Quickstart

Studer Python library for Xcom-485i

Date	:	25.06.2020
Version	:	V1.0.0

CONTENTS

1 INTRODUCTION

Wherever Python can run you can develop, whether it's a personal computer or a single board nano computer. Python is no longer to be presented and allows the realization of software applications in a reduced time. The Studer's libraries let you access each parameter and information containing into Studer's devices for monitoring and/or control. Examples for each case are provided, which can be used as a basic canvas or extended into a solution tailored to your requirements, let's your ideas become reality with Studer.

Ready to use, those libraries make you exploit the full potential of each Studer devices for your own developpment. Just Plug'n'Dev:

- Easy access to Studer gateways and gain full control over the Studer devices
- Forget about low-level configurations and time-consuming debugging
- Simply focus on your solution and adapt it to your needs
- Fast and easy development, thanks to Python language

The various use-case, this enables are:

- Create your own remote-control:
  - Build your own GUI
- Monitor data as you want
  - Log them into your preferred file format
  - Display them on your own dashboard
- Control the system as you need
  - Adapt it easily with your environment
  - Regulate it with your own algorithm
- Set up laboratories to educate students on energy production, conversion and storage
- Adapt pre-existing shared solutions
  - Get inspired from the community
  - Be efficient
- Be part of the community
  - share your DIY application

From a simple Python script reading the battery voltage to a more complex fully build application every case is possible; from plug'n'play to plug'n'dev every hobbyist, enthusiast, developer, integrator, teacher, student, etc… will find an answer with Open Studer.

2 STEP-BY-STEP START

This quickstart will guide you through the process of setting a communication between your Xcom-485i device and a controller, such as a personal computer, using the *xcom485i* Python library with one IDE (integrated developpement environnement).

The usefull links are:

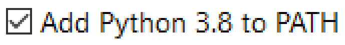
- General documentation : https://www.studer-innotec.com/en/downloads/
  - Then in *Software and updates* in *Communication protocols Xcom-485i*
- Source code and examples : https://github.com/studer-innotec/xcom485i
- Library documentation : https://xcom485i.readthedocs.io/en/latest/index.html

Follow the steps presented below to run your first example with the *xcom485i* Python library with *PyCharm* IDE. Other IDE are also working very well.

2.1 INSTALL PYTHON

Go to https://www.python.org/downloads/ to download and install Python if you didn't already have it, Studer Innotec software development team uses currently version **3.6.8** (https://www.python.org/downloads/release/python-368/).

Please Note: During the installation select the checkbox mentioned below, for the sake of convienciencence. This allows you to lauch Python just by typing python in a terminal and not the full path to the executable.



☑ Add Python 3.8 to PATH

2.2 STUDER PYTHON PACKAGE

Open a terminal (Press ⊞+R and enter "cmd" on Windows) to type the following commands:

- First check your freshly Python install by running:

```
> python --version

Python 3.x.x
```

- Install Studer Xcom-485i Python package by running:

```
> python -m pip install xcom485i

Collecting xcom485i
```

- Check package installation by running:

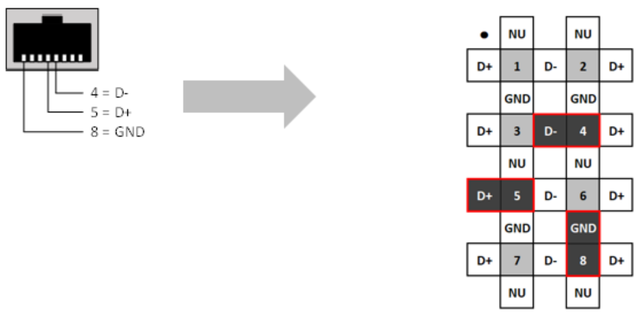
```
> python -m pip freeze

pyserial==3.4
uModbus=1.0.3
xcom485i==0.9.1
```

Note that version numbers are subject to change

2.3 HARDWARE SETUP

In order to communicate with the Xcom-485i gateway with your personal computer, or another controller, you should use a serial-to-USB dedicated cable and add a RJ-45 connector to the bare side. Studer Innotec software development team used those converters:

- https://www.ftdichip.com/Products/Cables/USBRS485.html
- https://www.delock.com/produkte/1011_Serial/64055/merkmale.html



Inside the Xcom-485i, a jumper array lets you select the pin assignment of the RJ-45 connector, as depicted above. This configuration is an example and you could do it as you want.

