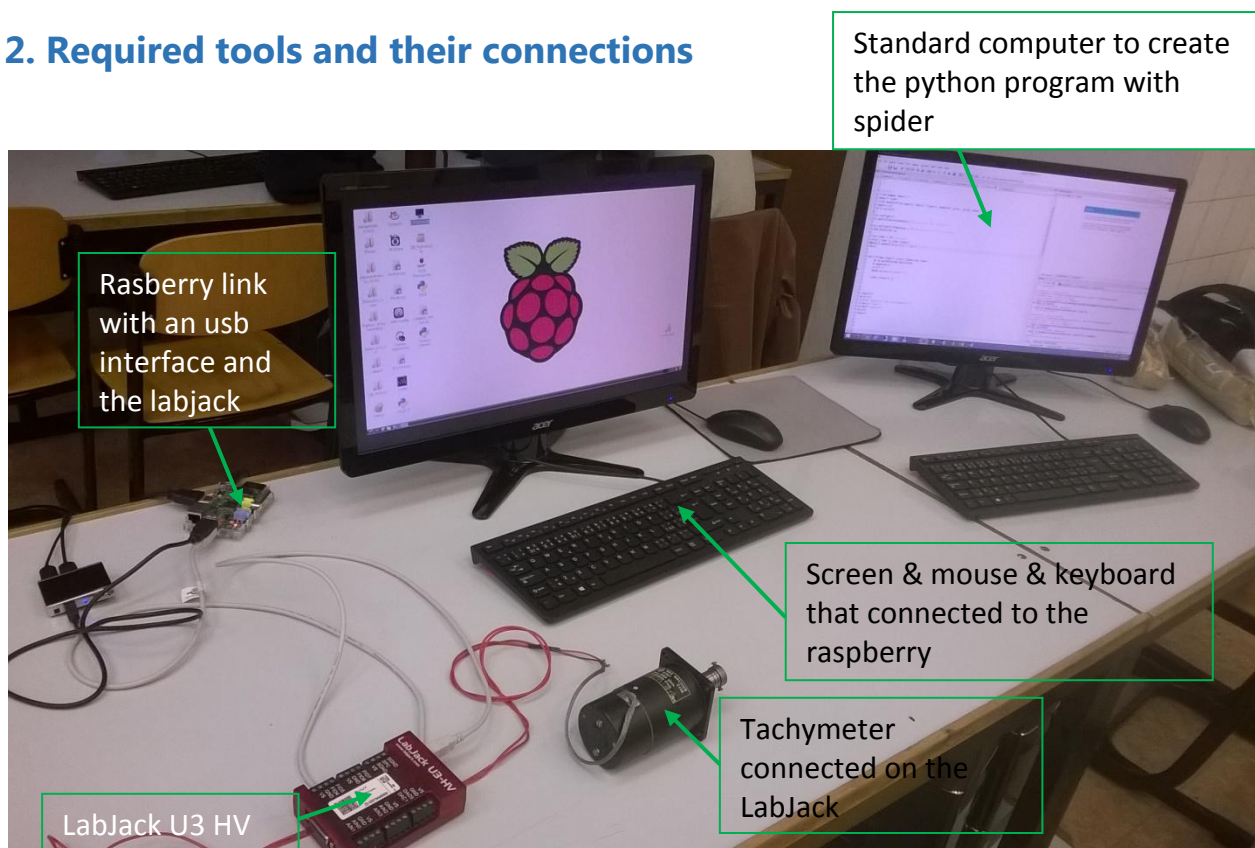
Mission of the project and required tools

