Native Scilab kernel for Jupyter

- Personal, Contact and Educational Information

- Last Name: Consoni
- First Names: Leonardo José
- Age: 25 years old
- Nationality: Brazilian
- E-mail Address: consoni_2519@hotmail.com
- Phone Number: +55 16 99710-8370
- Git User Name: Bitiquinho
- IRC nickname: leonardojc
- LinkedIn: https://br.linkedin.com/in/leonardo-josé-consoni-1a825170
- Permanent Address: Rua Cesar Ricomi, 233, Apt 18, São Carlos - SP, Brazil
- Educational Level: 2nd year of Master's Degree in Mechatronics Engineering
- Educational Institution: University of São Paulo, Brazil
- CV (available in English): http://lattes.cnpq.br/3097674717351914
- Main Technical Skills: C/C++, Python, C#, socket programming, Git

Since the first years of my graduation and scientific initiation I have been learning and improving my skills in C/C++ and C#, getting used to the concepts of procedural and object oriented programming, system calls, IPC, among others.

My Final Term involved work with Internet Protocol communications, in which I spent nearly one year using and understanding the low-level BSD Sockets API. In this period, I started doing some coding in Python as well.

All my research and technical works, that continued up to now, in my Master's Degree, are related to the combination of singleplayer and multiplayer computer games with something called Robotic [Tele]Rehabilitation, the application of robots in physiotherapy sessions. Over the years, I produced a good amount of code (https://github.com/Bitiquinho) and had some articles in this field published with me as author or co-author [1-4].

During all this academic period, I developed a growing interest in open-source software in general, changing my main desktop system to Linux and always trying to use free alternatives for the tools I needed. Among them, I adopted Scilab (over Matlab, the dominant software in the engineering world) for rapid prototyping of algorithms and data visualization, learning more about it and following its recent evolution.

Contacting the Scilab developers for this year's GSoC, I found out that my knowledge of C++, Python, IP protocol and socket programming might be useful for developing one of their project ideas (https://wiki.scilab.org/Contributor%20-%20Jupyter) as described here.

[1] Consoni, L. J., dos Santos, W. M., & Siqueira, A. A. G. (2015). A distributed game system for robotic telerehabilitation. In 23rd ABCM International Congress of Mechanical Engineering. Rio de Janeiro.

[2] Gonçalves, A. C. B. F., Consoni, L. J., & Amaral, L. M. S. (2013). Development and Evaluation of a Robotic Platform for Rehabilitation of Ankle Movements. In 22rd ABCM International Congress of Mechanical Engineering. Ribeirão Preto.

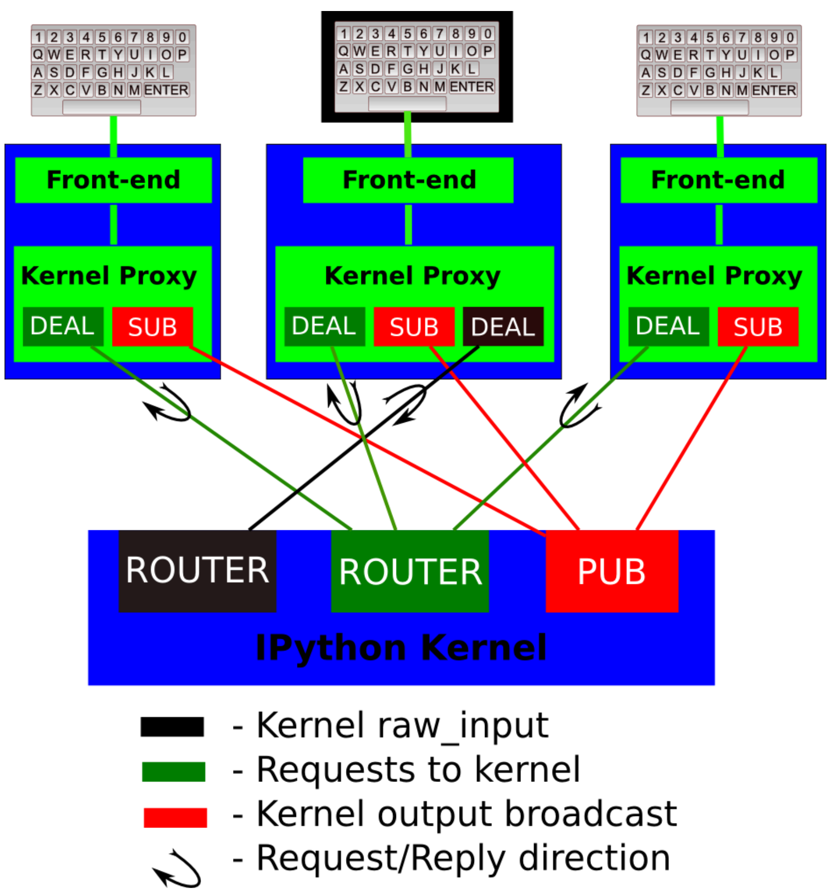
[3] Consoni, L. J., & Siqueira, A. A. G. (2015). Jogos computacionais para reabilitação robótica. In Encontro Nacional de Engenharia Biomecânica. Uberlândia.
[4] Consoni, L. J., Siqueira, A. A. G., Forner-cordero, A., Andrade, K. D. O., Joaquim, R. C., & Caurin, G. A. D. P. (2012). A Computational Game for Robotic Rehabilitation and Motor Learning Studies. In RoboControl 2012. Bauru.