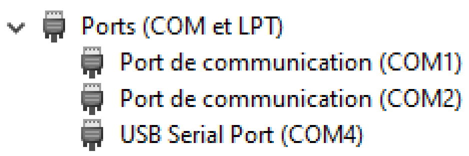
Be sure to set your desired Address Offset and Baud rate with the dip switches located inside the Xcom-485i gateway, configurations are shown in the tables below.

Position		Address Offset	Address range
1	2		
OFF	OFF	0	(0) 1 to 63
OFF	ON	32	(0) 33 to 95
ON	OFF	64	(0) 65 to 127
ON	ON	128	(0) 129 to 192

Position		Baud rate
7	8	
OFF	OFF	9'600 bps
OFF	ON	19'200 bps
ON	OFF	38'400 bps
ON	ON	115'200 bps

2.4 SOFTWARE SETUP

Once you have plugged the cable to the *External RS-485 Bus* interface of the Xcom-485i gateway, plug the USB side to your controller. You will have now to find the serial port interface. If you are using a computer running Windows, open the Device Manager (Press ⊞+R and enter "devmgmt.msc") and expand the *Ports (COM & LPT)* to find the *USB Serial Port (COMx)* where x is the serial port number.



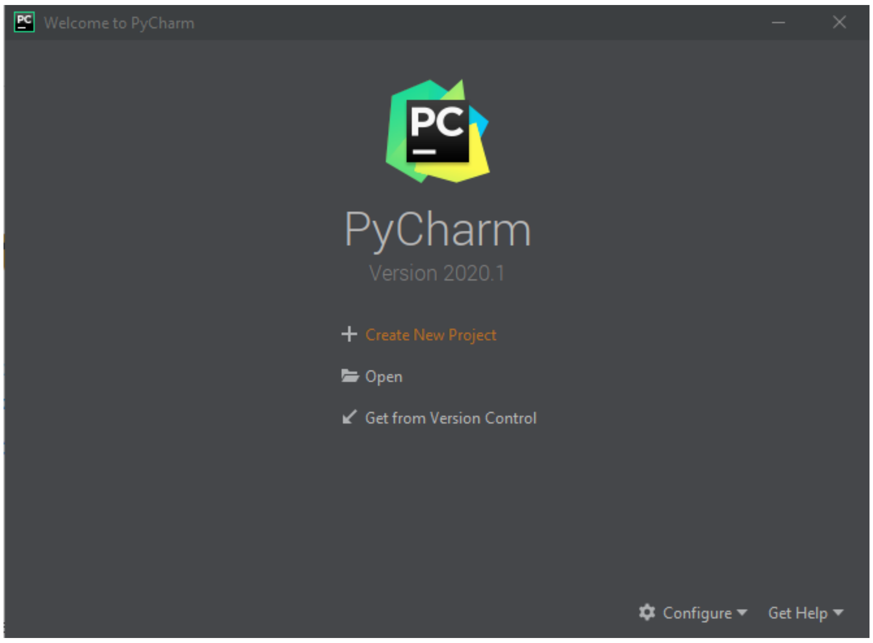
Ports (COM et LPT)
Port de communication (COM1)
Port de communication (COM2)
USB Serial Port (COM4)