1. Introduction

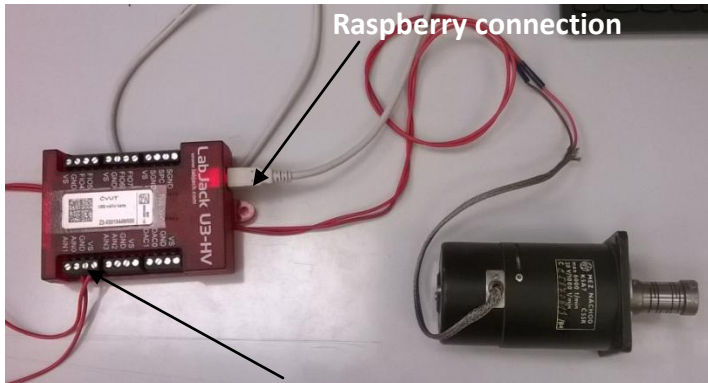
The goal of our project is to use a speed sensor (tachymeter) and create a program that shows us what the sensor measures. We will link the sensor with the computer by using a Labjack U3 HV. Then the program will be running by a raspberry computer.

In a first time, we have to make a program that will be able to read the value of the sensor in real time. In a second time, this program can be optimized by 2 alternatives ways. The first is to put the total graph at the end of the simulation. The second optimization consists in exporting values that had been measured in an external folder and reading it with another program whenever we want.

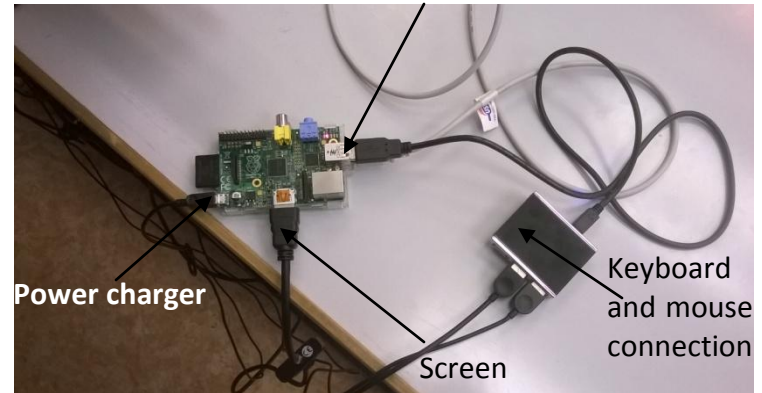
2. Required tools and their connections



First of all, the tachymeter is connected to the Labjack by an analog input and the ground. The Labjack is itself connected to the raspberry by a USB port. A USB external port is added to connect the screen and the mouse with the raspberry. A HDMI/VGA converter makes the link with the PC and the screen. The operator has to turn himself the tachymeter to generate an input signal.



Sensor analog input
Connection between the Labjack and the sensor



Connection about the Raspberry

3. Establishment of the program to have value of the tachymeter in real time

This code uses the LabJack to record one sensor at a regular cadence. It is able to read the value of the sensor in real time.

```
#library
import time
from matplotlib.pyplot import figure, subplot, plot, grid, show
import u3
d = u3.U3()

d.configU3()
d.getCalibrationData() #calibrate the labjack

d.configIO(FIOAnalog = 3) #configure F13 as analog
AIN_REGISTER = 0
FIO0_STATE_REGISTER = 6000

sim_time = 10 #seconds
start_time = time.time()
X=[]

while(time.time()-start_time<sim_time):
    x= d.getAIN(AIN_REGISTER)
    X.append(x)
    print x
    time.sleep(0.1)
```

At the beginning we call all the libraries that we need to work with LabJack (example; u3 and the more basics matplotlib). After we create the spacework of Python we can start with the configuration of the LabJack

Configuration of LabJack:

We want to configure with on analog input:

- **d = u3.U3()** : initialize the interface; assumes a single U3 is plugged in to a USB port
- **d.configU3()** : set default configuration
- **d.configIO(FIOAnalog = 3)** : configure the Flexible input number 3 as an analog input.
- **AIN_REGESTER = 0** : set the input to the 0 default value.
- **FIO_STATE_REGISTER = 6000** : open the port of flexible input output.