- Participation requirements

- Compilation of git master branch (confirmation screenshot): http://mailinglists.scilab.org/file/n4033704/scilab_screen.png
- Patches for reported bugs: http://bugzilla.scilab.org/show_bug.cgi?id=14480
  http://bugzilla.scilab.org/show_bug.cgi?id=14375
  https://bugzilla.scilab.org/show_bug.cgi?id=7265

- Proposal Abstract

Jupyter project provides a way to perform scientific/mathematical computations with several different programming languages using the same user interface. It achieves that by distributing the processing between a front-end (Jupyter clients, like the web based Jupyter Notebook) and a back-end (Jupyter kernels, written in Python, R, Java, etc.), that communicate among themselves with standardized format messages, transmitted through ZeroMQ connection (https://jupyter-client.readthedocs.org/en/latest/messaging.html).



(Jupyter protocol messaging between clients and the conforming IPython kernel, from Jupyter Docs)

Besides the ability to switch across different kernels from the same common interface, another advantage of this approach is the possibility to run the two software components on different machines, e.g., offloading the heavier backend processing to a remote server while sending commands and visualizing data from a mobile device running the frontend.

For Scilab, there is currently a kernel (https://github.com/Calysto/scilab_kernel) that uses a compatibility layer (https://github.com/blink1073/scilab2py) with file access data exchange and numpy calls to communicate with clients through the Python reference implementation (https://github.com/Calysto/metakernel).

This Google Summer of Code proposal aims to adapt or rewrite the current suboptimal solution, to make it a truly native kernel, linking ZeroMQ messages directly to Scilab API.

- Benefits to Community

The Jupyter web interface is very attractive as an easy way to provide access to scientific tools for a large number of users, without the burden of applications setup, learning of different GUIs, and platform limitations. This can help popularize Scilab, by minimizing some of the common obstacles of its adoption by new users and/or developers.

Furthermore, as stated by Scilab developers, there is an actual and current demand from some industrial users for the availability of a Jupyter accessible Scilab engine. So, real use cases are already known or expected.

On my academic field, offering a Matlab-like free tool on pretty much any device with internet access could encourage engineering students to abandon the habit to rely on illegal copies of expensive commercial software, and create a new open source environment.

For me, particularly, I am interested in using this work, or the knowledge derived from it, on the creation of tools for my research area, e.g., for remote graphical visualization of physiotherapy patients data, for therapists and other health-care professionals.

- Deliverables

As described in the project Wiki entry:

- A fully working (command passing, error/warning reporting, graphics visualization, etc.) Jupyter protocol kernel, compatible with the yet to be released Scilab 6.0 API
- Some working and interactive Jupyter Notebook (or any other Jupyter client) demos
- Stable and easy to use implementation
- Additionally, a blog updated with my progress during the work period

- Availability

I intend to dedicate 35 hours a week to this project, distributed according to my academic compromises. I do not have restrictions for working on weekends, if necessary.

For communication with mentors, I could answer on e-mail, IRC, mailing lists, Skype, Hangouts, or any other suggested contact channels.

- Related Work

Some important references for this code are the current implementation of the Scilab kernel and its Python compatibility code:

- https://github.com/Calysto/scilab_kernel
- https://github.com/blink1073/scilab2py

- Proposed Timeline

- 22 April -- 22 May (Community Bonding Period): Read the available documentation, learn about the Scilab and ZeroMQ APIs and the Jupyter protocol architecture and message formats

- 23 May: Working Period Begins

- 23 May -- 29 May: Implement messaging (ZeroMQ communication) for shell ROUTER/DEALER and control sockets
- 30 May -- 05 June: Implement messaging for the PUB/SUB socket
- 06 June -- 12 June: Implement messaging for stdin ROUTER/DEALER sockets, heartbeating and custom messages
- 13 June -- 19 June: Allow kernel to handle more than one Jupyter client at a time (one thread per client)

- 20 June -- 26 June (Mid-Term Evaluations): Testing, bugfixes and documentation work for the current code

- 27 June -- 03 July: Process stdin ROUTER/DEALER socket messages (request raw input from frontends)
- 04 July -- 10 July: Process PUB/SUB socket messages (passing it down to the Scilab C++ API and returning the result messages)
- 11 July -- 17 July: Process shell ROUTER/DEALER socket messages (passing it down to the Scilab C/C++ API and returning the result messages)
- 18 July -- 24 July: Process control messages (high priority client requests)
- 25 July -- 31 July: Process heartbeating and custom messages
- 01 August -- 07 August: Write test demos, fix bugs, fully working kernel
- 08 August -- 14 August: Write documentation
- 15 August -- 23 August: Buffer time

- 23 August: Final Mentor Evaluation Submission