2.5 Install an IDE

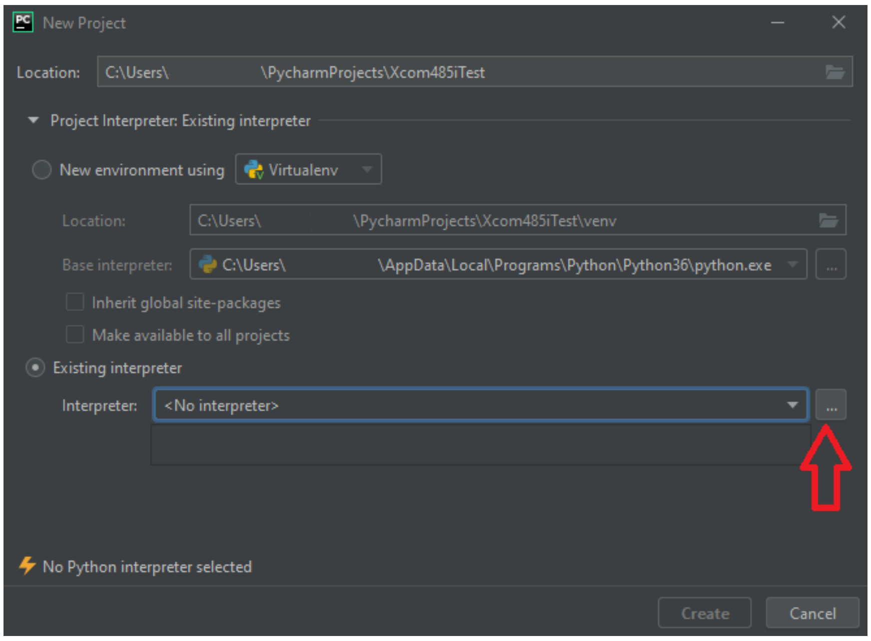
Go to https://www.jetbrains.com/pycharm/ to download and install *PyCharm* **Community** if you didn't have any IDE. You can use any IDE you like or just your command line interface if you are comfortable with it, but this document shows you how to run an example with *PyCharm*. No specifc configurations are required to install it.