In the LabJack the addresses 6000-6019 will set the digital output states, where addresses 6000-6007 correspond to FIO0-FIO7 .

Functions:

We use a loop “while” to create a time limit at the simulation. The simulation time is 10 seconds.

X=[] : set the variable X as an empty list that we will full with the data.

x = d.getAIN(AIN_REGISTER) : is the function that reads the input and registers this value in the variable x

X.append(x) : The method append() appends a passed object into the existing list. It adds each different values of x in the list (and the time that this value was red).

Print x : this command returns the value x to the console screen for the operator in real time. This x value is limited to the supply of the LabJack and his range is [-10 V to 10 V]. The sign gives us the rotation sense (positive is for clock sense and negative is for counter clock sense).

The sleep time is the time gap between two measures. It is set to 100 ms.

To conclude, this program returns the current value on the console screen in real time of the tachymeter. The Tachymeter could work with a voltage of 60V but the LabJack can't work at this range. For an optimization of the simulation we could add an electronic circuit that full filled the range of the sensor.

4. Alternative 1 : Measured curve just after the experiment

This program is an optimization of the program 1. The goal is still print the value in real time of what movements are applied on the tachymeter. Furthermore, at this end of the simulation, this program will trace the curve at the end of the simulation time.

```
#library
from numpy import *
import time
from matplotlib.pyplot import figure, subplot, plot, grid, show
import u3
d = u3.U3()

d.configU3()
d.getCalibrationData() #calibrate the labjack

d.configIO(FIOAnalog = 3) #configure FI3 as analog
AIN_REGISTER = 0
FIO0_STATE_REGISTER = 6000

sim_time = 10 #seconds
start_time = time.time()
X=[]

while(time.time()-start_time<sim_time):
    x= d.getAIN(AIN_REGISTER)
    X.append(x)
    print x
    time.sleep(0.1)

figure()
plot(X)
title('Speed of the tachimeter')
xlabel('time')
ylabel('Speed')
grid()
show()
```

The beginning of the program is the same as the program 1. The calibration the configuration and the function of print x in real time doesn't change. The modification is after the loop by adding few commands.

Show curves functions:

Figure() : this command opens a tracing window

Plot(X) : this command draws all points of the table X on the windows

title('X'),xlabel('X'),ylabel('X'),grid() : Theses command are the standard to make respectively a command to add a title, a title at the X axis, a title at the Y axis , draw a grid, on the curve windows.

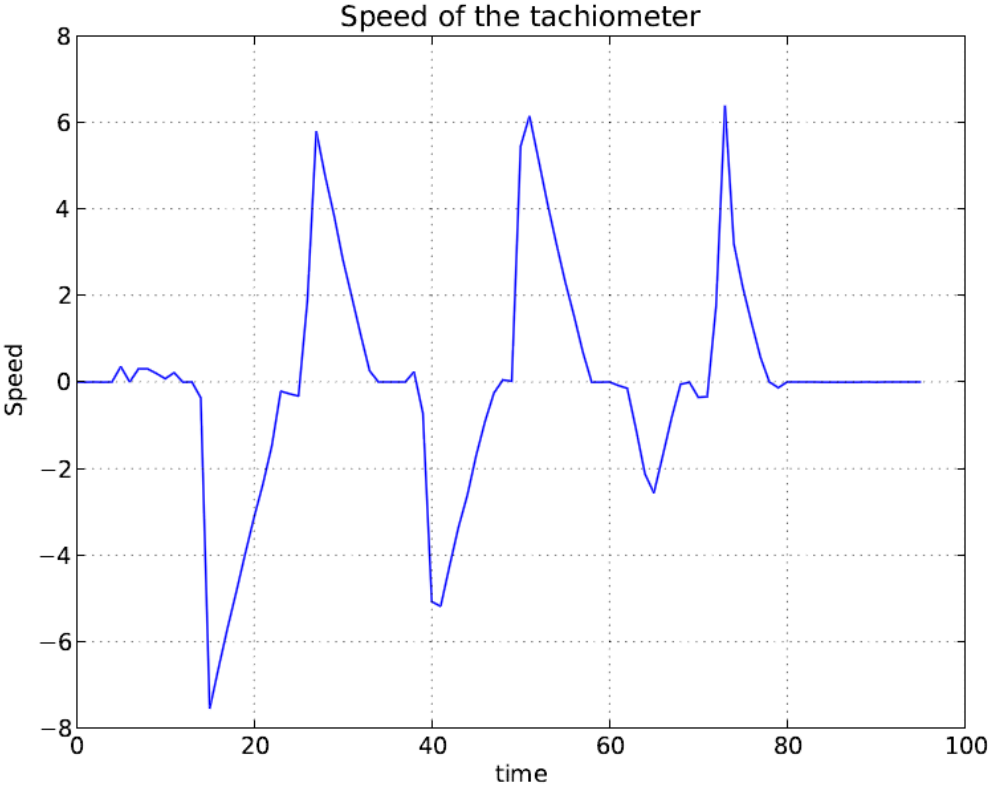
Show() : this command shows the windows to the operator.

4

Python project

Pierre COCAGNE & Mael Guillerm

To conclude, this program returns to the operator the real time value of the tachymeter and after, at the end of the simulation, a curve of what the tachymeter had measured.



5. Alternative 2 : Extract value in a external file and curve tracing program

Here is another alternative program to collect the data. It works in two times. First the program collects the data in an external file. Then, the program reads the external file and prints the curve of the simulation data.

The following code is the first part of the program.

```
#library
import time
import u3
d = u3.U3()

d.configU3()
d.getCalibrationData() #calibrate the labjack

d.configIO(FIOAnalog = 3) #configure FI3 as analog
FIO0_STATE_REGISTER = 6000

sim_time = 10 #seconds
start_time = time.time()
data = open("data.txt", "wb")
x=[]

while(time.time()-start_time<sim_time):
    x= d.getFIOState(FIO0_STATE_REGISTER)
    X.append(x)
    print x
    data.write(str(x)+"\n")
    time.sleep(0.1)

data.close()
```

The calibration stills the same as the program 1.

Creation of the external file

data = open('X') : create an empty external file that is named X to fill in after with the data. This file will be located in the parent folder of the program.

Fill the external file

Data.write(str(x)+"\n") : this command write in the file called data the current value of x and add a line to separate this value to the next value. The str is the format of the variable x which is in our case a string.

At the end of the simulation, all data are saved on the external file. I will explain how read this file.