2.6 Try an example with PyCharm

Once installed, you can run *PyCharm* and create a new project



Give it a name and select *Existing Interpreter*



Location: C:\Users\ \PycharmProjects\Xcom485iTest

Project Interpreter: Existing interpreter

New environment using Virtualenv

Location: C:\Users\ \PycharmProjects\Xcom485iTest\venv

Base interpreter: C:\Users\ \AppData\Local\Programs\Python\Python36\python.exe

Inherit global site-packages

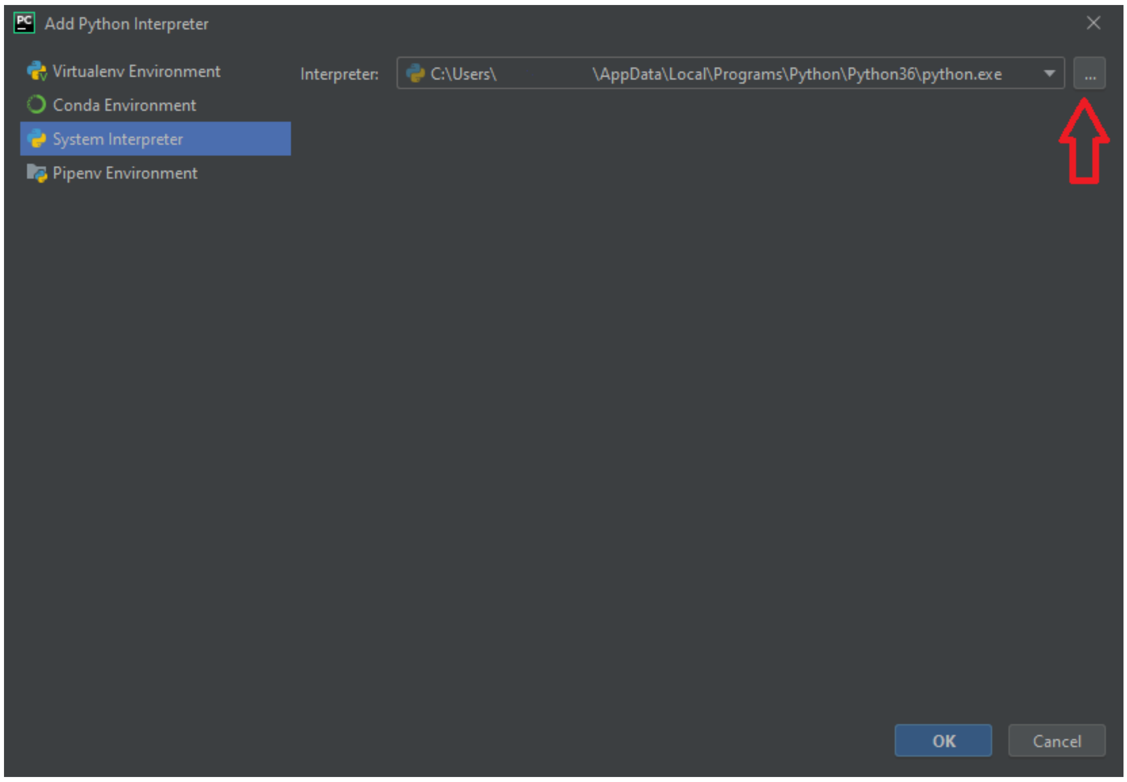
Make available to all projects

Existing interpreter

Interpreter: <No interpreter>

No Python interpreter selected

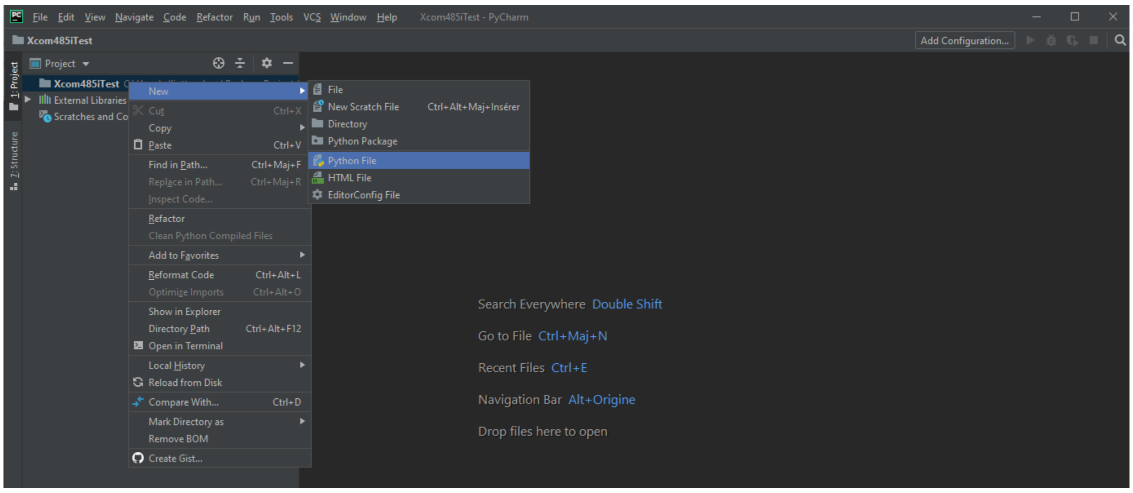
Choose *System Interpreter* and add the path where to find *python.exe*



Interpreter: C:\Users\ \AppData\Local\Programs\Python\Python36\python.exe

Click *OK* then *Create*

Select your folder's project, with the right click of the mouse select "New" then "Python file" and name it as you want.

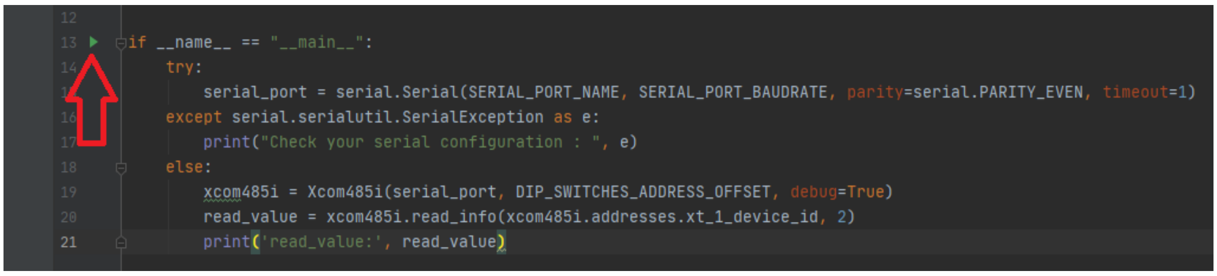


Go to https://github.com/studer-innotec/xcom485i/blob/master/examples/ex_read_info.py and copy the source code, paste it into your Python file inside *PyCharm*.