For read an external file we will import a special library that is called csv. The csv library read text files.

```
from numpy import *
from matplotlib.pyplot import *
import csv

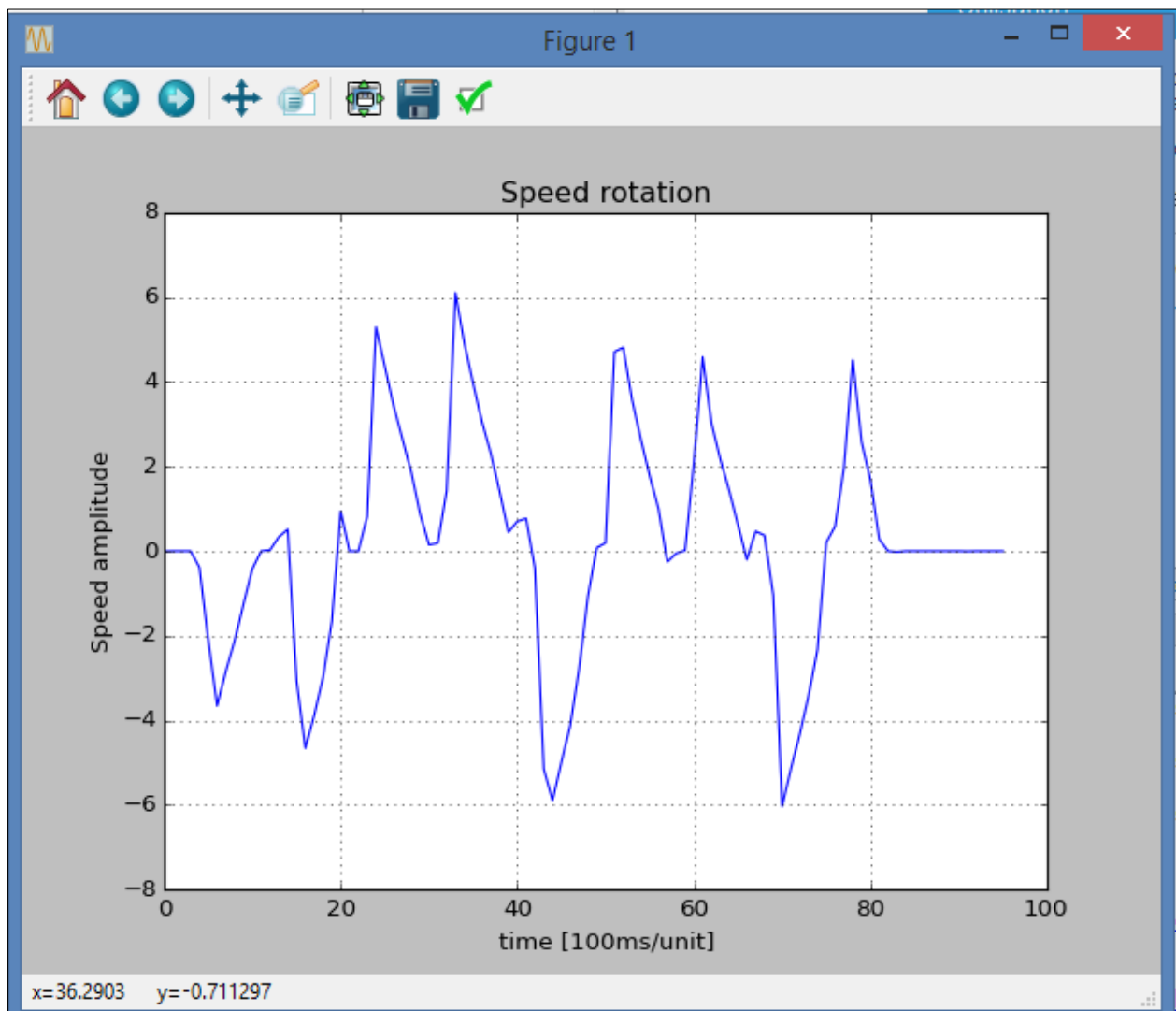
data=loadtxt("datafull1.txt")
figure()
plot(data)
title("Speed rotation" )
xlabel("time [100ms/unit]")
ylabel("Speed amplitude")
grid()
show()
```

data=loadtxt("x.txt") : This command opens the file and loads all the data in the variable data

After that, we use once again the same plotting command as the alternative 1 program to plot the curve.

plot(data) : plot all the list of the data variable on the curve.

We obtain that following result.



To conclude this program allows us to save all data in an external file that is useful to make a data base. Then the second part of the program opens again the external file and traces the data curve. In a scientist point of view, this program is more interesting than the alternative one because we don't lose time to remake the experiment if we lost the data. All data are saved on a file that could serve like proof for a principle show or thesis.

Conclusion of the project

By three program we demonstrate that python is a powerful language that could help us in every experiment. We expose three ways of programming with different goal each time with just one experiment. For this experiment just a sensor, a LabJack and a computer are needed to obtain some measures.

An optimization still be possible for this project. The principle is to use a digital input instead of an analog input. The interest is that the analog input needs a simulation time for the measures. A digital input will improve the program by measuring data only when the tachymeter turns. We will not measure wasting data during its static phase.