Please Note: You need to adapt the constants definitition at the top of the file with your specific serial port name, baudrate and address offset.

```
SERIAL_PORT_NAME = 'COM4'  # your serial port interface name
SERIAL_PORT_BAUDRATE = 9600  # baudrate used by your serial interface
DIP_SWITCHES_ADDRESS_OFFSET = 0  # your modbus address offset as set inside the Xcom485i device
```

Run your first example by clicking the green run button, this example read user info n°3001 on the first Xtender of the installation.



```
if __name__ == "__main__":
    try:
        serial_port = serial.Serial(SERIAL_PORT_NAME, SERIAL_PORT_BAUDRATE, parity=serial.PARITY_EVEN, timeout=1)
    except serial.serialutil.SerialException as e:
        print("Check your serial configuration : ", e)
    else:
        xcom485i = Xcom485i(serial_port, DIP_SWITCHES_ADDRESS_OFFSET, debug=True)
        read_value = xcom485i.read_info(xcom485i.addresses.xt_1_device_id, 2)
        print('read_value:', read_value)
```

You could go a little further and use the example presented below, don't forget to check the list of registers into the *Technical specification Studer Modbus RTU Appendix* document.

```
import serial
from xcom485i.client import Xcom485i

SERIAL_PORT_NAME = 'COM4'  # your serial port interface name
SERIAL_PORT_BAUDRATE = 9600  # baudrate used by your serial interface
DIP_SWITCHES_ADDRESS_OFFSET = 0  # your modbus address offset as set inside the Xcom485i device

BATTERY_REGISTER = 0  # Battery Voltage
TEMPERATURE_REGISTER = 2  # Battery Temperature BTS
STATE_OF_CHARGE_REGISTER = 14  # SOC
STATE_OF_XT_REGISTER = 98  # XT is ON or OFF

if __name__ == "__main__":
    try:
        serial_port = serial.Serial(SERIAL_PORT_NAME, SERIAL_PORT_BAUDRATE, parity=serial.PARITY_EVEN, timeout=1)
    except serial.serialutil.SerialException as e:
        print("Check your serial configuration : ", e)
    else:
        xcom485i = Xcom485i(serial_port, DIP_SWITCHES_ADDRESS_OFFSET, debug=False)

        batt_voltage = xcom485i.read_info(xcom485i.addresses.xt_1_device_id, BATTERY_REGISTER)
        print('\nBattery Voltage:', batt_voltage, 'Volts')

        batt_temp = xcom485i.read_info(xcom485i.addresses.xt_1_device_id, TEMPERATURE_REGISTER)
        if batt_temp == 32767.0:
            print('\nNo battery temperature info in the XT, there is no BSP')
        else:
            print('\nBTS Température:', batt_temp, '°C')

        soc = xcom485i.read_info(xcom485i.addresses.xt_1_device_id, STATE_OF_CHARGE_REGISTER)
        if soc == 32767.0:
            print('\nNo SOC info in the XT, there is no BSP')
        else:
            print('\nSOC:', soc, '%')

        xt_state = xcom485i.read_info(xcom485i.addresses.xt_1_device_id, STATE_OF_XT_REGISTER)
        if xt_state == 0.0:
            print('\nThe Xtender is OFF')
        else:
            print('\nThe Xtender is ON')

    serial_port.close()
```

That's it ! You can start build your own solution right now. If you are new to Python go to : https://docs.python.org/3.6/tutorial/ and don't forget that Python is based upon indentation.